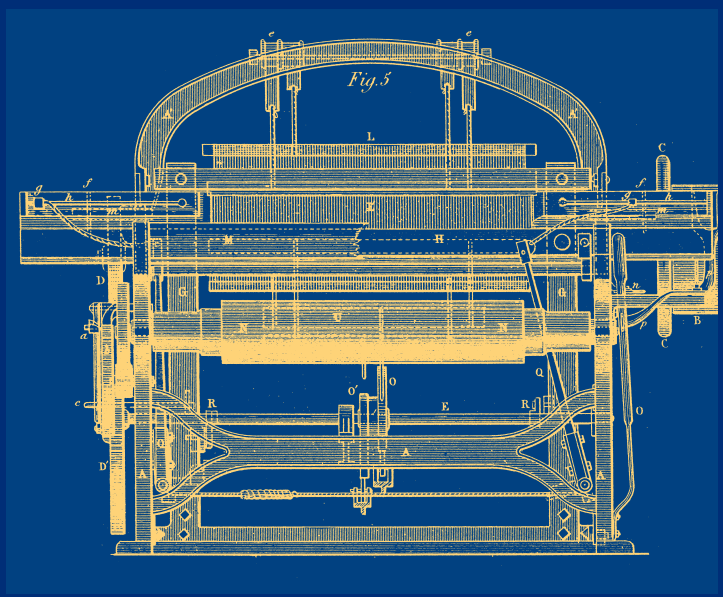
DECEMBER 2007 VOLUME 32 NUMBER 6

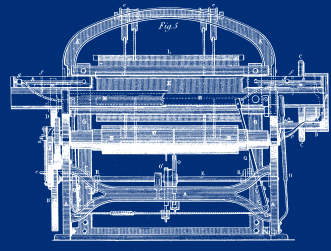
;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Sytems Association

USENIX Upcoming Events



2008 Linux Storage & Filesystem Workshop

Co-located with FAST '08

FEBRUARY 25–26, 2008, SAN JOSE, CA, USA

http://www.usenix.org/lsf08

Submissions due: December 3, 2007

6th USENIX Conference on File and Storage Technologies (FAST '08)

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

FEBRUARY 26–29, 2008, SAN JOSE, CA, USA

http://www.usenix.org/fast08

2008 ACM International Conference on Virtual Execution Environments (VEE '08)

Sponsored by ACM SIGPLAN in cooperation with USENIX

MARCH 5–7, 2008, SEATTLE, WA, USA

http://vee08.cs.tcd.ie

Usability, Psychology, and Security 2008

Co-located with NSDI '08

APRIL 14, 2008, SAN FRANCISCO, CA, USA

http://www.usenix.org/upsec08

Submissions due: January 18, 2008

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

Botnets, Spyware, Worms, and More

Co-located with NSDI '08

APRIL 15, 2008, SAN FRANCISCO, CA, USA

http://www.usenix.org/leet08

Paper submissions due: February 11, 2008

5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 16–18, 2008, SAN FRANCISCO, CA, USA

http://www.usenix.org/nsdi08

The Sixth International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)

Jointly sponsored by ACM SIGMOBILE and USENIX

JUNE 10–13, 2008, BRECKENRIDGE, CO, USA

http://www.sigmobile.org/mobisys/2008/

Paper titles and abstracts due: November 26, 2007

2008 USENIX Annual Technical Conference

JUNE 22–27, 2008, BOSTON, MA, USA

http://www.usenix.org/usenix08

Paper submissions due: January 7, 2008

2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)

Co-located with USENIX Security '08

JULY 28–29, 2008, SAN JOSE, CA, USA

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

Co-located with USENIX Security '08

JULY 29, 2008, SAN JOSE, CA, USA

17th USENIX Security Symposium

JULY 28–AUGUST 1, 2008, SAN JOSE, CA, USA

http://www.usenix.org/sec08

Paper submissions due: January 30, 2008

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

Sponsored by USENIX and SAGE

NOVEMBER 9–14, 2008, SAN DIEGO, CA, USA

8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)

Sponsored by USENIX in cooperation with ACM SIGOPS

DECEMBER 8–10, 2008, SAN DIEGO, CA, USA

http://www.usenix.org/osdi08

Paper submissions due: May 8, 2008

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

contents

;login:

VOL. 32, #6, DECEMBER 2007

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Casey Henderson
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.

$85 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

WELCOME TO THE NINTH SECURITY edition of *;login:*. As one of the authors in this issue mentions, the existence of a focus issue on security is evidence that security is still a concern. Of course, we all have plenty of first-hand experience, and evidence, that our computer security problems remain unsolved. At best, we have moved the goalposts a bit and redesigned the playing field. But our computer systems appear to be as insecure as ever.

Not that there hasn't been progress. Go back to the 1990s, when UNIX and Linux systems included exploitable services, running as root, out of the box. Over time, Linux in particular has improved in default security, a very good thing indeed (with BSDs way ahead in security by the late 1990s). And open source programs in general may not be bug-free, but their support teams can boast of being many times faster at releasing patches than most of their commercial counterparts.

For reading that at least brings a smile to my face every time, I can recommend the Symantec Internet Security Report [1]. You might think me cynical (and you would be correct) when I write this, but the stats included in their executive summary are incredible. For example, the good ol' USA is number one in a number of areas other than pollution, but Beijing is fast catching up. The U.S. boasts of more attack and malicious activity, phishing servers, and command and control systems, while being both the greatest source and the largest target of spam. Beijing, however, has more bot-infested computers than any other city in the world, and China has 29% of the world's bots as well. So the Chinese have another interesting problem to deal with, beyond the pollution created by their rapidly expanding economy.

The Symantec report is hardly a source of encouraging news, but there are some bright spots. HP has exchanged places with Sun (or is it Java?) as the slowest vendor to release a patch to a vulnerability (112 days). Microsoft was the winner for having the most enterprise-level unpatched vulnerabilities for the first six months of 2007, unchanged from winning this contest in the last half of 2006. Okay, so I am still being really cynical.

Apple does well in the report, with fewer reported vulnerabilities in Safari than IE or Mozilla but more than Opera. Apple can also boast of having the shortest time to patch browser bugs, beating

out the Mozilla Foundation this time, with an average window of exposure of four days (Mozilla's was only two days in the previous report). But speaking of Apple . . .

Sour Apples

By the time you read this, Apple's "bricking" of unlocked iPhones will be old news. You will remember the flurry of hacking activities that followed the release of the iPhone in late June and the discovery that the iPhone (and iPod) actually run Mac OS X. Deeper inspection of the iPhone revealed that all applications *run as root*, evidence either that the iPhone was rushed to market or that Apple programmers are extremely lame. I know the second conjecture is not true, so I must presume the first.

Many people I know have raved about the design of the iPhone, and they seem to be very happy with the cell phone as it exists. But other people I know were quite anxious to extend the capabilities of the iPhone to include shell access, an SSH client, WiFi scanner, and other tools that Apple neglected to include, even after several updates. Update 1.1.1 disabled all added applications, and changes to the firmware of the iPhone disabled the cell phone functionality, "bricking" the phone, of those iPhones that had been unlocked (allowed to work with the SIMs from carriers other than AT&T).

I don't intend to muse about the propriety of Apple destroying the functionality of a device people had purchased (at an inflated price). Instead, consider the implications of update 1.1.1. With this update, Apple, belatedly, added protection to the iPhone's firmware that prevents unlocking. Why wasn't that in the first version? To the OS that runs your cell phone—any cell phone, to my knowledge—the radio looks like a modem. Remember how you commanded modems back in the day, using AT commands? Cell phone radios work similarly. So the cell radio appears to the OS as a serial device, and any application that runs as root can access that serial device and send it AT commands.

Depending on the cell radio, the set of commands that can be issued may include changing the radio frequencies that the phone uses, as well as the power used when transmitting. In the U.S., the FCC (Federal Communications Commission) regulates the use of the radio spectrum, and it takes a dim view of putting this level of flexibility, and control, into the hands of cell phone users. Carriers are also loath to allow their customers to change settings, because in doing so they may disrupt the operations not just of the adjusted phones, but also of other cell phones operating in the area. The cell radio interface must be protected.

This brings us back to the iPhone, where all applications run as root. That implies that any application installed on the iPhone may have access to the cell radio. I like to see well-designed security, particularly when a device must comply with national laws, and the iPhone appears to fall far short of this goal. But with this design, it should be obvious that Apple *had to* prevent third-party applications from running on the iPhone.

Apple has had other security issues this year. The mDNSResponder, a friendly service that broadcasts information about your Mac to any device that can listen, as well as accepting information from any system in range, has had at least two root exploits this year (one that has not been reported). I checked to see if the patched mDNSResponder was still running as root after the patch, and it was (I've since disabled it, so can't be sure if this is still true). But mDNSResponder, which, according to the Darwin source code,

only needs to be able to do *one thing* as root, does everything as root. Notice a pattern here?

Part of what mDNSResponder (a.k.a. Bonjour) does is parse variable-length input broadcast from other systems, and parsing variable-length input happens to be the exact same issue that plagued SNMP and other ASN.1 notation-based systems in the not distant past. I'd like to see Apple change how mDNSResponder works by separating out the code that must run as root, advice that dates back to Matt Bishop's article from 1987 that provides advice about writing set-user-id programs [2]. Although Matt focused on setuid, the notion of partitioning a program into two parts, a privileged part and a larger part that runs without privileges, has become part of accepted security practice today.

Then there was the Airport exploit, during BlackHat 2006. This may appear to be old news, but you can now read what had been hidden from view about this kernel-level exploit [3]. At the time, Apple vehemently denied that the exploit actually worked on Apple MacBooks [4], but Maynor explains that he was perplexed when a nontargeted MacBook crashed several minutes after he had conducted fuzzing using WiFi beacon and probe packets. His article is useful to anyone interested in understanding panic.log files found on Mac OS X systems after kernel crashes.

I much prefer that vendors be honest about the security of their products, even if the news is bad. History, for example that of Windows NT, has shown that no matter how secure you claim your system is, the more people who use it, the more bugs will be discovered. Mac OS X does have some advantages over NT, as it represents a much more mature technology than NT was in 1998. But there are other more disturbing parallels between Microsoft and Apple, such as the lack of any documentation about how security mechanisms work in Mac OS X. Like other security professionals, I prefer to understand how a mechanism works and see how it is implemented, rather than to trust a vendor who simply states, "Don't worry, it's secure." Where have I heard that before?

The Lineup

As a tribute to the advances in exploit technology, we begin this issue with two articles about P2P (peer-to-peer) command and control as used in recent Windows trojans. I wanted to write "bots" here, but Dave Dittrich has convinced me that bots have that name because they are controlled via IRC servers, and I have to agree with him. The first article, by Dave Dittrich and Sven Dietrich, explores how remote-control trojans have evolved over time, moving from clunky handler/agent command and control to P2P. The change to P2P makes trojans more difficult to write, but also much more difficult to detect, and nearly impossible to trace back to the owner of a collection of trojans. The authors do provide some advice about what you can do to detect and traceback trojans in your own networks.

The next article, by Stover et al., dissects two recent trojans that use P2P for command and control. Stover and associates focus on the network communications of both trojans, rather than disassembling them, to reveal how they behave when viewed from outside the box. In the case of Storm, there is one configuration file, used to seed the list of peers when the trojan begins to communicate. With Nugache, the list of initial peers is embedded in the executable, and communications are further hidden through the use of encryption between each peer.

Tal Garfinkel and Andy Warfield entertain us next, with an article describing current and future security uses of virtualization. Being an old-school kind

of guy, I like the idea of isolating services, that is, running a single service per system. Virtualization allows us to do something similar, that is, run one service per guest operating system, allowing much more efficient use of hardware, along with the improvements in system management possible via virtualization. Tal and Andy also point out future uses of VMM—for instance, running anti-virus software and HIPS within the VMM. Currently, malware commonly disables AV and hides itself using rootkit technology, so the notion of moving the protection outside of the guest OS into the VMM makes lots of sense. I liked the idea, but did point out during the review process that if they move the protection into the VMM, that will change the focus to attacking the VMM. Within a week, a vulnerability that allows executing commands as root within domain 0 (the VMM OS) was announced, allowing me to feel like a prophet [5].

Gernot Heiser offers his own view of the future of OS security. Gernot has been working on the L4 microkernel for many years, and I recently learned that seL4, a secure version of L4, was going through the process of being formally proven. That is, the design of seL4 would be verified via theorem provers that it matches its specifications. I had challenged Gernot to prove to me that proving any OS really matters to anyone, and this article is his response. To be honest, I was playing devil's advocate in this case. I would like to see small and verified OSes in the future of servers, desktops, and embedded systems, and seL4 is a great leap forward.

I had lunch with Gary McGraw this summer, while we were at USENIX Annual Tech in Santa Clara. Gary waxed enthusiastic about his new book, *Exploiting Online Games*, and we decided that an interview-style article would be an interesting addition to this issue. I also wanted to bring out some of the topics we had in our own discussions that don't appear in his book or in the summary of the IT he gave at USENIX Security (see the summaries section of this issue, and watch the video, listen to the MP3 of the talk, and view his presentation slides on www..usenix.org/events/sec07/tech/).

Mike Rash, who has a new book out about Linux firewalls, has explained a couple of the tools he has written that augment netfilter: fwsnort, which allows you to add snort-based rules to Linux firewalls, so you can block TCP connections that match these rules, and psad, which performs passive system identification, using the same database as p0f, but using iptables log messages. If you want to learn more about netfilter and iptables, as well as Mike's tools, I recommend reading this article.

David Blank-Edelman has fully joined our conspiracy to discuss security in this issue. David writes about using Perl modules to embed the use of nmap and p0f. I had heard about embedding nmap into Perl scripts, a technique that adds a lot to the usability of nmap in large scans. Using p0f within Perl was a new concept for me.

Dave Josephsen also plays along with the security theme. Dave explains how to analyze netflows to monitor your networks and to improve their security. Steve Romig wrote about the use of netflow logs in security way back in the September 1999 [6] issue of *;login:*, and I'm glad that Dave decided to bring us up to date on this topic.

Heison Chak was invited to the security table, but he chose instead to explain speech recognition software as used in Asterisk. Heison, like most of us, finds little charm in most automated phone systems, and he shows us how to embed speech recognition within Interactive Voice Response (IVR) systems that will, hopefully, be well-designed and not drive callers to acts of anger and frustration.

Robert Ferrell, staying with our security focus, attacks the naming conventions used by AV companies. Not content with boring names such as Win32/DEL.100907.ZZA, Robert has some interesting suggestions for the worms and trojans of the near future.

In the book reviews section, Elizabeth Zwicky starts off with a quick review of the new edition of *The Practice of System and Network Administration* and then covers a book on designing reliable sites. Sam Stover, not content with just writing an article, reviews the tome named *The Art of Software Security Assessment* and comes away very favorably impressed. Finally, Ming Chow reviews the book I mentioned earlier, *Exploiting Online Games*, with unsparing accuracy.

We have many conference reports in this issue, starting with USENIX Security in Boston. I wrote some about cell phone issues in this column. There was a panel, an Invited Talk (about Symbion viruses), and a paper about cell phone security at the symposium, and you will find in-depth summaries about these talks and others here. Five workshops were co-located with USENIX Security, and we have summaries from three of them here: USENIX/ACCURATE Electronic Voting Technology Workshop, First USENIX Workshop on Offensive Technologies, and MetriCon 2.0: Second Workshop on Security Metrics. Finally, Matt Bishop sent us summaries of the 2007 New Security Paradigms Workshop.

Afterthoughts

I just tried imagining what it would be like if we had secure computer systems. Sure, I would be out of a job, but I like to consider the benefits to the wider world. If our computers were secure, we could use online banking with little fear (although the banks could still screw up our accounts). We could communicate with complete assurance that no one could read our email (and then there's the NSA). We would no longer have to pay fees to AV companies whose software is, at best, only partially able to protect systems from malware. We could view the Web without fear of our systems being infected with spyware, adware, or trojans (porn without fear!). And we could trust that our personal data, whether financial or health-related, would remain secure on the servers where it is stored.

Honestly, at this point in time, imagining a future of secure operating systems and software is hard to do. But perhaps, having already used a sports analogy, I can use a car analogy as well. Just imagine what cars were like in the 1920s: Horseless carriage was a good description. Today's cars are both safer and a lot more reliable than cars from just 20 years ago. Perhaps when software is 100 years old, it will reach the level of reliability that we see in cars today.

I just don't want to have to wait that long.

REFERENCES

[1] Symantec Internet Security Report: http://www.symantec.com/business/theme.jsp?themeid=threatreport (see pp. 3–8 for the executive summary).

[2] Matt Bishop's paper on writing setuid programs: nob.cs.ucdavis.edu/bishop/secprog/1987-sproglogin.pdf.

[3] David Maynor's article about how he uncovered the flaw in the Apple Airport driver and created a proof of concept exploit: http://uninformed.org/index.cgi?v=8&a=4&p=2.

[4] "Apple Fixes Mac Wi-Fi Flaws": http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1217510,00.html.

[5] Bug in pygrub used within Xen: http://bugzilla.xensource.com/bugzilla/show_bug.cgi?id=1068.

[6] S. Romig, M. Fullmer, and S. Ramachandran, "Cisco Flow Logs and Intrusion Detection at the Ohio State University": http://www.usenix.org/publications/login/1999-9/index.html.

DAVID DITTRICH AND SVEN DIETRICH

command and control structures in malware

FROM HANDLER/AGENT TO P2P

Dave Dittrich is an affiliate information security researcher in the University of Washington's Information School. He focuses on advanced malware threats and the ethical and legal framework for responding to computer network attacks.

dittrich@u.washington.edu

Sven Dietrich is an assistant professor in the Computer Science Department at the Stevens Institute of Technology in Hoboken, NJ. His research interests are computer security and cryptography.

spock@cs.stevens.edu

INTERNET ATTACK TOOLS HAVE evolved, similarly to the way that operating systems and applications themselves have evolved. We will focus on one particular aspect, the mechanisms to allow control of the increasing number of hosts being exploited. The result is an increase in efficiency that allows attackers today to rapidly marshal the computing resources of millions of personal computers across the globe in order to use them for a wide range of criminal activities. In particular, we consider the impact of P2P command and control mechanisms and other features of distributed attack tools that result in a distributed attack network resilient to current methods of detection, monitoring, and takedown by any individual defender or rival.

We look at the impact these structures have on incident response and muse about the trends for the years to come.

Structure of Distributed Intruder Tool Networks

One of the central problems of a distributed intruder tool network is the topology of the network and the implications for traffic monitoring, enumeration of the entire network, and traceback to the attacker. The latter problem, traceback, is often made more difficult through the use of *stepping stones* (hosts used to redirect connections and "bounce" off a third-party system or network), which leave attackers many "hops" away from their victims.

SPECIFIC DISTRIBUTED SYSTEM NETWORK STRUCTURES

Since the advent of distributed intruder tools [1, 9], there have been three principal structures employed by distributed system intruder tools: *handler/agent* relationships, central command and control mechanisms (e.g., IRC channels and *botnets*), and *P2P* networks.

To compare and contrast the various structures of the three principal attack network topologies, we depict each using the same circular layout for highlighting the command and control relationship with attack agents in Figures 1, 2, and 3. The number of nodes (30) in these diagrams was chosen

simply to be manageable in visualizing these attack network structures. In each diagram, nodes depict computer hosts and edges depict the direction of establishing connections, along with two stepping stones and the attacker's primary host (the three nodes that line up on the left of the diagrams). Figures 1 and 2 each has controlling hosts (handlers or IRC servers), whereas Figure 3 does not involve central handlers of either type. To illustrate how traceback is performed from a single known attack agent, one agent or peer (e.g., node A1 or P1) has been selected and all incoming/outgoing connections associated with that host are shown by bold black arrows; all other arrows are in grey.

Typical real-world distributed networks can range from several hundred up to several hundred thousand at a time, with a medium to large botnet in 2006 being on the order of 10,000 hosts. The controversy on how to accurately enumerate botnets remains [12].

HANDLER/AGENT NETWORKS

The UNIX-based *Stacheldraht* DDoS tool employed the *handler/agent* model of command and control. In the handler/agent model, the attacker uses a computer with a special malware command/control program (the handler) that coordinates a set of hosts running a different malware attack component (the agents). The attacker connects to the handler, optionally through stepping stones, and controls the attack network. (There are typically no connections between the handlers themselves.) This topology is depicted in Figure 1.

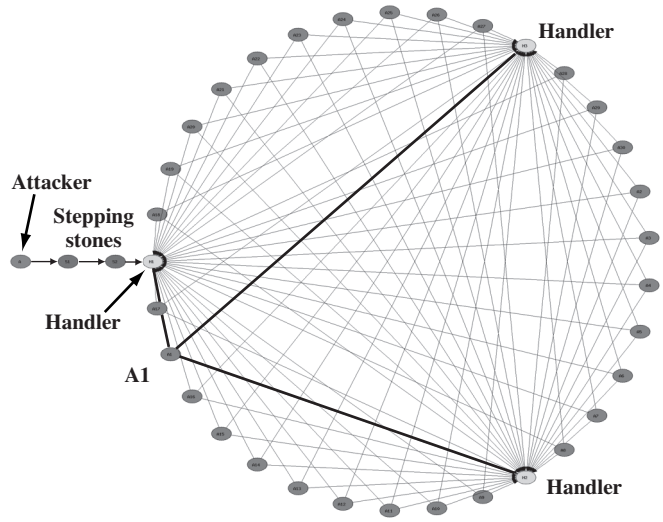


FIGURE 1: HANDLER/AGENT ATTACK NETWORK

In most handler/agent DDoS networks, the agents had a predefined list of handlers compiled into the executable program image to which the agent would initially connect on startup. The list would often include at least three handlers, in case one or two had been found and disabled. If all of the handlers had been disabled, the agent would become "lost" and cease to be of use to the attacker, unless or until it was updated with a new copy of the agent that included new handlers. For this reason, a backdoor was sometimes also set up on the compromised host. Some tools allowed for dynamic updating of the handler hosts, but that information would get lost upon restart of the tool. The update affected the memory copy but not the binary stored on disk. For malware only existing in memory and spreading copies of itself, this would of course propagate the correct copy.

If a defender or rival group were to identify an agent (e.g., by noticing the initial attack through which the agent was installed or when local network bandwidth was exhausted during a DDoS attack), it would be easy to trace connections directly back to the handler(s). Figure 1 shows these command and control connections as bold lines from node A1 to the three handler nodes. In the same figure, all three handler connections are shown; however, in practice only one of them may be active at a time. One could then get in touch with someone responsible for security operations at the site that hosted the handler and then monitor network traffic to identify the agents being controlled by that handler. Alternatively, one could seize the host, copy the file system, and recover the list of agents being maintained by the handler. Later versions of handler/agent malware used encryption or obfuscation of the list of agents to make it harder to identify them through file system analysis. Monitoring of network traffic at the site hosting a handler would also allow identification of incoming command and control connections from the stepping stones, allowing traceback to begin toward the attacker.

Even with encryption being used, it was easy to identify the role of handler, agent, and stepping stone in this structure and to act accordingly. (In 1999 and 2000, handler/agent networks of several hundred hosts could be identified from the point of one attacking agent, and the entire network could be identified and taken down in several hours, in the best case.)

One of the primary limiting factors in handler/agent networks resulted from the use of TCP sockets connecting each agent to its handler. Normal UNIX systems intended for desktop use would have their default kernels configured to support a very limited number of concurrently open file handles. Rarely if ever did an attacker stumble upon a highly tuned host that could handle more than the typical default of 1024 file handles, which meant these type of attack networks could not exceed just over 1000 agents total.

Another drawback to handler/agent DDoS tools was that each had its own specialized command and control protocol, which the author would have to maintain in addition to adding code to perform the new functions. This prevents interoperability of handlers and agents from different attack tools. For example, a *Stacheldraht* handler could not be used to control *trinoo* agents, or vice versa.

INTERNET RELAY CHAT NETWORKS

Internet Relay Chat (IRC) networks can scale much larger than the handler/agent model will allow, owing to the use of peering servers that can be spread around the globe, each capable of handling a much larger number of concurrent connections (as described in the previous section), in turn relaying chat messages from server to server. This is depicted in Figure 2.

IRC not only serves as a client/server communication network but also provides a defined protocol that can be used by special programs designed to identify specific individuals and commands that appear in IRC channels. The programs that identify these commands and act on them are called *bots*, which is short for *robot*. A set of bots acting together as a group in a single IRC channel is referred to as a *botnet*. (The "topic" of the channel, a string that usually advertises to humans the central chat subject focused on in the channel, sometimes serves as a default command for the bots to act upon when initially joining the channel.) Bots can be implemented in one of two ways: (a) by loading new modules to an existing IRC client (e.g., TCL command scripts executed by the *mIRC* client [7]) or general-purpose bot (e.g.,

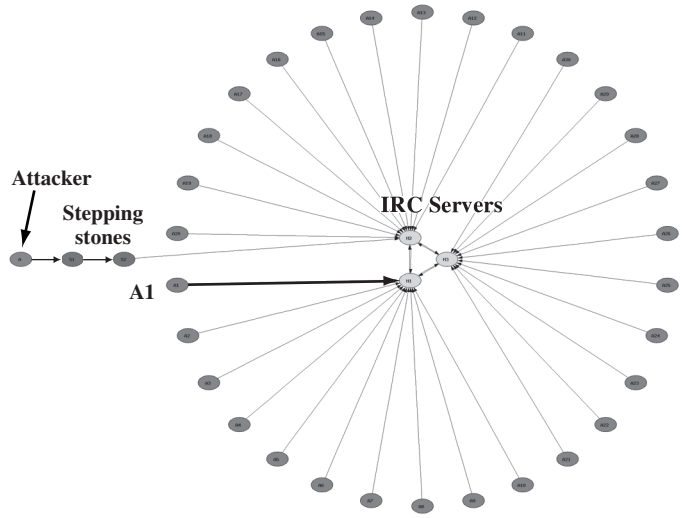


FIGURE 2: IRC ATTACK NETWORK

eggdrop [16]) or (b) by programming a new bot that "speaks" the IRC protocol and allows direct connection to an IRC server (e.g., knight and Agobot.)

IRC operators (IRC ops) are well aware of botnets and the disruption they cause, so they are constantly on the lookout for thousands of "users" showing up in a short period of time in a given channel or moving from channel to channel *en masse*. More advanced bot herders are skilled at moving bots around, sheltering or combining bots as needed to perform the acts the attacker wishes (such as sending spam, spreading malware, retrieving product keys, or performing distributed denial of service attacks). Bot herders may employ one or more tactics: use of dynamic DNS entries, or short TTLs in DNS records, to point to specific servers for short periods of time (one variation of this is known as *fast flux* [5]); having all or some bots switch IRC channels (*channel hopping*); having all or some bots switch IRC servers (*server hopping*); being redirected to an IRC server or port by downloading a file with HTTP protocol; use of proxies that use port numbers other than the standard IRC server ports (e.g., 6667/tcp); and avoiding the standard IRC networks altogether by setting up customized botnet-tuned IRC server programs on compromised third-party hosts (often called "rogue IRC servers").

The principal difference from a response perspective between the handler/agent and the IRC command and control structure is the fact that the three IRC server nodes in Figure 2 themselves act similar to stepping stones, so that any node (such as A1 in Figure 2) is only connected to one, while the last stepping stone used by the attacker can be connected to another node, preventing direct traceback from agent to handler to stepping stone. If, for example, the IRC bots were all using encryption for the traffic going over the IRC channel, it would be nearly impossible to trace a connection from our known bot back to the final stepping stone, because there may be hundreds of thousands of connections in total across all three IRC servers and no information that ties any one flow to other flows. Even when encryption is not being used, many bots obfuscate their identity in the chat channel, preventing a direct association between a bot's name and its actual IP address.

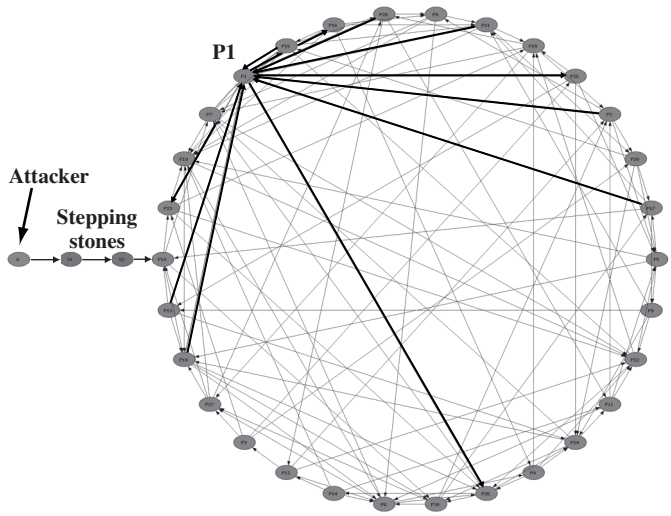


FIGURE 3: SAMPLE OF IRC BOT COMMAND TRAFFIC

```
Feb 19 13:36:40  <~foobar>          FRA|XXXXXX .login toldo
Feb 19 13:36:40  < FRA|XXXXXX>  [r[X]-Sh0[x]]: .:( Password Accettata ):. .
Feb 19 13:36:41  <~foobar>          .opencmd
Feb 19 13:36:42  < FRA|XXXXXX>  [CMD]: Remote shell already running.
Feb 19 13:36:54  <~foobar>          .cmd mkdir c:\windows\system32\kernel
Feb 19 13:36:55  < FRA|XXXXXX>  mkdir c:\windows\system32\kernel
Feb 19 13:36:56  < FRA|XXXXXX>  C:\Documents and Settings\KiM>
Feb 19 13:37:00  <~foobar>          .cmd cd c:\windows\system32\kernel
Feb 19 13:37:01  < FRA|XXXXXX>  cd c:\windows\system32\kernel
Feb 19 13:37:02  <~foobar>          .cmd dir
Feb 19 13:37:03  < FRA|XXXXXX>  C:\WINDOWS\system32\kernel>dir
Feb 19 13:37:04  < FRA|XXXXXX>   Le volume dans le lecteur C n'a pas de nom
Feb 19 13:37:05  < FRA|XXXXXX>   Le numero de serie du volume est A443-2CAF
Feb 19 13:37:07  < FRA|XXXXXX>   Repertoire de C:\WINDOWS\system32\kernel
Feb 19 13:37:09  < FRA|XXXXXX>  19/02/2005  13:37          .
Feb 19 13:37:10  < FRA|XXXXXX>  19/02/2005  13:37          ..
Feb 19 13:37:11  < FRA|XXXXXX>     0 fichier(s)                0 octets
Feb 19 13:37:13  < FRA|XXXXXX>     2 Rep(s)    8'990'302'208 octets libres
```

FIGURE 4: PEER-TO-PEER ATTACK NETWORK

PEER TO PEER NETWORKS

Visually, there is a clear difference between the P2P model and the first two topologies. Figures 1 and 2 show a distinct symmetry, whereas Figure 4 shows a randomness between the number and the direction of connections between peers. In the P2P model, all attack agents form a randomly connected network, where no single host or network of hosts is responsible for central communication. Unlike either of the two previous models, the P2P model does not need any central host or hosts responsible for command and control, or even for joining the P2P network. The SpamThru Trojan [14], which also uses a P2P model for some command and control, also employs a central server for its spam templates. This is a hybrid of the IRC and P2P models. (See the analysis of the Storm and Nugache trojans in this issue, pp. 18–27.)

Commands are retransmitted through the P2P network a limited number of times, enough for all peers to see and act on the command. The attacker is able to connect to any of the peers using a special client program and initiate

commands, which are then relayed throughout the P2P network. Although we have included the two stepping stones in each diagram for consistency, the stepping stones become unnecessary in the case of P2P networks. In fact, the P2P network becomes both the stepping stones and the command and control channel at the same time, very effectively hiding the IP address of the originator of the commands as well as the complete set of peers. Responses are similarly routed through the network until they reach the intended recipient (or are dropped because they have exceeded a "time-to-live" threshold).

Use of P2P command and control in malware has been attempted before and thought of as early as 2000 [17], but with limited success until 2006. The 2002 Linux Slapper worm source code [3] claimed to use a P2P algorithm that could support 16 million peers; however. the propagation itself was so noisy that the P2P mechanism was never really tested. In 2003 Agobot began to see widespread use. Agobot included a rudimentary P2P mechanism that does not appear to have been popular. This assumption is based on the fact that over 20 versions of Agobot/Phatbot source code analyzed by the authors had no difference at all in any of the P2P related source files, whereas other sections of the code underwent regular and extensive enhancement and bug fixing. In 2004, a version of Phatbot used the WASTE protocol [10] to communicate among the bots. According to Stewart's Phatbot analysis [6], the encryption capabilities of WASTE were removed to avoid either the problem of key distribution or the weakness of use of a commonly shared key and so that the largest usable P2P network could not exceed much more than 50 peers. In practice, Agobot/Phatbot is almost always controlled using clear-text IRC. When this occurs, all command and control traffic is visible, as is seen in Figure 3. In all these instances, the goal of assembling very large P2P botnets was not realized; however, use of P2P as a means of *spreading* has been successful. A 2006 report indicated that 35% of successful malware infections involved spreading through email, P2P networks, and instant messaging [8].

In the arena of file sharing, specifically anonymous file sharing, P2P mechanisms have also been pursued [11]. Looking at just those P2P networks that use some form of public key exchange and strong cryptography, we see that there are at least five such P2P networks in development.

FIGURE 5: POSSIBLE CONNECTIVITY GRAPH FOR TWO P2P NODES

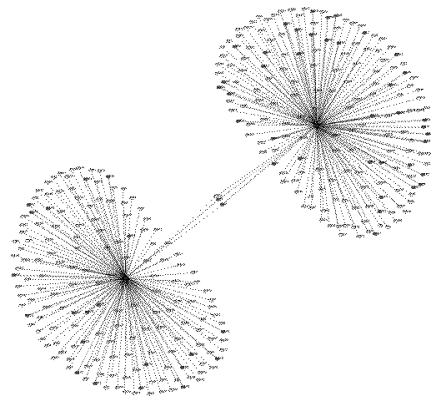
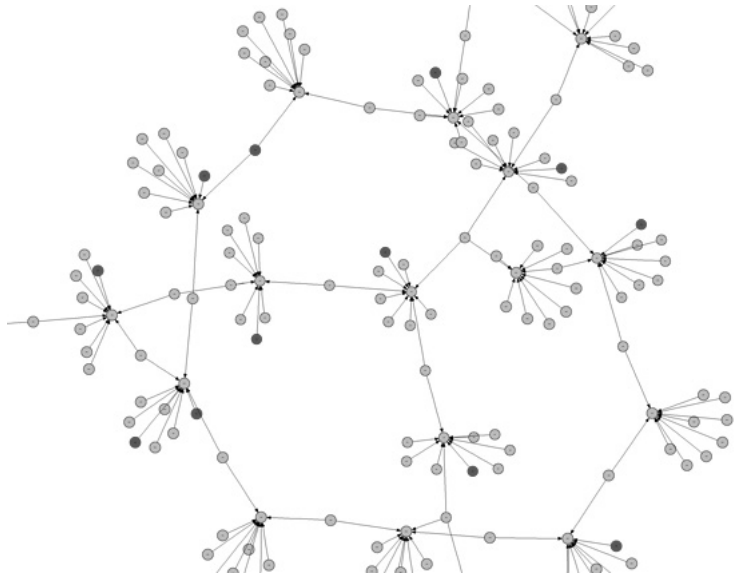


FIGURE 6: POSSIBLE STRUCTURE OF A SURVIVABLE P2P NETWORK



Impact of Structure on Traceback and Response

TABLE 1: COMPARISON BETWEEN ATTACK NETWORK STRUCTURES

When the three structures just described are compared as far as traceback and mitigation are concerned, we can see a progression of hardening and in-

Structure	C2 Links per Agent	C2 Links per Handler	C2 Method	Impact of Crypto on Traceback
Handler/Agent	1 per handler	1 (from attacker)	Direct	Low
IRC	1	1 per IRC server	Indirect	Moderate
P2P	Random	NA	Indirect	High

direction that tends toward more resilience as each new structure is employed. This comparison is illustrated by Table 1.

The advent of peer-to-peer command and control of botnets has many implications for incident response and mitigation. In the early days of DDoS, the handler/agent model of command and control was prevalent. To completely mitigate an entire DDoS network, it was necessary to perform manual traceback from one or more DDoS agents to a handler, where an incident responder would then do one of two things:

- Do further manual traceback through traffic analysis to identify all agents in communication with that handler.
- Do host forensics and retrieve the list of agents from a file within the handler's file system, possibly also having to decrypt that file first.

Once all of the agents were identified, the responder would need to send a series of reports and cleanup requests to the owners (as identified by WHOIS records) of all of these IP addresses.

Those steps alone are very time-consuming and complicated, take relatively advanced skills and understanding of DDoS attack tools, methods, operating systems involved, and host- and network-based analysis, and are complicated by differences in time zone, language, legal structures, available re-

sources, available tools, and available skills. For this reason, DDoS-related incidents can last for many months [15].

In case of IRC-based command and control, one could try to detect activity on the IRC port 6667/tcp, assuming the attackers are using a standard IRC server port. Even if they chose not to do so, one could look with ngrep for IRC commands, such as joining a channel (JOIN). Mitigation would then follow.

The current detection and mitigation approaches, such as DNS-based or text-based signature detection, will be impeded by the features described in this article. There are, however, still actions the user can take, such as using a variety of antivirus programs, rootkit detection programs, or the Microsoft Malicious Software Removal Tool (MSRT). On the network, one would have to look for anomalous behavior on the local networks (e.g., spreading of the malware, spamming, or denial of service). These methods are not foolproof, as the cat-and-mouse game between rootkit authors and defenders continues.

Trends

More recently, botnets have become the principal subject of research and investigation in academic, commercial, and legal circles. The principal means of fighting botnets is to focus on identifying and removing command and control channels on IRC servers [2, 13]. These efforts have proven effective against the typical botnet in use on standard IRC networks, but they are less effective against "rogue" IRC servers (such as those in [4, 15]) that are either established at sites under control of the person(s) using the botnet, are friendly to them, or are on hosts where little assistance is available to take the system off-line.

Command and control traffic that *does not* utilize a single IRC channel and is heavily encrypted significantly increases the difficulty and time it takes for the entire attack network to be identified and mitigated.

Attackers are realizing the value of sophisticated botnets, if it means that their network can evade detection and perform longer, sometimes for many months. The increased use of various topologies for the botnet (as illustrated in Figures 5 and 6), encryption to evade direct detection, and P2P command and control to avoid a central point of command and control that could be taken down are all characteristics of this development. For examples of two P2P bots, we suggest reading the article by Sam Stover et al. beginning on p. 18 in this issue.

Conclusion

We have seen the implications of the use of a P2P command and control mechanism, as opposed to either the classic handler/agent or the more contemporary IRC-based central server mechanism, on detection and reaction aspects of investigating and mitigating malicious botnets.

We have also seen the increase in sophistication and breadth of features in malicious software, so it should be expected that more powerful command and control mechanisms will increase the flexibility and dynamism of distributed intruder tool networks in the future. The motivation for this comes from the economic interest in maintaining and retaining control of these attack networks in the face of response or competitors. Two types appear to be

emerging: one that is easy to use by the attacker, characterized by a simplistic or throwaway network, and another, more sophisticated and economically lucrative, characterized by resiliency and survivability of the botnet.

Coming soon to a networked computer near you. Perhaps it's already there?

ACKNOWLEDGMENTS

The authors would like to thank John Hernandez for his valuable contributions.

REFERENCES

[1] CERT Coordination Center, Results of the Distributed-Systems Intruder Tools Workshop, December 1999: http://www.cert.org/reports/dsit_workshop.pdf.

[2] E. Cooke, F. Jahanian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," *Proceedings of SRUTI '05: Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005: http://www.usenix.org/events/sruti05/tech/cooke.html.

[3] The Honeynet Project, Scan of the Month 25: Slapper Worm .unlock .c source file, 2002: http://www.honeynet.org/scans/scan25/.unlock.nl.c.

[4] The Honeynet Project, "Know Your Enemy: Tracking Botnets," 2005: http://www.honeynet.org/papers/bots/.

[5] The Honeynet Project, "Know Your Enemy: Fast-Flux Service Networks," 2007: http://www.honeynet.org/papers/ff/fast-flux.html.

[6] LURHQ, "Phatbot Trojan Analysis," June 2004: http://www.lurhq.com/phatbot.html.

[7] K. Mardam-Bey, mIRC homepage, 2006: http://www.mirc.com/.

[8] Microsoft Research, "The Windows Malicious Software Removal Tool: Progress Made, Trends Observed," June 2006: http://www.microsoft.com/downloads/details.aspx?FamilyId=47DDCFA9-645D-4495-9EDA-92CDE33E99A9&displaylang=en.

[9] J. Mirković, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms* (New York: Prentice Hall PTR, 2004).

[10] Nullsoft, WASTE, 2003: http://waste.sourceforge.net/.

[11] PlanetPeer.de, PlanetPeer anonymous filesharing wiki: http://www.planetpeer.de/wiki/index.php/Main_Page.

[12] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging," in First Workshop on Hot Topics in Understanding Botnets (HotBots '07), April 2007: http://www.usenix.org/events/hotbots07/tech/full_papers/rajab/rajab_html/.

[13] R. Naraine, "Botnet Hunters Search for 'Command and Control' Servers," eWeek.com, 2005: http://www.eweek.com/article2/0,1759,1829347,00.asp.

[14] SecureWorks, "SpamThru Trojan Analysis," October 2006: http://www.secureworks.com/analysis/spamthru/.

[15] United States Department of Justice, U.S. v. James Jeanson Ancheta: http://news.findlaw.com/hdocs/docs/cyberlaw/usanchetaind.pdf.

[16] Wikipedia, "Eggdrop," 2006: http://en.wikipedia.org/wiki/Eggdrop.

[17] M. Zalewski, "I Don't Think I Really Love You": http://seclists.org/lists/vuln-dev/2000/May/0159.html.

SAM STOVER, DAVE DITTRICH,
JOHN HERNANDEZ, AND SVEN DIETRICH

analysis of the Storm and Nugache trojans: P2P is here

Sam Stover is the Director of Tech Ops for iSIGHT Partners, a startup which produces human and electronic intelligence-fused products and services. Research interests include detection and mitigation methods for vulnerabilities and malware.

sam.stover@gmail.com

Dave Dittrich is an affiliate information security researcher in the University of Washington's Information School. He focuses on advanced malware threats and the ethical and legal framework for responding to computer network attacks.

dittrich@u.washington.edu

John Hernandez, a recent graduate of the University of Washington, is currently working for Casaba Security LLC as a security consultant. Research interests include reverse engineering and malware forensics.

lafkuku@msn.com

Sven Dietrich is an assistant professor in the Computer Science Department at Stevens Institute of Technology in Hoboken, NJ. His research interests are computer security and cryptography.

spock@cs.stevens.edu

SINCE THE ADVENT OF DISTRIBUTED intruder tools in the late 1990s, defenders have striven to identify and take down as much of the attack network as possible, as fast as possible. This has never been an easy task, owing in large part to the wide distribution of attacking agents and command and control (C2) servers, often spread across thousands of individual networks, or Autonomous Systems in routing terms, around the globe. Differentials in the abilities and capabilities of these sites, as well as knowledge of what role the site plays in distributed attack networks (potentially many active at one time), make mitigation harder, as do differences in legal regimes, etc. [1]. Still, there has grown a huge population of researchers, security vendors, and organizations focused on identifying and mitigating distributed attack networks.

In another article in this issue ("Command and Control Structures in Malware from Handler/Agent to P2P"), the authors discuss how topologies for C2 have changed over time, for various reasons. The most popular method of C2 in recent years has been the use of Internet Relay Chat (IRC), either in standard form or through use of customized implementations of IRC servers and clients intended to thwart mitigation efforts. Programs that use IRC for C2 are known as "bots" (short for "robot"), and a distributed network of bots is known as a "botnet." The terms "bot" and "botnet" are now becoming so widely used that they are losing much of their original meaning. Botnets will likely be around for some time, causing a huge amount of grief for network operators, victims of DDoS attacks, and other victims, but IRC-based bots are not the be-all and end-all, and the advent of Peer-to-Peer (P2P) mechanisms for C2 may spell the eventual death of IRC as a means of C2. At that point, when IRC is secondary (at best) and possibly not involved at all, will the terms "bot" and "botnet" still have their original meaning, or will they become general terms that are synonymous with "trojan," "worm," etc.?

There are some obvious advantages to decentralizing the C2 mechanism, and P2P networking fits the bill. The Overnet protocol is particularly attractive because of the transient, "self-healing" nature of the network, along with "servers" that actually share files. Each peer is constantly advertising its

presence, as well as requesting updates from other peers. Throughout this article, there are references to eDonkey and Overnet portions and fields in the packets. This is because the Overnet protocol was based on eDonkey—the primary difference is that pure eDonkey clients are all equal (i.e., there are no "servers"). Overnet expanded on eDonkey to allow peers to communicate information about the P2P network via eDonkey methods, but it added the capability to introduce servers that could host files, if necessary [2]. As we'll see, Storm requires the server file-sharing technique offered by Overnet networks, so technically Storm does not use just the eDonkey protocol, but the Overnet protocol. To reduce confusion, when generically discussing the P2P capabilities, the term "Overnet" will be used. However, when a legacy eDonkey protocol characteristic is being discussed specifically, for example an eDonkey Connect request, the specific name will be used.

Our goal was to compare two different trojans, with a particular emphasis on how they rely on P2P mechanisms. We infected a number of fully patched Windows XP SP2 test systems with samples of Storm and Nugache trojans and analyzed the resulting network traffic captures. Because a large volume of research exists on the impact to the host, we decided not to focus on this (although we did monitor some environmental changes to track the progress of the infecting malware), but more on the communication methods. We hoped to be able to define some mitigation methods at both network and host levels, which will be presented at the end of the article.

To adequately describe the similarities and differences between the P2P capabilities of Storm and Nugache, a brief walkthrough of both is required. The Storm trojan is primarily designed to send spam, but because of its modular nature, it can easily acquire other capabilities such as the DDoS module that was used to great effect early in 2007 [3]. Once installed, the trojan joins and participates in an extensive network that utilizes the Overnet protocol to distribute information and eventually supply the infected peer with the tools it needs. As we will see, the Overnet P2P mechanisms it uses to propagate information provide a very effective means to this end.

Several different Storm binaries were collected and used to infect the test systems, but behavior was consistent throughout the exercise. A complication referred to by one of the authors as the "P2P chicken and egg problem" exists when a new P2P peer is created: It has no knowledge of the current state of the network. A certain amount of information must be delivered with, or in, the binary, which directs the trojan to other infected peers. In truth, only one peer is needed, but the self-healing nature of P2P networks requires a very transient state. If all newly infected systems were given one "superpeer" to connect to, it would only be a matter of time before that IP address was discovered and addressed via firewall rules, IDS signatures, legal action, etc. Storm takes the opposite tack, in that it seeds each new peer with approximately 300 static peers in a text file called spooldr.ini, although this name may change (e.g., it was wincom.ini previously). This file contained approximately 300 rows, with two fields per row. Figure 1 shows the first few lines of the spooldr.ini file delivered with our trojan:

```
[config]
[local]
uport=11873
[peers]
00000000000000009C2DB8A6F34A9C69=452FC581466700
00010CED75C2E4C6222534E6BD5BB4A1=D5868ADE16C900
0001351DE60D58519C2DB8A6F34A9C69=452FC581466700
00037A3051FE23B6BE8B8C79BE6DD56A=41FF4E35835600
```

FIGURE 1: BEGINNING OF CONFIGURATION FILE FOR STORM

The first field in each line contains the peer hash, which uniquely identifies the peer node. The second field contains the IP, port, and peer type, in hex, for the original set of peers with which the trojan will start. These hosts are the only ones currently known to the trojan, and they will be contacted in the hope of receiving up-to-date information about the network. The first packet that our infected system sent out was an eDonkey Publicize packet, shown in Figure 2, which is designed to alert existing peers that a new system is available to the network:

```
06:14:22.949925 IP 192.168.168.152.2506 > 81.248.26.210.20136: UDP,
length 25
    0x0000:  00c0 4f1e 2844 5254 0012 3456 0800 4500  ..O.(DRT..4V..E.
    0x0010:  0035 014c 0000 8011 6361 c0a8 a898 51f8  .5.L....ca....Q.
    0x0020:  1ad2 09ca 4ea8 0021 77bd e30c 89ae f92f  ....N..!w....../
    0x0030:  20bb bac5 dce0 e6ee 6d51 1cda 0000 0000  ........mQ......
    0x0040:  ca09 00
```

FIGURE 2: EDONKEY PUBLICIZE PACKET

Converting the destination IP address (81.248.26.210) into hex yields 0x51F81AD2. Searching for that string in the spooldr.ini file gives:

```
F3032DA7F7C1E94A4FE9D59838C67D40=51F81AD24EA800
                                 ^^^^^^^
```

Logic suggests that the destination port follows the IP, and sure enough 0x4EA8 is 20136, the port that our packet went to. The final two digits are the Overnet Peer Type designation, which we will see later.

Another important aspect of the "chicken and egg" problem is that the new peer is unsure of its external IP address. A breakdown of the Publicize packet will demonstrate this, as well as lay the groundwork for all of the eDonkey fields. The eDonkey portion of the packet starts with 0xe3 (byte offset 0x002A), which designates the eDonkey protocol, followed by the type of eDonkey packet, which in this case is 0x0c for Publicize. The remaining portion of the Publicize data represents characteristics of the Overnet Peer:

Hash Identifier:	0x89ae f92f 20bb bac5 dce0 e6ee 6d51 1cda
IP Address:	0x0000 0000 (0.0.0.0)
Port:	0xca09 (2506)
Peer Type:	0x00 (0)

Since the trojan does not know its external IP address, the value for the IP address field is 0.0.0.0. Our system must rely on a replying peer to provide that information. When a live peer receives a Publicize packet, it responds with a Publicize ACK (0xe30d), a very simple packet with a 2-byte payload that informs the publicizing peer that it is willing to communicate. Upon receiving a Publicize ACK, the new system now has someone to talk to and quickly sends out two very important packets; an eDonkey Connect (0xe30a) and an IP Query (0xe31b). The purpose of the Connect request is to gain current information about the P2P network, while the goal of the IP Query is to determine its own routable IP address. Upon receiving a Connect Reply (0xe30b) and an IP Query Answer (0xe31c), the new peer has conquered the chicken and egg problem: It is now aware of new peers, and it knows its routable IP address.

Until now, the sole purpose of the malware was to find its place in the network by learning about other peers and advertising itself to them. Now that it has integrated into the network it is time to get to work, and for this trojan, that means spam. This is not to say that the malware stops advertising and learning—that process continues throughout the life of the infection. But once a certain steady state in the network is reached, the traffic shifts

from pure Publicize and Connect requests to searches for specific kinds of data, followed by the initiation of TCP sessions. Until this point, all traffic has been UDP, which is consistent with the eDonkey protocol description: "In the Edonkey [*sic*] network the clients are the nodes sharing data. Their files are indexed by the servers. If they want to have a piece of data (a file), they have to connect using TCP to a server or send a short search request via UDP to one or more servers to get the necessary information about other clients sharing that file"[4]. In the case of the Storm network, TCP connections are made to servers that hold the keys to the spam kingdom. There are email lists, mailserver names, and email templates to be had for the asking. Once a peer knows where to look, it initiates a TCP session to that server, or in our case, group of servers, looking for the goods. In one test session, our system repeatedly attempted to establish TCP sessions with ten IP addresses, three of which were successful: one registered in Kiev, one in Romania, and one in Illinois. Of the three sessions, one was very short, only 9 bytes, which implies that the server, although active, did not have the information being searched for. The other two sessions, however, were exactly 52,806 bytes each, and after these sessions were completed, the trojan began to spam.

First, before actually sending any spam, MX record queries were made. Once the response was received from the DNS, numerous short conversations were held with the mailservers (see Fig. 3).

```
06:17:59.971764 IP 209.191.89.172.25 > 192.168.168.152.1092: P 1:136(135) ack 1 win 65535
E....'@./..u..Y.....DS..at..aP.....421 Message from (76.2.252.62) temporarily
deferred
06:18:00.018835 IP 65.54.244.72.25 > 192.168.168.152.1096: P 1:311(310) ack 1 win 65535
E..^..@.o...A6.H.....HcK.....P.....220 bay0-mc5-f1.bay0.hotmail.com Sending
Unsolicited commercial or bulk e-mail to Microsoft's computer network is prohibited.
06:18:00.091036 IP 67.91.84.250.25 > 192.168.168.152.1097: P 1:120(119) ack 1 win 65535
E....F@.m..|C[T.....IK.8....uP.....220 mail1.nuparts.com Microsoft ESMTP MAIL
Service, Version: 6.0.3790.3959 ready at Sun, 23 Sep 2007 03:43:24 -0700
```

FIGURE 3: SNIFFED SPAM COMMUNICATIONS WITH MAILSERVERS

After a number of these mailserver tests, the infected system begins to send spam using standard SMTP.

Nugache

As far as Nugache is concerned, our evaluation is based on examination of several binaries, spread over many months. The functionality of this trojan evolved over time, with its command set increasing in conjunction with its attack and spreading capabilities. Although one of its main purposes is DDoS, it is also capable of acting as a password-protected proxy, propagating itself, downloading and running arbitrary programs, and also collecting and returning keystroke information, possibly compromising user-entered data. In the early stages of development, Nugache, also known misleadingly as the "tcp/8 bot," used fixed ports and IRC for command and control. Later on, it dropped below the radar, as its command and control communications moved to random high-numbered ports and evaded detection in most cases. Keeping track of only a few neighbors, a single trojan would not be aware of the entire network.

Nugache is simpler in architecture than Storm, and in some ways it is simpler in its use of P2P C2 communications. This simplicity comes partly from handling the seeding of peers in a different way from Storm (either by preseeding the compromised host's Windows Registry with a list of peers prior to first running the malware or through bootstrapping the list from a small set of hard-coded default hosts within the binary itself). The default list is in binary form (not ASCII "IP:PORT" strings) and packed into the binary to further conceal them.

Once the list of peers is produced, Nugache peers can join the network using an RSA key exchange to share seed material used to generate and return a Rijndael 256-bit OFB (Output Feedback) mode key. This exchange is shown in Figure 4. Once encryption is set up, an internal protocol is used to perform other tasks. One of these tasks is to negotiate a "connect-back" process to determine the initiating peer's routable IP address and listening port number, at the same time determining whether this peer can act as a "servent" or is only capable of being a client. As with Storm, knowing whether a peer is capable of being a servent is important to determine.

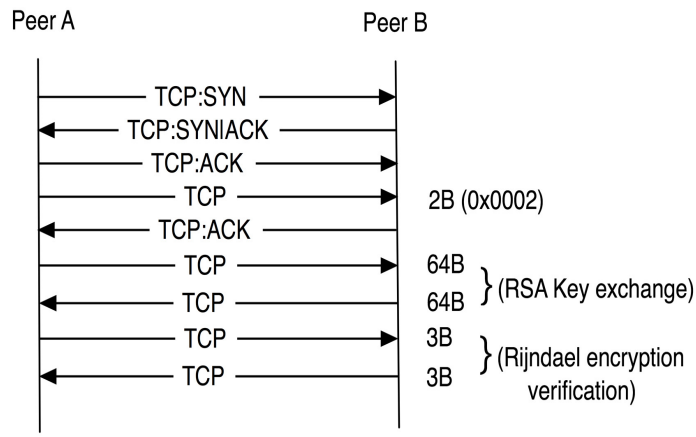


FIGURE 4: SAMPLE KEY EXCHANGE AND ENCRYPTION SETUP

Other tasks involve checking to see if the peer should have its binary instance updated, or whether an updated peer list is needed. Nugache suffers from the same problem as Storm in regard to needing an up-to-date list of peers to use for rejoining the P2P network. After that, a "PING" and "PONG" keep-alive exchange is engaged, and peers inform their connected siblings of any newly discovered servant peers that join the network. This latter information is used to "refill" the seed list in the Windows Registry in the event that excessively nonresponsive peers have been dropped from the list.

Now that we've briefly discussed Nugache and Storm, we can identify the similarities and differences between the two trojans, specifically with respect to how they utilize P2P to function. We've broken the types of comparisons into nine different categories: primary command and control mechanisms, initial peer list seeding, use of cryptography to secure the C2 channel, use of DNS, connectivity, updates, listening port for P2P connections, architecture, and detection.

PRIMARY COMMAND AND CONTROL MECHANISM

Nugache almost entirely handles C2 via the encrypted P2P channel established between connected client and servant peers. There is an IRC capabili-

ty built in, and Nugache peers could respond to (or direct) classic IRC bots in standard IRC channels, but IRC sessions have very rarely been seen in practice.

Storm's C2 is handled via TCP initiated to servers that were not discovered via pure P2P client integration communications. That is, when the peer joins the network, there is a lot of communication with other Overnet peers, but these peers are utilized purely to (a) determine the state of the network and (b) join it. No file exchanges take place via P2P communications; however, IP addresses of "servers" hosting email templates, email lists, and mailservers are disseminated using P2P communications. As such, Storm is more of a "pull" C2 technology—servers do not "push" commands down to the clients; rather, the clients "pull" data from servers.

INITIAL PEER LIST SEEDING

For a Nugache peer to join the network, it must first know of at least one active servant peer that will accept an incoming connection. Once connected, peers will be told of other servant peers and will maintain a list of up to 100 such peers for future use in rejoining the network. How that list of 100 peers is first loaded is controlled by a "bootstrap" process. How it is kept up-to-date is a function of the Nugache P2P algorithm.

The initial seeding is done one of two ways. Either a helper program is run to fill the Windows Registry, which then starts the Nugache program, or the Nugache program uses an internal list to attempt to find an active servant peer, which then provides a current list of up to 100 recently seen servant peers.

Once connected to one or more peers, those peers will report new connecting servant peers. This allows connected peers to learn of recently active servant peers, which have a high probability of being available for reconnecting after a system reboot or shutdown.

Storm seeds its initial peer list via a text file ("spooldr.ini" in this investigation). The infected systems sends eDonkey Publicize packets to all of the peers located in the spooldr.ini file. Once a Publicize ACK is received from a live peer, Connect requests that ask for up-to-date information on peers are delivered. Publicize and Connect packets are consistently sent to any and all peers throughout the life of the infected system.

USE OF CRYPTOGRAPHY TO SECURE THE C2 CHANNEL

Nugache uses a variable bit length RSA key exchange, which is used to seed symmetric Rijndael-256 session keys for each peer connection. Rijndael is also used to encrypt keystroke log files prior to transfer, using a key that is derived from information unique to the peer sending the keylog data. IRC sessions (when used, which is rarely) are not encrypted in any known version of Nugache.

Storm uses a hash mechanism for encrypting data requests to peers and servers. In previous examinations of this trojan, the encrypted string was stored in the meta-tag field of an eDonkey Search Result packet. In the version we investigated, the meta-tag field was either empty or contained a cleartext string (e.g., "20765.mpg;size=78092;"). This indicates that the obfuscation and encryption method is evolving.

USE OF DNS

Aberer and Hauswirth [5] cite the description of P2P as given by Clay Shirky (The Accelerator Group), which states that, "P2P nodes must operate outside the DNS system, and have significant or total autonomy from central servers" [6]. In this respect, Nugache is a true P2P malware artifact. DNS is almost entirely unused, save for immediately prior to DDoS events, joining IRC channels on rare occasions, or for other activity that peers are tasked with via commands, and there is no central C2 server (i.e., peers operate fully autonomously). As there is no use of DNS for seeding peer lists, for identifying C2 channels, or for joining the network, any DNS-based detection or mitigation mechanism will be entirely blind and useless in dealing with Nugache.

Similarly, Storm relies on DNS purely for MX record requests. Mailserver names are passed to the trojan (e.g., "gmail.com"), it performs an MX record query, and upon receiving the answer, it proceeds to connect to port 25 and send spam. Although no DDoS activity was observed, it seems logical to assume that a similar process would be followed: A domain name would be passed to the trojan, DNS queries would be made, and then DDoS activity would commence. At no time was any DNS activity observed that was related to the P2P or to the TCP/C2 communications.

CONNECTIVITY

Nugache peers maintain an in-degree of connections that totals no more than ten clients at any time. The out-degree varies, but it is typically less than half of the ten-client limit. The result is a typical peer with at most about 13–15 connections active at any given time.

Storm seems to collect as many active peers as possible. Because connections to these peers utilize UDP, the malware is chatty, but the traffic is lightweight. Constant Publicize and Connect packets are delivered while the system is active. There were approximately 300 peers in the original spooldr.ini file, but the file would grow as new peers were discovered. In one case, after less than 30 minutes, the spooldr.ini file contained over 430 entries. Theoretically, this number could, if monitored, be used to infer the size of the P2P network.

UPDATES

Nugache uses an internal release number to indicate the current version of the currently running code. When peers connect to the P2P network, they compare version numbers, and a peer with a lower version number will request an update from the peer it just connected with. This allows the entire network to continually upgrade itself as peers come back online after an absence. (Nugache stopped using the fixed port 8/tcp well over a year ago, yet a handful of connection attempts to port 8/tcp can still be observed occurring today.)

Storm updates itself in two main ways. Peer updates and information queries utilize UDP P2P communications, whereas TCP sessions are used to download important data such as new functionality (i.e., the DDoS module), Mailservers, and email templates.

LISTENING PORT NUMBER FOR P2P CONNECTIONS

Although Nugache is known to some analyses as the "tcp port 8 bot," it has not used this fixed port since June 2006. Each peer chooses its own randomly generated high-numbered port to listen on, ranging from 1025 to 65535. On connecting to the Nugache P2P network, the connecting peer will report what port it listens on, and the connected (servant) peer will check to see if a connection can be made back to the connecting peer. If a connection can be made, the routable address of the connecting peer is sent to that peer and the IP:PORT combination is reported to other active peers for future reference. (See the section on "Initial Peer List Seeding.")

Storm picks a random high port for communications and advertises that port in every packet that it sends out. Publicize and Publicize ACK packets are used to establish initial communications as well as to verify the proper IP address and port number. In the case of a peer receiving a Publicize packet that contains the 0.0.0.0 IP, that peer will transmit an Identify Packet requesting the proper IP.

ARCHITECTURE

Nugache is a monolithic binary executable, written in Visual C++ and packed with a simple home-grown packer. Other helper programs have been observed (e.g., used for seeding the initial peer list and installing the Nugache binary on infected systems), but these programs are secondary and not required to infect a host with Nugache. State is maintained primarily through the use of the Windows Registry; however, a keystroke log is kept until retrieved via commands from the attacker controlling the Nugache P2P network.

Storm is packed åusing a more complex method. The first stage of unpacking is decrypted using the XOR function; then a TEA decryption algorithm is applied; finally, the binary is reconstructed using the TIBS unpacker. After the binary is extracted it drops a copy of the original binary plus the spooldr.ini file into the Windows directory and also extracts a rootkit driver called spooldr.sys to the system32 directory. Then it loads this driver into the kernel via an unexpired call from tcpip.sys called SfcFileException, allowing the binary to hide from the OS [7]. After the binary is set, it attempts to communicate to the peers located in the spooldr.ini file and establish itself in the P2P network. Storm is a multicomponent modular set of programs. Each component has a different purpose, and the programs are installed in a set after the initial infection has occurred and the Storm program has successfully found a C2 server.

DETECTION

There is no static IDS signature that will detect Nugache P2P flows. The RSA key exchange is dynamic enough that one cannot get a 100% successful hit rate on detecting the key exchange. (The "Bleeding-Snort" [8] signatures for Nugache that appeared in May 2006 are insufficient to detect all Nugache flows.) More research is necessary to come up with an effective means of IDS detection of Nugache flows.

Nugache can be detected on hosts through various signatures of the infection itself, including Windows Registry keys in HKCU\SOFTWARE\GNU\, a MUTEX lock "d3kb5sujs50lq2mr," the keystroke log file (e.g.,

C:\Documents and Settings\user\Application Data\FNTCACHE.BIN), and one of at least three known names for the binary itself (C:\WINDOWS\system32\mstc.exe, system32\mvwatvx.exe, and system32\wmipvs.exe).

Storm can be detected in several different manners, none of which is foolproof.

On the host, one way (outside of noticing the trojan installing itself) to identify Storm is to find the spooldr.ini file. A host-based IDS could be configured to look for that file and understand its contents. Once the file is found, it can be removed, and the system cannot function. When we cleared that file of peers, the trojan was unable to complete the initial Publicize advertisements and was effectively neutralized. Obviously, this is not a long-term solution, as once it becomes public, the format of the file will change. As with Nugache, signatures could be written for different stages of the initial infection. One example of this kind of detection would be to install a system-monitoring tool such as Capture-BAT [9], which was utilized during this investigation to determine what the malware was doing (e.g., the many "writes" to spooldr.ini were one indication that the file was being updated). Incorporating the Capture-BAT output file into a HIDS strategy would provide a good source of intel for which signatures could be written.

On the network side, it's much more difficult to differentiate Storm P2P traffic from legitimate P2P communications. It would be much easier to detect the voluminous amount of outbound TCP/25 traffic from an infected system; however, this is a very reactive strategy. User education is likely the only mitigation method to prevent installation of the malware.

Conclusion

We have just seen a comparison of two recently successful distributed malware networks that employ P2P concepts, in slightly different roles and degrees, for command and control. We must assume that these are just two of the first successful attempts to move away from the central C2 mechanism of IRC botnets, toward distributed attack networks that are significantly harder to detect, to shut down, or to trace back to the attackers who are controlling them. Many papers and articles have predicted this eventuality, and the task now is to understand how the threat landscape has shifted and to adjust mitigation strategies accordingly. As we have seen in the past, the old tools and tactics do not entirely go away, but are joined by new tools and tactics. Likewise, defenses are not to be thrown out, but they must expand to accommodate the new reality of a multiplicity of attack tools and tactics, as well as the old. The trick is to adjust fast enough to avoid giving the attackers the advantage for long, and that is the challenge that we revel in rising to meet.

ACKNOWLEDGMENTS

The authors would like to thank Christian Seifert for his assistance with Capture-BAT.

REFERENCES

[1] D. Dittrich and K. E. Himma, "Active Response to Computer Intrusions," in *The Handbook of Information Security*, edited by H. Bidgoli (Wiley,

New York, 2005).

[2] http://en.wikipedia.org/wiki/EDonkey2000.

[3] http://www.secureworks.com/research/threats/storm-worm/?threat=storm-worm.

[4] "The eDonkey 2000 Protocol": ftp://ftp.kom.e-technik.tu-darmstadt.de/pub/papers/HB02-1-paper.pdf.

[5] "An Overview on Peer-to-Peer Information Systems": http://www.p-grid.org/publications/papers/WDAS2002.pdf.

[6] http://www.scripting.com/davenet/2000/11/15/clayShirkyOnP2p.html.

[7] http://www.reconstructer.org/papers/Peacomm.C - Cracking the nutshell.zip.

[8] http://www.bleedingsnort.com/.

[9] http://www.nz-honeynet.org/capture-standalone.html.

TAL GARFINKEL AND
ANDREW WARFIELD

what virtualization can do for security

Tal Garfinkel is part of the advanced development group at VMware and is currently on leave from Stanford University, where he plans to submit his Ph.D. thesis any day now.

talg@vmware.com

Andrew Warfield has completed his Ph.D. at the University of Cambridge and now divides his time between XenSource and an Adjunct Professor position at the University of British Columbia. He lives in Vancouver.

andrew.warfield@xensource.com

VIRTUAL MACHINE (VM) TECHNOLOGY is rapidly gaining acceptance as a fundamental building block in enterprise data centers. It is most known for improving efficiency and ease of management. However, it also provides a compelling approach to enhancing system security, offering new ways to rearchitect today's systems and opening the door for a wide range of future security technologies.

Virtualization emerged as a technique for managing mainframes in the 1960s. In the late 1990s, it was rediscovered in the midst of a perfect storm. Widespread adoption of IT infrastructure based on inexpensive commodity PCs led to explosive growth in the number of machines in enterprise environments. Concurrently, intense growth in the commodity software industry left a legacy of large, feature-rich, and complex applications and operating systems—so complex, in fact, that the best practice for managing and securing them was (and still is) commonly held to be isolating each server application to its own, incredibly underutilized host.

Remedying this inefficiency through server consolidation is perhaps the most well known use of virtualization. More recently the benefits of virtualization for management, such as simplifying provisioning and allowing hardware upgrades without incurring downtime, are becoming equally well known. Another property of this technology that has received less attention is the benefits it can provide for security.

Our objective is to provide readers with a better sense of what virtualization can contribute in this area. We begin by looking at the basic security benefits VMs can provide today (e.g., its power as a mechanism for isolation). We then survey some of the emerging security technologies supported by virtualization that we may see in the years ahead.

Virtual Machines for Isolation

The oldest and simplest path to enhancing security with VMs is by separating multiple applications on a single OS into multiple VMs, with each application in its own VM. As with application sandboxes, jails, etc., this contains the damage an exploited application can inflict on other services. This works equally well in situations with stand-alone applications (e.g., protecting a DNS or email server from a compromised Web server) and for services

consisting of a more complicated aggregate of middleware, such as an e-commerce application with a Web server front end, back-end logic for dealing with user transactions, and a database—each of which could be run in its own VM.

This isolation presents two major benefits that we discuss in detail throughout this article. First, by virtue of running in separate virtual machines, applications are obviously much less vulnerable to compromises that start in other applications; that is, the result of a system compromise is better *confined* in a correctly configured virtualized environment. Second, many of the security mechanisms that we have today are best applied at a host granularity. The *encapsulation* afforded by more single-purpose VM-based environments allows much stricter security policies to be applied at a (virtual) host granularity.

MAKING MACHINE-LEVEL ISOLATION UBIQUITOUS

Isolating applications on their own machine to contain a compromise (e.g., in a company's DMZ) has long been considered best practice. By reducing the physical and management costs of whole system isolation, this practice can be applied far more ubiquitously. Of course, replacing physical isolation with virtualization does come at some cost in assurance, and there are some situations (e.g., separating machines in the DMZ from those inside the firewall) where the additional security this affords makes sense. However, in the common case, the benefits of virtual-machine-monitor-based (VMM-based) isolation outweigh the resulting loss in assurance. A hybrid approach seems most sensible—for example, relying on a VMM to compartmentalize applications on either side of the firewall, while putting a physical gap between these two sides.

Running several virtual machines on a single platform, each hosting a single application, incurs only nominal overhead over running these applications all on the same OS. Depending on the applications and platform configuration, this overhead can potentially even be less, as on multicore platforms, and virtualization can sometimes make it easier to reduce resource contention and expose latent concurrency.

Similarly, application setup times on a virtualized platform are often less than on a traditional platform. Uniform virtual hardware means only a single "base" operating system image is required, which can then be specialized on a per-application basis. Increasingly, administrators will simply have totally configured application VMs on hand in "template" form, where a new VM (e.g., a mail server) can simply be instantiated and deployed as needed. Also gaining popularity is the use of a prebuilt virtual appliance (i.e., an application that has been bundled along with an operating system in a VM by the software vendor).

WHY WHOLE MACHINE ISOLATION?

Using virtual machines for isolation frequently elicits the question, "Why use something so coarse-grained?" BSD jails, application sandboxes such as Systrace or AppArmor, and fine-grained OS-level access controls such as SELinux have long been available. The simple answer is that VMs offer potential benefits over these solutions by way of simplicity and assurance, in terms both of implementation and of configuration.

Correctly utilizing OS-level isolation often requires a deep understanding of both OS semantics and the behavior of the particular application(s) being

secured (e.g., file system usage), to securely configure a system and to debug reliability problems associated with configuration errors. In contrast, the basic abstraction that VMs provide is familiar and easy to understand—at most, an administrator must configure a firewall for containment or set up a network file system for sharing. Beyond these tasks, no OS-specific knowledge is required. Further, securing an OS against exploits is a demonstrably difficult problem, owing to their many functions and broad attack surfaces. The narrow interface that a virtual machine presents offers a comparatively harder target for attackers, thus potentially providing higher assurance isolation.

Of course, virtualization is not a substitute for OS-level mechanisms. If an application is compromised, using a defense-in-depth strategy to contain that compromise only improves protection. Thus, if an attacker must break out of a jail, sandbox, or other OS-based protection mechanisms before attempting to overcome the isolation imposed by the VM, all the better. Further, a VMM-based approach is not always suitable: VMMs excel at isolation, but in situations where there is a great deal of controlled sharing, an OS-level solution may be more suitable.

SYSTEM SPECIALIZATION

Beyond reducing the possibility that a compromise in one application can spread to another, putting each application in its own virtual machine conveys a variety of benefits. First, it eliminates complexity inside the guest, making access controls and other hardening measures easier to set up and allowing a thinner OS to be used, reducing the size of the application's trusted computing base. Next, each VM's attack surface can be reduced, as each VM only requires the interfaces (network ports, RPC daemons, etc.) a single application needs to be enabled.

This approach converges on a virtual appliance model. Just as on a hardware appliance, in a virtual appliance the operating system is specifically tailored from the ground up for the application that it's running. An example of the extreme end of this spectrum is BEA Systems Liquid VM [1], a custom operating system tailored specifically to run its JRocket JVM and middleware platform. Other vendors offer a middle ground, providing custom versions of existing operating systems specifically tailored to the needs of appliances. Both approaches offer the possibility of a smaller trusted computing base and reduced attack surface for applications. Additionally, this allows securing applications to shift from system administrators to ISVs, who can often use their greater knowledge of an application to employ more complex hardening measures.

DATA CENTER PARTITIONING

Technologies for segmenting data centers, such as VLANS, have become increasingly compelling, as they provide an intuitive model of separation. Using virtual machines, this model of separation can be pushed onto the end host, making network-based partitioning an end-to-end proposition. For example, prior to its adoption of virtualization, the NSA used to use physically separate machines to access networks with data at different levels of classification. This provides excellent isolation, but of course an architecture like this is expensive and unwieldy. This prompted the NSA's move to their Nettop architecture [5], which instead used virtual machines for isolation on the same physical host.

With the cost of such an end-to-end solution reduced, such an architecture becomes feasible for normal businesses. One common pattern is to partition a user's desktop into two parts: One partition has access to all company internal resources, while the other can communicate with the outside world using Web browsers, IM, etc., with all the ensuing perils this entails. This pattern can also be applied inside a company, as with the NSA example. Organizational units such as marketing, engineering, and sales may be configured to have strongly isolated virtual resources, compartmentalizing these organizational roles in the face of compromise.

Virtualization facilitates another interesting sort of partitioning. As backup, logging and monitoring, remote display, and other functionality migrate out of the application VM and into separate protection domains on the virtualization layer, a natural separation occurs between the management plane (with all the infrastructure services that are important for security and management) and the application plane (with services actually used by end users). This again naturally supports an end-to-end separation of duties, with management functionality living on a separate set of virtual machines, separate network, etc., from running services—again, making it easier to gain confidence in the integrity of this part of a data center, even in the face of application compromise.

Virtual Machines for Fine-Grained Protection

Another unfortunate consequence of the rapid growth in size and complexity in commodity operating systems has been a predictable reduction in our ability to have faith that these systems cannot be compromised.

Virtualization can help us address this problem by adding additional protection for code running inside virtual machines. Virtualization may be used both to provide a higher-assurance protection layer than is afforded by the guest OS and to offer an additional degree of defense in depth, for example, preventing data from being leaked from a host, even if the guest OS is compromised.

QUIS CUSTODIET IPSOS CUSTODES?

The question, "Quis custodiet ipsos custodes?" ("Who will guard the guards?")is an important one for modern commodity operating systems. Today's antivirus and host-based intrusion detection and prevention systems are stuck with a difficult chicken-and-egg problem. Much of what they are seeking to detect is OS compromise, especially in light of the growing popularity of kernel rootkits. However, once the kernel has been compromised, the detection tools themselves are left unprotected. Thus, the arms race between attackers and defenders rapidly devolves into a complex game of core wars. This state of affairs has been institutionalized with the introduction of PatchGuard in Microsoft Windows Vista, and, unsurprisingly, the arms race to disable this mechanism is already in full swing [6].

Virtualization provides a way out of this situation by allowing the "guards" to be run in an entirely different protection domain, *outside* the influence of the OS that they are protecting. This can be accomplished in a number of ways. For example, a kernel rootkit detector could be protected by the VMM in situ (i.e., in the same address space as the operating system it is protecting) by preventing modifications to detector code and ensuring that it is executed with a specified frequency. Alternatively, the detector could be moved outside of the guest OS entirely and run in a totally separate VM [2]. Both approaches offer a simple and clear advantage over today's systems, where

intrusion detection and prevention systems must paradoxically rely for protection on the OS they are trying to protect.

GETTING BETTER HIPS

VMs can enhance monitoring and enforcement capabilities in AV and HIPS by allowing efficient interposition on hardware events in the guest OS. x86 virtualization has long played technically exciting tricks involving the virtualization of the hardware memory management unit (MMU); MMU virtualization is required to prevent VMs from accessing each other's memory, and providing high-performance implementations is one of the major challenges of effective x86 virtualization. Leveraging this capability for security, a system can enforce exactly what code should be permitted to execute on a host [4] or, alternately, perform up-to-the-moment detection of malware, offering superior detection capabilities when compared with the file-scanning approaches of today's AV systems. Interposing on devices offers many other possibilities: for example, preventing malware from being written to disk, scanning a USB disk when it is plugged into the machine, or filtering network traffic.

LOCKING DOWN DATA

Because it acts as a small privileged kernel that runs *under* an existing guest OS, the virtual machine monitor can impose nearly arbitrary protection policies inside the guest. Potential applications range from protecting key data in an SSL stack, to preventing sensitive documents from being leaked from a VM, to enforcing fine-grained information flow policies. Although few technologies offering this capability have been deployed to date, the possibilities are rich and promising.

Logging and Snapshots: More of What You Want and Less of What You Don't

With the shift from physical to virtual machines, what was once hardware state, locked away on devices, becomes entirely software-based. This change allows us to rethink how we manage and take advantage of system state.

DOWN WITH HARD STATE

One of the biggest challenges in managing security within an installed base of software is keeping it secure as it ages. Users and administrators generally accept that a freshly installed application running on a freshly installed OS usually works just fine. However, as other software is installed and uninstalled, applications are run, and time passes, confidence in the safe and correct configuration of a system diminishes. Moreover, once a system has been compromised—or in many cases even *may have been* compromised—the only way to restore confidence is to return to a "clean" state, which is often achieved by completely reinstalling the OS, or at least reimaging the system.

VMs provide a very compelling way to deal with this sort of complexity. They can trivially checkpoint and return to points in history, allowing the unknown permutations to a system to be dropped. Further, we may specify a positive filter on what changes we *do* want to keep, perhaps preserving the contents of a bookmarks file while reverting a Web browser VM to a freshly installed state with each restart. In this sense, VMs allow us to treat millions

of lines of complex software as a very coarse-grained transformation on a small and confined set of data. Any changes made by the VM to internal state other than the interested set can simply be dropped and reverted to a fresh install, providing much stronger levels of confidence in the system.

FORENSICS AND REMEDIATION IN THE LOG-STRUCTURED DATA CENTER

Carrying this notion of logging and reverting the state of a system one step further allows us to consider simply recording to a log everything any VM in a data center ever does. Considerable research work on virtualization has explored the notion of logging and replaying a VM's execution at a fine granularity, using techniques as fine-grained as cycle-accurate instruction replay or as loose as logging network traffic and periodic VM checkpoints. Regardless of the technique used, it's quite reasonable to imagine production systems that include a facility to return to arbitrary points in the history of their execution.

This detailed logging has the potential to solve the very challenging task of separating good from bad in an exploited system. In normal situations, once a system is found to have been compromised the best that an administrator can possibly do is to revert to a backup and attempt to comb through the file system, searching for changes and determining which should be preserved. More often, this represents an unrealistic effort and the post-backup changes are simply lost.

Having a detailed log of a system's execution allows a compromise to be analyzed retrospectively. Detailed analysis tools may be built and run against the system's execution log and attempts can be made to isolate malicious changes, improving our ability to recover data. As a gedanken experiment for what this means in restoring compromised systems, imagine the ability to rewind the execution of a system to just before the point that it was attacked and there insert a firewall rule that refuses to admit the exploit traffic. The system would then play forward without maliciousness, ideally preserving a considerable amount of "good" activity.

Logging and analysis similarly provides the ability to more speedily understand malicious software and attacks on systems, because it allows forensic analysis to be run both forward and backward in time to diagnose the root of a system compromise and determine what malicious activities took place.

The Managed Desktop

The desktop is an inevitable final frontier for virtualization. Its support for a wide range of hardware and the latency-sensitive nature of its applications have posed a higher technical barrier for desktop entry than for the server space. However, from a security perspective the rewards are high: Desktop systems are exactly where many of the advantages that we have described here are most desirable.

In addition to richer policies, virtualization of the desktop allows security policies to be applied uniformly across the enterprise. Firewall policies, network quarantine, monitoring, and the like can be applied whether the user is at his or her desk or in the data center, regardless of the integrity of the guest OS.

Virtualization also provides the ability to strictly limit the actual hardware to which applications contained in VMs have access. For example, the discreet use of a USB memory stick to transfer applications or data off of a machine

may be disallowed, a VM's access to the network may be very tightly controlled, and VM data stored on disk can be automatically encrypted.

A Brave New World

We believe the benefits of virtualization for efficiency, platform flexibility, and ease of management alone will make it ubiquitous in enterprise data centers. As with provisioning and management before it, the benefits of virtualized platforms for improving security will take time to be realized. Part of the current unrealized potential is a lack of deep operational experience in modern IT environments, widespread understanding of how this technology can be leveraged, and tools that help facilitate best practices.

Our enthusiastic endorsement of virtualization's potential should be tempered with the observation that the flexibility and power that make it such a boon for system management also give rise to a variety of new security challenges [3]. Coping with these will require a varied combination of new technologies and best practices. As always, the responsibility for ensuring the secure adoption of virtualized platforms will lay in the hands of both platform vendors and those deploying them.

REFERENCES

[1] BEA Systems, Inc., "Stop Worrying About Computing Capacity—and Start Making the Most of It," 2007: http://www.bea.com/content/news_events/white_papers/BEA_Virtualization_wp.pdf.

[2] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," *Proceedings of the Network and Distributed Systems Security Symposium,* February 2003.

[3] T. Garfinkel and M. Rosenblum, "When Virtual Is Harder Than Real: Security Challenges in Virtual Machine Based Computing Environments," *Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS X)*, May 2005.

[4] L. Litty and D. Lie, "Manitou: A Layer-Below Approach to Fighting Malware," *Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability (ASID)*, October 2006.

[5] R. Meushaw and D. Simard, "NetTop: Commercial Technology in High Assurance Applications," 2000: http://www.vmware.com/pdf/TechTrendNotes.pdf.

[6] Uninformed, "PatchGuard Reloaded. A Brief Analysis of PatchGuard Version 3," September 2007: http://uninformed.org/index.cgi?v=8&a=5.

GERNOT HEISER

Your system is secure? Prove it!

Gernot Heiser is professor of operating systems at the University of New South Wales (UNSW) in Sydney and leads the embedded operating-systems group at NICTA, Australia's Centre of Excellence for research in information and communication technology. In 2006 he co-founded the startup company Open Kernel Labs (OK), which is developing and marketing operating-system and virtualization technology for embedded systems, based on the L4 microkernel.

gernot@nicta.com.au

NICTA is funded by the Australian government's Backing Australia's Ability initiative, in part through the Australian Research Council.

COMPUTER SECURITY IS AN OLD problem which has lost none of its relevance—as is evidenced by the annual Security issue of *;login:*. The systems research community has increased its attention to security issues in recent years, as can be seen by an increasing number of security-related papers published in the mainstream systems conferences SOSP, OSDI, and USENIX. However, the focus is primarily on desktop and server systems.

I argued two years ago in this place that security of embedded systems, whether mobile phones, smart cards, or automobiles, is a looming problem of even bigger proportions, yet there does not seem to be a great sense of urgency about it. Although there are embedded operating-system (OS) vendors working on certifying their offerings to some of the highest security standards, those systems do not seem to be aimed at, or even suitable for, mobile wireless devices.

Establishing OS Security

The accepted way to establish system security is through a process called *assurance*. Assurance examines specification, design, implementation, operation, and maintenance of a system.

The most widely used assurance process is the international standard called the *Common Criteria for IT Security Evaluation*, or Common Criteria (CC) for short. CC evaluation is performed against a *protection profile* (PP), which represents a standardized set of security properties the system under evaluation is expected to meet. The idea is that purchasers of IT systems can define their security requirements through a PP (or a combination of PPs) and can then select any system that is certified to match that PP.

CC compliance is evaluated to a particular *evaluation assurance level* (EAL). These range from EAL1, the easiest (requiring little more than a demonstration that the system has undergone some testing), to EAL7, the toughest. The goal of a CC evaluation is to obtain certification from an accredited authority that the system satisfies all the required criteria for a particular PP at a certain EAL. A higher evaluation level means a more thorough examination of the system. This does not, however, guarantee more security; it means only that a more thorough and systematic attempt is made to eliminate vulnerabilities.

A number of operating systems have been certified under CC, including Mac OS to EAL3, versions of Windows, Linux, and Solaris to EAL4, and the hypervisor of IBM's z-Series to EAL5. The Green Hills Integrity microkernel is said to be undergoing evaluation to EAL6.

But what does this mean? At the toughest assurance level, EAL7 (which to my knowledge has not yet been achieved by any OS that provides memory protection), CC evaluation is characterized as "formally verified design and tested." In a nutshell, this means two things:

- The system has an unambiguous specification. At EAL7 this must be in the form of a formal (mathematical) model, and there has to be a formal proof that the specification satisfies the requirements of the PP (e.g., that no unauthorized flow of data is possible in the system).
- There is a correspondence between the mathematical model and the actual implementation of the system. This is established by a combination of means, including a formal high-level design, an at least semiformal low-level design, formal or semiformal correspondence between them, a detailed mapping of design to implementation, and *comprehensive independent testing*.

There is also a requirement that the system under evaluation be "simple." This is a reflection of the security principle of least authority (POLA) and economy of mechanisms, which imply that a system's *trusted computing base* (TCB) should be as small and simple as possible.

Testing Required

CC, even at EAL7, relies on *testing*. Although mathematical proofs are required for security properties of the system's API, there is no proof that these properties hold for the actual implementation. This is why testing is still required. Testing, as Dijkstra famously stated, "can only show the presence, not the absence, of bugs." Hence, even a system certified at EAL7 must be suspected to contain security flaws.

Why does CC not go further and require an actual correctness proof of the implementation? After all, formal proofs for computer programs have been around for decades. Presumably the answer is that it was not considered feasible. Formal code proofs, doable for small algorithms, scale very poorly with code size. Systems that are undergoing CC certification at EAL6 or EAL7 are typically *separation kernels*, very simple OS kernels whose sole purpose is to provide strict (static) partitioning of resources among subsystems. A typical separation kernel consists of maybe 4,000 lines of code (LOC), which may be small as kernels go but is huge as far as formal verification is concerned.

The Next Step

So, are we stuck with trusting the security of our computer systems to traditional debugging approaches such as testing and code inspection, enhanced by model checking (a class of formal methods that may be able to prove the absence of certain categories of bugs but not *all* bugs)?

I think not. One of the most exciting developments in this respect is that it now seems feasible to fully verify the implementation of a complete *microkernel*. A microkernel is a much more powerful construct than a separation kernel, as it is a platform on which a general-purpose OS can be implemented. A well-designed microkernel is a superset of a separation kernel, in that

it can provide the same functionality, plus more. However, it is inherently more complex: A minimal microkernel that has sufficient functionality to support high-performance systems of (virtually) arbitrary functionality weighs in at some 7,000–10,000 LOC.

In spite of this, complete formal verification of a microkernel is nearing completion at NICTA. In a project that has been running since January 2004, the API of seL4, the latest member of the L4 microkernel family, has been formalized as a mathematical model in a theorem prover. A number of security properties have been proved about this API, with more to come: The aim is to provide a complete set of proofs corresponding to at least one of the CC PPs. The seL4 kernel can then be used as the basis of systems whose TCB is truly *trustworthy*.

The implementation proof is progressing concurrently with the security proofs of the API. It uses the *refinement* approach, which is a multistep procedure involving intermediate representations (between the specification and the code). Each refinement step proves that the lower-level representation has all the relevant properties of the higher level.

In the case of seL4, there are three levels: The formal specification is the highest, and the actual C and assembler code of the kernel implementation is the lowest. The intermediate level (which roughly corresponds to CC's low-level design) has a concrete meaning, too: It corresponds to a prototype of the kernel implemented in the functional programming language Haskell, which serves as an executable specification for porting and evaluation purposes.

The first refinement step is completed; the second (and final) one is in progress and is due for completion during the second quarter of 2008.

This still leaves a gap: It assumes that the correctness of the implementation is established by showing the correctness of the code (C and assembler). Although CC makes the same assumption, this nevertheless leaves the C compiler and the assembler as trusted components in the loop. Given the quality, size, and complexity of a typical C compiler, this is still an uncomfortable level of trust.

The problem could be solved by performing a third refinement step, from C/assembler to actual machine code. This would require a considerable effort, but it is inherently no more difficult (and most likely easier) than the previous refinement steps. However, there is promising work performed elsewhere on compiler verification. A verified compiler could be leveraged to close the gap without a further refinement step on the kernel.

Let's Get Serious About Security!

Security has far too long been treated with insufficient rigor, given what's at stake. CC, despite best intentions, could actually be counterproductive there. By stopping short of the requirement for formal verification at the highest assurance level, CC has the potential to create a false sense of security. After all, a system certified to EAL7 can rightly be claimed to have passed the highest hurdle of security evaluation. The problem is that this is still incomplete, and a potential for security flaws remains.

If complete formal verification is possible, it must become a requirement.

FURTHER READING

The approach taken in designing and implementing seL4 is described by K. Elphinstone et al., "Kernel Development for High Assurance," *Proceedings of the 11th Workshop on Hot Topics in Operating Systems*, San Diego, May 2007, USENIX.

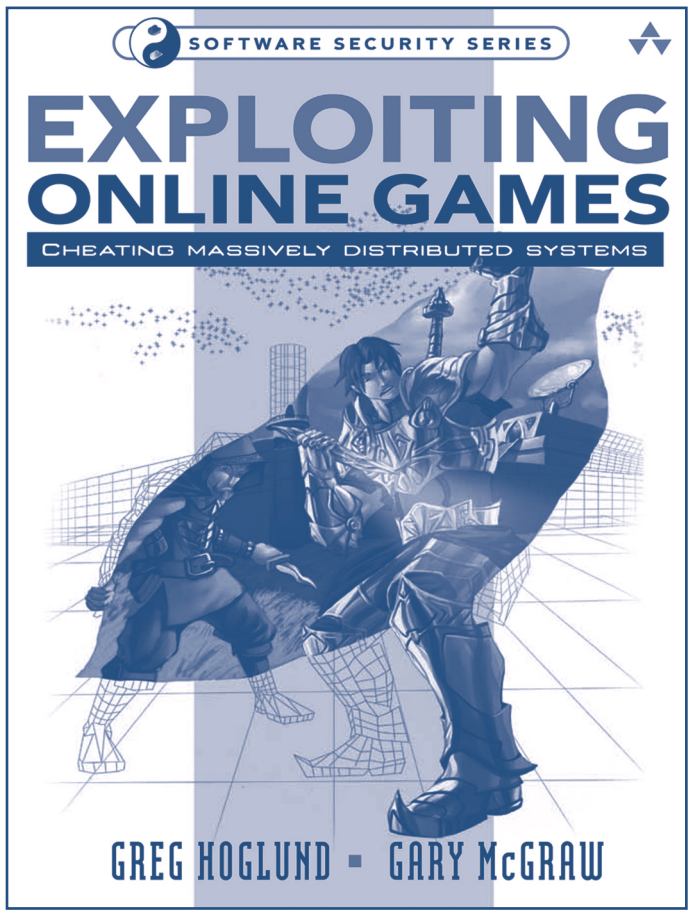
Further information on seL4 can be found on the project Web site, http://ertos.org/research/sel4/, and the Web site of the verification project, http://ertos.org/research/l4.verified/.

The Common Criteria document is available at http://csrc.nist.gov/.

GARY MCGRAW WITH RIK FARROW

exploiting online games: an interview

Gary McGraw is the co-author of *Exploiting Online Games* and the author of *Software Security*. He is also the CTO of Cigital, Inc., a software security and quality consulting firm that has provided services to some of the world's best-known companies for a decade.



I HAD LUNCH WITH GARY DURING the 2007 USENIX Annual Technical Conference in Santa Clara, CA. We talked about his new book, *Exploiting Online Games* [1], written with Greg Hoglund, of Windows Rootkit fame [2]. Gary presented invited talks about the ideas in his book during USENIX Annual Tech and, more recently, USENIX Security. The summary of his USENIX Security IT appears in this issue, along with a summary of a talk by Greg about advanced topics in writing Windows rootkits.

I interviewed Gary via email as he traveled around the country promoting his book, with the goal of going beyond what I had already heard from him.

RIK: You mention the ability to teleport in both your book and your talk. If I understand you correctly, this simply involves poking in new values for your location, right?

GARY: The technique you're referring to is called "telehacking" by most game hackers. Note that some online games explicitly allow teleporting as one normal way to get around, but most don't. In our book, we use World of Warcraft (a game distributed by Blizzard Entertainment) as one of our key examples. We do this because WoW, as it is known to gamers, is the most popular massive multiplayer online role playing game (MMORPG) in the world with over 9 million users, some 400,000 of whom are usually playing together at any given time. In WoW, teleporting is not a normal movement option.

In this case, the mistake the game designers made was to allow the "state" representing character position to be controlled by the game client running on the gamer's PC. This is a fine design decision as long as gamers play by the rules. But if gamers want to cheat, all they need to do to telehack is change the memory values on their PC corresponding to position. You can do that by poking memory values. We have code for WoW telehacking in the book in Chapter 7: Building a Bot.

The real problem here is that the game designers made a critical mistake thinking about trust boundaries and where trust should and should not reside. Clearly any state that is allowed to be completely controlled by the game client on the potential attacker's PC should be considered with some skepticism when it arrives at the game server! If you trust everything the game client says and yet the game client is under the complete control of a cheater, big

problems ensue. This is the kind of error Greg and I expect to see more of as massively distributed software systems become more common.

I wrote an article about this trust boundary issue for IEEE's *Security & Privacy* magazine [3].

Of course, in the book we also discuss why people would want to cheat. Briefly, the why has to do with the value of virtual stuff, which can be sold on a booming middle market. Cheating pays off financially, and it is not at all clear that it is against the law.

RIK: Will monitor programs, like Blizzard's Warden, detect if you have suddenly changed your location (set of coordinates)? What about other changes, for example, changing your level, condition or health, or the amount of gold you have?

GARY: The Warden, which is a monitoring system used exclusively by Blizzard, watches your PC as you play a game. You give Blizzard permission to do this monitoring on your PC when you agree to the end user license agreement (EULA). Note, however, that this granting by the user of permission to install spyware xdoes not make it legal! My suspicion is that Blizzard's use of the Warden is not likely to be legal in California, where there are strict antispyware laws on the books.

The Warden as it is currently set up only monitors things outside the game client process itself. That is, it is concerned with things such as Window titles (of non-Blizzard Windows), other processes running on your PC, URLs in your browser, and Instant Messaging buddy lists. It also checks for DLL-replacement hacks, which we describe in Chapter 6: Hacking Game Clients. My co-author Greg was worried enough about what the Warden does that he developed a program called the Governor that keeps tabs on the Warden. Much of the code for the Governor is included in Chapter 2: Game Hacking 101.

So far, the Warden does not appear to keep track of state changes in the game client such as a change of position. It would be pretty silly to have the Warden do that if you think about it, because a cheater sophisticated enough to poke new values into memory can change the Warden's brain just as easily as the game client itself! (The Warden is simply another user mode process.) The real answer is not to monitor what the client is doing on the client, but, rather, to keep better track of things on the server so that when something crazy like a bazillion-mile hop occurs, a red flag is flown to alert the game minders.

I think one idea that might be useful is a "low-res" vector computed over all of the state that is controlled by the game client. This vector would be computed and stored on the server side at the time that the state crossed the trust boundary. Any such computation would need to be quick and dirty. Then when client state changed or maybe after so much time has elapsed, a new low-res vector could be very quickly computed and compared with the stored value. Too big a change in the vector might signal trouble and could lead to more scrutiny for the player.

The values that you described in the second part of your question would be ideal candidates for inclusion in the low-res vector. In fact, important inventory items are already tracked closely on the WoW server.

There is no perfect answer to the trust boundary/time and state issue, but there are a number of basic things that game companies could do that they are not doing now.

RIK: Earlier on, you mention a low-res vector as a reasonable solution, and that any checking of that low-res vector needs to be quick and dirty. You ap-

pear to be hinting at the enormous computational load that would be required to actually check that a user's state does not indicate that that user is cheating. But shouldn't game vendors, or SOA vendors, be expected to maintain the integrity of their offerings, instead of presuming that there isn't a problem? Perhaps vendors should provision their computing base to handle checking state.

GARY: Managing security risk always comes down to making tradeoffs. In this case, the hard question to ask is how much state checking is enough to thwart cheating? So far the market is answering that it is OK to have none, but I expect that answer to become less acceptable over time. The stakes in online gaming have changed, and the security tradeoffs must be reconsidered in this new light.

RIK: You mentioned manipulating the video card, so that when players using this hack play games, they can see through walls or even the earth in the game, or have their opponents show up in glowing orange. How is it that these hacks could avoid detection by gaming companies' countermeasures?

GARY: This is a kind of hack used against first-person shooter games such as Counter-Strike. The resulting approach is called an "aimbot." We describe aimbots in Chapter 2: Game Hacking 101. The attacker's insight is that the video card knows plenty more about the state of the game than the player actually sees in a traditional view on the screen. After all, the card has to render all of the graphics in a reasonable amount of time. It does this by precomputing and caching lots of things the gamer should not know. It may know, for example, the exact coordinates of an enemy player.

Early aimbots had no countermeasures. Current countermeasures are mostly statistical in nature. If a player is too good to be true, that player comes under scrutiny. Sometimes really good players are banned simply because they are "too good to be true."

RIK: Aren't monitor programs akin to rootkits? I know that EULAs that players agree to allow the gaming company to monitor their systems and even kill programs. But isn't this crossing the line, similar to what Sony [BMG] did with their CD rootkit for copy protection?

GARY: Monitor programs are not really akin to rootkits technically, but they are conceptually. Most monitoring programs that we're aware of run in user mode. Rootkits run in the kernel. A monitoring system in the kernel would be more effective since it could use stealthy rootkit technology to cloak itself. On the other hand, most gamers who play WoW are completely unaware of the Warden or the fact that they agreed to be monitored.

Make no mistake about it; the Warden does in fact work. I've been told by people who experienced it how very quickly the Warden detects the use of Bubba's Warcraft Hack and bans a player.

In my opinion, however, relying on monitoring software that users are not aware of does cross the line as a security mechanism. This won't stop such systems from being used, though. Think about some of the data security technologies currently on the market. They work by installing kernel-level monitors and basically spying on the user. Corporations are paying good money for these things.

I would prefer that game companies keep a handle on state at the server and not spy on the gamers' other processes running on their own PCs.

RIK: What is the coolest game hack you've come up with yet?

GARY: Greg has come up with some doozies, a few of which I covered in the talk. One involves the use of a kernel-level process to manipulate the game

program undetectably by living under it. Another involves freezing the computation, doing some calculations to help you cheat, and then injecting state before unfreezing the computation. The freezing can be accomplished with hardware breakpoints.

Basically what we have is a classic computer security arms race. In this case, because the cheaters have complete control over their PCs, they have a pretty large advantage given current game design. Better security design could help even the scales.

RIK: Did you ever try game hacks that got your account closed or banned?

GARY: When we were writing this book we wanted to start with basics and go from there. To do that, we wrote some pretty easily detectable code. During testing of the Hoglund_WoW_macro code in Chapter 2: Game Hacking 101, the character Xanier was detected and banned. Greg reports that he has had over 20 characters banned for various reasons, including characters that he bought using his wife's name.

I have never played WoW myself, so I have never clicked on the EULA or agreed to its terms, nor have I had characters banned from the game.

RIK: Do you see much of a difference between fat game clients and Web 2.0, where much of an application resides on the client side?

GARY: This is a critical point of the work that I alluded to before. I think the world is currently evolving toward software systems with lots of distributed fat clients, and I am really worried about the broken trust modeling that many of these systems are likely to suffer from. In my opinion, the kinds of time and state errors that are so pervasive in MMORPGs are the same kinds of attacks we can expect to see against SOA systems and Web 2.0 systems in the future.

If this interests you, you should read that IEEE *Security & Privacy* paper I mentioned before.

RIK: Can you (briefly) point out some lessons learned (or that should be learned) for future fat-client software programmers from the MMORPG world?

GARY: The basic lesson is that there is no substitute for considering the attacker's perspective while designing and implementing software. This is a very basic software security lesson that still needs emphasis. Expect attackers to do precisely what "nobody would do" and plan for it. Misuse and abuse cases at the requirements level (as I describe in my book *Software Security* [4]) are very useful for this.

Another critical lesson is that trust boundaries are important. Understanding who controls what information and what that means when you're trying to protect your system is essential. This gets particularly tricky when a distributed system is "massive" (with, say, over 250,000 simultaneous users) and big swaths of the system are outside the game's control. MMORPGs have thorny and interesting trust issues.

RIK: In my experience, programmers focus on the functional requirements of the software they have been asked to create. Just getting the software to come close to meeting the requirements is hard enough.

It appears that you are expecting programmers to think outside the box of the specifications for their software. Shouldn't there be a way that specifications could be written in a manner that forced programmers to expect aberrant behavior?

GARY: Yes. This is the main challenge faced by software security. We have had some early success among developers and architects building software

for financial institutions, and there is every reason to believe this can work for other kinds of developers as well.

In my book *Software Security* I spell out these kinds of best practices (called touchpoints in my parlance) in great detail. There is some brief coverage in the last chapter of *Exploiting Online Games* as well.

RIK: Can you imagine a game client that the gaming company could totally trust? Would doing so require hardware support?

GARY: I don't think we need to create completely trustworthy clients. What we do need to do is be much more skeptical about the data that untrustworthy clients send the server. If a client is being used to cheat, that should be made to stick out like a sore thumb at the server. That way, impossible goals like "create a completely trustworthy client" can be safely avoided.

Hardware support might help create a trustworthy platform, but these platforms would be trustworthy by the corporations that control them, not by the people who own them! In any case, approaches relying on hardware have also been successfully attacked (think satellite TV systems and the xBox).

RIK: You mentioned some advanced game-hacking techniques in your IT, for example, using multicore systems, and running the game-hacking thread on one core while the game runs on the other core. Have you actually succeeded in using this technique? How about running a game client from within a VM? Do gaming companies attempt to determine whether their software is running in a VM (or a debugging tool)?

GARY: We have a section in the book related to this question in Chapter 7: Building a Bot. Many of the techniques we describe there do work in the lab. We drew a line in the book and decided that it would not be helpful to the industry to release undetectable botting kits based on the techniques we outline. The fact is that we're not the only people thinking about these things, though. There are plenty of others writing and sharing game-hacking code all over the Internet.

An undetectable botting system is very valuable. All of the sweatshops in China (where people are paid a pittance to play the game and develop virtual wealth, which is then sold in a middle market) have created quite a demand for botting systems that are not yet banned (that is, that are not obviously detectable).

There are lots of things that remain to be explored, including rootkit-level bots, VM-based hacks, multiprocessor attacks of many kinds, and so on. It's important that the game companies understand just how dedicated to cheating some of their adversaries are.

My hope is that software security will progress nicely in the online game world and that will in turn teach us valuable lessons for building other software. Ultimately, I am a huge proponent of software security.

REFERENCES

[1] G. Hoglund and G. McGraw, *Exploiting Online Games: Cheating Massively Distributed Systems* (Reading, MA: Addison-Wesley, 2007).

[2] G. Hoglund and J. Butler, *Rootkits: Subverting the Windows Kernel* (Reading, MA: Addison-Wesley, 2005).

[3] Article on trust boundary issues: http://www.cigital.com/papers/download/attack-trends-EOG.pdf.

[4] G. McGraw, *Software Security: Building Security In* (Reading, MA: Addison-Wesley, 2006).

MIKE RASH

IDS signature matching with iptables, psad, and fwsnort

Michael Rash holds a Master's degree in Applied Mathematics and works as a Security Architect for Enterasys Networks, Inc. He is the creator of the cipherdyne.org suite of open source security tools and is author of the book *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort,* published by No Starch Press.

mbr@cipherdyne.org

THE ANALYSIS OF LOG DATA IS BECOMing an increasingly important capability as more applications generate copious amounts of run-time information. This information often has interesting things to say for those who are listening (including evidence of events that are significant from a security perspective), but the sheer volume of information often requires automated tools to make sense of the data. The iptables firewall is built on top of the Netfilter framework in the Linux kernel, and it includes the ability to create verbose syslog messages of the network and transport layer headers associated with IP packets. In addition, through the use of the iptables string match extension, the application layer can be searched for evidence of malicious activity and iptables can then log or take action against such packets.

This article explores the use of psad and fwsnort [1] to automate the analysis of iptables log messages with a particular emphasis on passive OS fingerprinting and the detection of application-layer attacks. Both psad and fwsnort are open-source software released under the GNU Public License (GPL). Some familiarity with iptables and the Snort rules language is assumed in this article [2]. Also, see the INSTALL file bundled with the psad and fwsnort sources for installation instructions.

Network Setup and Default iptables Policy

I will illustrate network traffic against a Linux system that is protecting a small internal network with an iptables policy that implements a default "log and drop" stance for any traffic that is not necessary for basic connectivity. In particular, the iptables policy provides NAT services to allow clients on the internal network to issue DNS and Web requests out through the firewall (with the internal network having the RFC 1918 subnet 192.168.10.0/24 and the external interface on the firewall having a routable IP address), and the firewall accepts SSH connections from the internal network. The iptables policy uses the Netfilter connection tracking capability to allow traffic associated with an established TCP connection to pass through; also allowed are packets that are responses to UDP datagrams (which may include ICMP

port unreachable messages in response to a UDP datagram to a port where no server is bound). All other traffic is logged and dropped (with iptables log messages reported via the kernel logging daemon klogd to syslog). This iptables policy is implemented by the following iptables commands [3]:

```
iptables -F INPUT
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p tcp -s 192.168.10.0/24 --dport 22 \
   -m state --state NEW -j ACCEPT
iptables -A INPUT -i ! lo -j LOG --log-ip-options \
   --log-tcp-options --log-prefix "DROP "

iptables -F FORWARD
iptables -P FORWARD DROP
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.10.0/24 --dport 80 -m state \
   --state NEW -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.10.0/24 --dport 443 -m state \
   --state NEW -j ACCEPT
iptables -A FORWARD -p udp -s 192.168.10.0/24 --dport 53 -j ACCEPT
iptables -A FORWARD -i ! lo -j LOG --log-ip-options \
   --log-tcp-options --log-prefix "DROP "

iptables -t nat -A POSTROUTING -o eth0 -s 192.168.10.0/24 \
   -j MASQUERADE
```

Passive OS Fingerprinting

With the iptables policy active on the Linux system, it is time to see what it can show us from a logs perspective. First, from a system on the internal network (with hostname "int" and IP address 192.168.10.50), we attempt to initiate a TCP connection to port 5001 on the firewall (where the firewall's internal IP is 192.168.10.1):

```
[int]$ nc 192.168.10.1 5001
```

This results in the following iptables log message for the incoming TCP SYN packet, which is blocked and logged by iptables:

```
Sep 13 21:22:24 fw kernel: DROP IN=eth1 OUT= \
MAC=00:13:46:3a:41:4b:00:0c:41:24:56:37:08:00 \
SRC=192.168.10.50 DST=192.168.10.1 LEN=60 TOS=0x00 PREC=0x00 \
   TTL=64 ID=51104 DF PROTO=TCP SPT=57621 DPT=5001 \
WINDOW=5840 RES=0x00 SYN URGP=0 \
OPT (020405B40402080A1ECB4C4C0000000001030302)
```

The log message contains, among other things, source and destination IP addresses, the IP ID and TTL values, source and destination port numbers, TCP flags (with just the SYN flag being set in this case), and the options portion of the TCP header (preceded by the "OPT" string). From the perspective of passively fingerprinting the operating system that generated the TCP SYN packet against the firewall, the most interesting fields in the log message are as follows:

- IP length: LEN=60
- TTL: TTL=64
- The Don't Fragment bit: DF
- TCP window size: WINDOW=5840
- TCP flags: SYN
- TCP options: OPT (020405B40402080A003F83040000000001030302)

(Note that the TCP options string is only included within an iptables log message if the --log-tcp-options argument is given on the iptables command line when adding the LOG rule.) These fields are important because they are the same fields that the best-known passive OS fingerprinting software, p0f, uses to fingerprint operating systems [4]. This illustrates the completeness of the iptables logging format, because it is possible to implement the same passive OS fingerprinting algorithm used by p0f but use iptables log messages as input instead of sniffing packet data off the wire with a packet capture library. The psad project implements the p0f fingerprinting algorithm over iptables log messages, and the TCP log message just listed conforms to the following p0f fingerprint:

```
S4:64:1:60:M*,S,T,N,W2:   Linux:2.5::Linux 2.5 (sometimes 2.4)
```

This fingerprint specifies a series of requirements on packet headers separated by colons and is read as follows:

- "S4" requires that the TCP window size be four times as large as the Maximum Segment Size (MSS). The MSS value is part of the TCP options field.
- "64" matches the TTL value and requires that the initial TTL is 64. (This value has to be estimated for packets that traverse the open Internet.)
- "1" requires that the Don't Fragment bit is set.
- "60" requires that the overall size of the SYN packet (including the IP header) be 60 bytes.
- "M*,S,T,N,W2" describes the options field of the TCP header; "M*" means any MSS size, "S" means Selective Acknowledgment is OK, "T" means that the TCP options contain a time stamp, "N" requires a NOP option, and "W2" requires a window scaling value of 2.

Decoding the options string from the iptables log message is the most complex portion of the fingerprinting activity. The options string follows Type Length Value (TLV) encoding, where each TCP option has one byte for the option type, one byte for the option length, and a variable number of bytes for the option value [5]. Hence, the options string decodes to the following, which matches the requirements of the "Linux:2.5::Linux 2.5" p0f signature (and psad reports this fingerprint within email alerts that it generates [6]):

```
- MSS: 1460
- Selective Acknowledgment is OK
- Timestamp: 516639820
- NOP
- Window scaling value: 2
```

Snort Rule Matching with fwsnort

In the previous section, we saw that it is possible to collect iptables log messages for SYN packets sent from arbitrary hosts and, in many cases, infer the OS that generated these packets. Passively fingerprinting operating systems is a nice trick and can reveal interesting information about an attacker, but in the threat environment on the Internet today the real action is at the application layer (OS fingerprinting only requires the inspection of network and transport layer headers). To get a feel for how important application-layer inspection is to computer security, one need only examine the Snort rule set. In Snort version 2.3.3 (the last version of Snort that included rules released under the GPL instead of the VRT service from Sourcefire), there are about 150 signatures out of 3,000 that only test packet headers and have no application-layer match requirement.

In the Snort rules language, elements that test the application layer include the "content," "uricontent," "pcre," "byte_test," "byte_jump," and "asn1" keywords, whereas elements such as "flags," "ack," "seq," and "ipopts" (among others) test packet header fields. Maintaining an effective intrusion detection stance for network traffic requires the ability to inspect application-layer data, and 95% of all Snort rules are focused on the application layer.

The fwsnort project translates Snort rules into iptables rules that are designed to detect (and optionally react to) the same attacks, and the Snort 2.3.3 rule set is packaged with fwsnort. Because the detection capabilities of iptables are limited to matches on strings via the string match extension [7], many Snort rules (such as those that contain a pcre match) cannot be translated. Still, about 60% of all Snort 2.3.3 rules can be translated into iptables rules by fwsnort because iptables provides a flexible set of facilities to the user for matching traffic in kernel space. Chief among these facilities is the ability to match on multiple content strings instead of just a single string; iptables 1.3.6 introduced this capability by allowing multiple matches of the same type to be specified on the iptables command line. In the following we will see an example of a Snort rule that looks for two malicious content strings returned from a Web server and will see how iptables can be made to look for the same attack in network traffic.

Some of the most interesting and devastating attacks today exploit vulnerabilities in client applications that are attacked via malicious or compromised servers. Because in many cases thousands of independent client applications communicate with popular servers, an attacker can take advantage of this multiplying effect just by compromising a heavily utilized server and forcing it to launch attacks against any hapless client who connects to it.

An example of a Snort rule that looks for a client-side attack against a Web browser is rule ID 1735, which is labeled as "WEB-CLIENT XMLHttpRequest attempt." This rule detects a possible attempt to force a Web browser to return a list of files and directories on the system running the browser back to the attacker, via the responseText property, after redirecting the browser to point to the local filesystem. Fortunately, this attack applies to older versions of the Netscape and Mozilla browsers, but if a Web server sends data that matches this Snort rule back to a Web browser running on my network, I would want to know about it regardless of whether or not the browser is vulnerable. The XMLHttpRequest attack is tracked in the Common Vulnerabilities and Exposures (CVE) database as CVE-2002-0354 [8]. Here is the Snort rule for this attack:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any \
(msg:"WEB-CLIENT XMLHttpRequest attempt"; \
flow:to_client,established; content:"new XMLHttpRequest|28|"; \
content:"file|3A|//"; nocase; reference:bugtraq,4628; \
reference:cve,2002-0354; classtype:web-application-attack; \
sid:1735; rev:7;)
```

Note that the Snort rule is looking for two content strings that emanate from an external Web server (with the source IP being $EXTERNAL_NET and the source port being $HTTP_PORTS) back to a Web client that is on the internal network (with the destination IP being $HOME_NET and the destination port being "any," since the local TCP stack would choose a random high port for the Web session). The two content strings are "new XMLHttpRequest|28|" and "file|3A|//". Each of these strings specifies one byte by its hex code between pipe characters: "|28|" in the first content string, and "|3A|" in the second. So, when translating this Snort rule into an iptables rule, we must account for that. With fwsnort installed, let's use it to translate

Snort rule ID 1735 and then load it into the iptables policy on the firewall (some output below has been abbreviated):

```
[fw]# fwsnort --snort-sid 1735
[+] Parsing Snort rules files...
[+] Found sid: 1735 in web-client.rules
[+] iptables script: /etc/fwsnort/fwsnort.sh
[fw]#  /etc/fwsnort/fwsnort.sh
[+] Adding web-client rules.
```

Examine the /etc/fwsnort/fwsnort.sh script and you can see the iptables command below. This command uses the --hex-string argument so that the Snort content fields can be specified "as is" within the iptables command (with the bytes between the pipe characters being properly interpreted), and the rule target instructs iptables to log any matching packet with the prefix "[1] SID1735 ESTAB". This prefix informs the user that Snort rule ID 1735 was detected within an established TCP connection (fwsnort interfaces with the Netfilter connection tracking capability for this), and the rule is the first rule "[1]" within the FWSNORT_FORWARD_ESTAB chain.

The "--algo bm" argument instructs the string match extension to use the Boyer-Moore string-matching algorithm to conduct the application-layer match. With kernels in the Linux 2.6 series, the string match extension leverages a text-matching infrastructure implemented in the kernel which supports multiple string-matching algorithms; the Boyer-Moore algorithm exhibits excellent performance characteristics and is commonly used within open source and proprietary intrusion detection systems. Finally, the iptables comment match is used to include the Snort rule "msg," "classtype," and "reference" fields within the iptables rule for easy viewing under a command such as "iptables -v -n -L FWSNORT_FORWARD_ESTAB." We then have:

```
$IPTABLES -A FWSNORT_FORWARD_ESTAB -d 192.168.10.0/24 -p tcp \
--sport 80 -m string --hex-string "new XMLHttpRequest|28|" \
--algo bm -m string --hex-string "file|3A|//" --algo bm -m comment \
--comment "sid:1735; msg:WEB-CLIENT XMLHttpRequest attempt;
classtype:web-application-attack; reference:bugtraq,4628; rev:7; \
FWS:1.0.1;" -j LOG --log-ip-options --log-tcp-options --log-prefix \
"[1] SID1735 ESTAB "
```

Now let us simulate the XMLHttpRequest attack through the iptables firewall against an internal Web browser. For this, we use Perl and Netcat on a dummy Web server at IP 11.11.1.1 (a randomly selected IP address for illustration purposes only). The following Perl command sends data matching the two content fields in Snort rule ID 1735 back to the Web client as soon as it connects:

```
[webserver]# perl -e 'printf "new XMLHttpRequest\x28AAAAAAAfile\x3A//"' |nc -l -p 80
[int]$ nc -v 11.11.1.1 80
Connection to 11.11.1.1 80 port [tcp/www] succeeded!
new XMLHttpRequest(AAAAAAAfile://
```

The last line here shows that the Web client received data that matches the Snort rule; iptables has not interfered with the traffic and has happily let it pass into the internal network. On the firewall, we see the following iptables log message (note that the "[1] SID1735 ESTAB" log prefix and the ACK and PSH flags are set, since this packet was matched within an established TCP connection):

```
Sep 14 08:39:24 fw kernel: [1] SID1735 ESTAB IN=eth0 OUT=eth1 \
SRC=11.11.1.1 DST=192.168.10.50 LEN=85 TOS=0x00 PREC=0x00 TTL=63 \
```

```
 ID=23507 DF PROTO=TCP SPT=80 DPT=34646 WINDOW=91 RES=0x00 ACK PSH \
URGP=0 OPT (0101080A650A550A1F663D7A)
```

At this point we are confident that iptables is able to detect the attack. However, because iptables is a firewall, it is also inline to the traffic whereas Snort (unless deployed in inline mode) is merely able to passively monitor the traffic. Let us take advantage of this by changing the fwsnort command. This time we use the --ipt-reject command-line argument to have fwsnort use the iptables REJECT target against the Web connection in order to knock it down with a TCP RST packet:

```
[fw]# fwsnort --snort-sid 1735 --ipt-reject
[+] Parsing Snort rules files...
[+] Found sid: 1735 in web-client.rules
[+] iptables script: /etc/fwsnort/fwsnort.sh
[fw]#  /etc/fwsnort/fwsnort.sh
[+] Adding web-client rules.
```

Let us run the attack simulation once more:

```
[webserver]# perl -e 'printf "new \
   XMLHttpRequest\x28AAAAAAAfile\x3A//"' |nc -l -p 80
[int]$ nc -v 11.11.1.1 80
Connection to 11.11.1.1 80 port [tcp/www] succeeded!
```

We see that the client is again able to successfully establish a TCP connection with the Web server (that is, the TCP three-way handshake is allowed to complete), but no data comes across. This is because of the TCP RST generated by the REJECT target against the Web server. The REJECT target only sends the RST to the IP address that triggered the rule match within iptables, so the Web client never sees it. However, the iptables REJECT target is a terminating target, so it also drops the matching packet (in this case the packet that contains the XMLHttpRequest string). Hence, the malicious traffic never makes it to the targeted TCP stack, and this is an important capability when some attacks only require a single packet in order to do their dirty work (the SQL Slammer worm is a good example). Only an inline device can prevent individual malicious packets from reaching their intended target.

On the firewall, fwsnort has also created a logging rule that produces the following log message (note that the log prefix now includes the string "REJ", indicating that the packet was rejected):

```
Sep 14 08:41:24 fw kernel: [1] REJ SID1735 ESTAB IN=eth0 OUT=eth1 \
SRC=11.11.1.1 DST=192.168.10.50 LEN=85 TOS=0x00 PREC=0x00 TTL=63 \
  ID=46352 DF PROTO=TCP SPT=80 DPT=52078 WINDOW=91 RES=0x00 ACK PSH \
URGP=0 OPT (0101080A650ACA031F66B26C)
```

Conclusion

This article has focused on two relatively advanced usages of functionality provided by the iptables firewall: the completeness of the log format, which makes passive OS fingerprinting possible, and the ability to inspect application-layer data for evidence of malicious activity. The psad and fwsnort projects automate both of these tasks and can provide an important additional security layer to an iptables firewall. The Snort community has guided the way to effective attack detection on the Internet today, and iptables can leverage the power of this community to extend a filtering policy into the realm of application inspection. Armed with such a policy, iptables becomes a sentry against application-layer attacks.

REFERENCES

[1] http://www.cipherdyne.org/.

[2] Snort rule writing documentation: http://www.snort.org/docs/writing_rules/chap2.html.

[3] A script that implements the default iptables policy can be downloaded from http://www.cipherdyne.org/LinuxFirewalls/ch01/iptables.sh.tar.gz.

[4] Passive OS fingerprinting is really passive stack fingerprinting. That is, the IP and TCP stacks used by various operating systems exhibit slight differences, and detecting these differences (i.e., "fingerprinting" the stack) can allow the operating system that uses the stack to be guessed; see http://lcamtuf.coredump.cx/p0f.shtml.

[5] There are two exceptions to this: The "NOP" and "End of Option List" options are only one byte long. See RFC 793 for more information.

[6] See http://www.cipherdyne.org/psad/docs for examples of such alerts.

[7] The iptables u32 extension is being reintegrated with the Netfilter framework after it was deprecated late in the 2.4 series kernel, so more complicated arithmetic tests can be written against both packet headers and application-layer data. The u32 extension essentially emulates the "byte_test" operator in the Snort rules language.

[8] See http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0354.

DAVID N. BLANK-EDELMAN

practical Perl tools: Perl meets Nmap and pof

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 20+ 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

AS COMPUTER SECURITY CONCERNS start to become fodder for the more mainstream media and the digital world starts to converge in weird ways, I can easily envision the day where late-night channel flipping will land me on a commercial that goes something like this: "Hey Boys and Girls! Here at K-Tel Records and Software, have we got something special for you. That's right, a compilation of all of your favorite security tools. Who could forget such great hits as . . ." (Hey, don't scoff at the idea of this sort of convergence. Do a Google search on "Nmap porn" and then see if you are still snickering.)

If such a thing does come to pass, there are two tools that would definitely be on that album: Nmap and p0f. We're going to take a look at how to drive both of these tools from Perl so you can build some interesting applications that take advantage of their superpowers. For those of you who are new on the scene and haven't heard of either of these tools, let me give you a very quick rundown.

Nmap

Nmap (http://insecure.org/nmap/index.html) is one of the most impressive security scanners you'll ever encounter. This free tool can pull out virtually every known trick in the book to probe even huge networks quickly and return a list of devices present on your network. In most cases it can tell you what network ports are open, the services provided on those ports, and often even the version of the operating system these devices are running. If you've never played with Nmap, you should do so (on a network you have the legal and ethical right to explore).

Here's an excerpt from a sample Nmap scan of a host:

```
$ sudo nmap -O -sV 192.168.0.9
Starting Nmap 4.20 ( http://insecure.org ) at 2007-09-27 22:07 EDT
Interesting ports on 192.168.0.9:
Not shown: 1693 filtered ports
PORT     STATE  SERVICE       VERSION
22/tcp   open   ssh           OpenSSH 4.3 (protocol 1.99)
139/tcp  open   netbios-ssn
445/tcp  open   microsoft-ds  Microsoft Windows XP microsoft-ds
3389/tcp open   microsoft-rdp Microsoft Terminal Service
MAC Address: 00:0B:DB:54:3A:22 (Dell ESG Pcba Test)
```

```
Device type: general purpose
Running (JUST GUESSING) : Microsoft Windows 2000|XP|2003 (90%)
Network Distance: 1 hop
Service Info: OS: Windows
Nmap finished: 1 IP address (1 host up) scanned in 44.524 seconds
```

pof

p0f (http://lcamtuf.coredump.cx/p0f.shtml) describes itself as a "passive OS fingerprinting tool." In some regards it can do something even more impressive than Nmap. Nmap functions by spewing all sorts of interesting packets at its targets to gather information, but p0f is totally passive. It doesn't need to send a single packet to work its magic. p0f can sit on a network and just listen, showing you information about the hosts on the network based on what it hears.

Here's some sample output from p0f showing my home network being scanned by other hosts over my cable modem connection (thanks guys!):

```
$ p0f -i en1 -p
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'en1', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
202.163.213.11:64162 - FreeBSD 6.x (2) (up: 1148 hrs)
  -> 192.168.0.3:49153 (distance 22, link: ethernet/modem)
85.164.236.96:28029 - Windows 2000 SP2+, XP SP1+ (seldom 98) [priority1]
  -> 192.168.0.3:49153 (distance 27, link: pppoe (DSL))
90.193.162.36:4888 - Windows 2000 SP4, XP SP1+ [priority1]
  -> 192.168.0.3:49153 (distance 20, link: ethernet/modem)
```

Clearly these are really powerful tools. Wouldn't it be great to be able to harness their power from your own programs? Let's work on that now.

Nmap from Perl

There are two very capable Perl modules for Nmap control on CPAN: Nmap::Parser and Nmap::Scanner. They have roughly the same functionality. I tend to like Nmap::Parser better because it doesn't force you into using iterators (i.e., get_next_port()), but you should look at both and see which one catches your fancy.

Both Nmap::Parser and Nmap::Scanner have two modes of operation: batch and event/callback. With the first mode you specify the parameters for your scan using the conventions of that module (sometimes it is just passing explicit command-line arguments to Nmap), tell the module to kick off the scan, and then sit back and wait for the scan to complete. Once Nmap has done its work, the module makes available to you a few Perl data structures containing the results. With these answers in hand, your program can go off and make use of all of the yummy data you've collected.

For the event/callback-based way of working, in addition to defining the scan parameters at the start, you also provide the module with code that gets run as certain events are triggered (i.e., your code gets a callback at certain points in the scan run). These events can include things such as "found a host" or "found an open port."

There are a few advantages of using event-based approaches over batch approaches. For instance, if you are performing a scan on a huge network block an event-based program will likely be less memory-intensive, because

you have the chance to store only the data you care about instead of the results from the entire scan. Your program can be a bit more responsive because you can choose to act on partial results, perhaps spinning off tasks related to the results in parallel (see the February 2007 column) or even aborting early if you see fit. The downside of this approach is that your program is now responsible for the collection and storage of the data found, something the batch modes make easy. Just FYI: Nmap::Scanner has a few more event-based hooks than Nmap::Parser, but they both work roughly the same way.

Let's look at one batch scan and one event-based scan using Nmap::Parser.

This code looks for all of the hosts specified on the command line that have their HTTP, HTTPS, and SSH ports open all at the same time:

```
use Nmap::Parser;

my $nmapexec = "/opt/local/bin/nmap";  # location of executable
my $nmapargs = "-p 80,443,22";         # look for http, https, and SSH

my $np = new Nmap::Parser;

# scan the hosts listed on the command line
$np->parsescan( $nmapexec, $nmapargs, @ARGV );

# iterate over the result set looking for the hosts that have
# all 3 ports open
for my $host ( $np->all_hosts() ) {
   my @open = $host->tcp_open_ports();
   print "found: " . $host->hostname() .
        " (" . $host->addr() . ")" . "\n"
      if scalar @open == 3;
}
```

Here's the same code using a callback-based approach:

```
use Nmap::Parser;

my $nmapexec = "/opt/local/bin/nmap";  # location of executable
my $nmapargs = "-p 80,443,22";         # look for http, https, and SSH (all 3)

my $np = new Nmap::Parser;

# call print_all_open each time we finish scanning a host
$np->callback( \&print_all_open );

# scan the hosts listed on the command line
$np->parsescan( $nmapexec, $nmapargs, @ARGV );

# print only the hosts that have all three ports in question open
sub print_all_open {
   my $host = shift;
   my @open = $host->tcp_open_ports();

   print "found: " . $host->hostname() .
        " (" . $host->addr() . ")" . "\n"
      if scalar @open == 3;
}
```

So what can you do programmatically with the results of an Nmap scan? There are tons of applications, including network visualization, policy enforcement, and security testing. Let's look at an example that demonstrates the first two on that list. Let's say our employer is either really paranoid or just highly idiosyncratic and has a strict policy that the MySQL servers on site are not allowed to provide mail services. Here's some code that maps a

network block looking for the hosts with either an SMTP or a standard MySQL port open:

```
use Nmap::Parser;
use Graph::Easy;

my $nmapexec = "/opt/local/bin/nmap";    # location of executable
my $nmapargs = "-p 25,3306";
my %services = ( 25 => 'SMTP', 3306 => 'MYSQL' );
my $np = new Nmap::Parser;

# scan the hosts listed on the command line
$np->parsescan( $nmapexec, $nmapargs, @ARGV );

my $graph = Graph::Easy->new();

for my $host ( $np->all_hosts() ) {
   # graph the last octet of the IPv4 address
   my $hostnumber = ( split( /\./, $host->addr() ) )[3];
   for my $port ( $host->tcp_open_ports() ) {
      $graph->add_edge( $services{$port}, $hostnumber );
   }
}

print $graph->as_graphviz();
```

If we run this script like so:

```
$ perl nmap.pl 192.168.0.0/24|dot -Tpng -o fig1.png
```

we get a picture like the one in Figure 1.

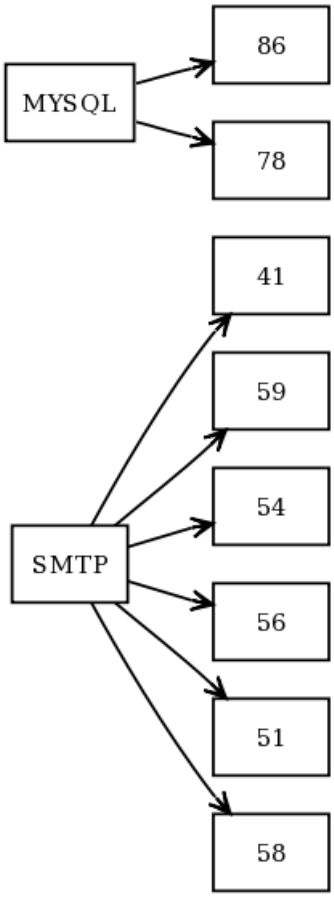


FIGURE 1

Even a simple picture like this can be helpful. Besides providing an easy way to see if our policy is being followed, it also provides an easy-to-scan display of the hosts providing a service. Should 192.168.0.41 be offering SMTP services when we did not expect it, it would likely be what we in the technical world call "a bad thing." It would be pretty easy to write code that compares the results between Nmap runs and highlights any differences (perhaps changing the color of a box in the final output). Nmap::Parser comes with a sample script called nmap2sqlite that stores the output of a scan in an SQLite database if you are looking for a little help in that direction.

There are plenty of visualization tools besides those offered by Graphviz/Graph::Easy that would be happy to eat the Nmap data we collected and spit out pretty diagrams of the network.

p0f from Perl

Nmap is a fantastic tool for mapping a known (and potentially large) set of hosts, but what if you don't know which hosts are interesting to you? In the case of publicly visible hosts such as Internet-facing Web servers, you don't really have a good way of knowing which hosts will connect to you. p0f is a good choice for those situations where the network traffic comes to you. It can provide some basic information about a host simply by listening to the packets sent by that host.

This information can be useful for any number of reasons. Perhaps you want to know roughly what percentage of your loyal Web site users are using Linux. Maybe you want to create a policy that says machines running operating systems written near Seattle should be scanned for vulnerabilities when they first arrive on a network. p0f can help with these goals. I know of a commercial anti-spam product that relies on information from p0f to make decisions

about how to handle incoming connections (on the theory that hijacked Windows boxes make up the vast majority of a spammer's arsenal these days).

p0f itself can run in two different modes. The most common usage is to have p0f listen for packets from the wire or read a standard tcpdump capture file and output the fingerprint results to STDOUT or to another file. Another possibility is to run p0f as a daemon that receives packets from another program passed over a local stream socket. This works well in those cases where you already have something listening for packets that would like to consult p0f as it goes along. The Perl module Net::P0f can handle both of these modes. We're only going to look at code for operating in the first mode. See the submodule Net::P0f::Backend::Socket in the Net::P0f distribution for information on using p0f in socket-read mode.

Two meta things need to be said about Net::P0f before we see some code:

- The module hasn't been updated since 2005, although it seems to work even with recent versions of p0f.
- Net::P0f operates using callbacks, similar to what we saw for Nmap::Parser. You are expected to define a subroutine that will receive fingerprint information as it is produced. If you want to keep tabs on everything that has been seen since the program first started listening, you'll need to write the coalesce code yourself.

Let's look at a very simple example so you can get an idea of how things can work using this module. This code listens for 25 initial connect packets (SYN) destined for any host on the network and reports back an IP address and details for any machine p0f determines to be a Windows box:

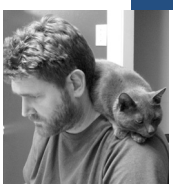
```
use Net::P0f;

# listen on /dev/en1 for all packets
my $p0f = Net::P0f->new( interface => 'en1', promiscuous => 1 );

# listen for 25 packets and then exit, handing each packet to our callback
$p0f->loop( callback => \&show_win32, count => 25 );

# print information on host if it is identified as a Windows machine
sub show_win32 {
   my ( $self, $header, $os_info, $link_info ) = @_;

   print $header->{ip_src} . ":"
      . $os_info->{genre} . " "
      . $os_info->{details} . "\n"
      if $os_info->{genre} eq "Windows";
}
```

The good news is that using Net::P0f really doesn't get any more sophisticated than that. From here you can use any of your standard Perl techniques to slice, dice, collate, or analyze the data you collect. You could also easily add code from our last section to have Nmap scan hosts running certain operating systems or coming from selected link media identified by p0f.

Now you have the information you need to write your own applications that harness the power of Nmap and p0f. Create something interesting and report back! Take care, and I'll see you next time.

DAVID JOSEPHSEN

iVoyeur: mystical flows



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

YOU KNOW YOU HAVE A HEALTHY, well-implemented monitoring system when people in other departments begin approaching you with their information requirements. Monitoring is usually something we prefer to do ourselves. It's just unlikely that some other organizational unit will collect the information you want in the way you want it collected without making life difficult for you in the form of bloated, unstable, and/or insecure agent software. It's rare in my experience for tech staff to even consider whether anyone else in the organization is already doing systems monitoring before implementing their own tools.

So when someone voluntarily comes to you and asks if your system can monitor this or that, you know you're doing things well. By this metric I've had more than my share of failures thus far in my career. I take solace in the words of an old martial arts instructor of mine who once said, "Nobody ever learned anything from winning a fight," and man, have I learned plenty. But rather than dwell on my "temporary setbacks," I'd like to share with you a success I've had, because I think there's something to be learned from it as well. My crowning achievement—arguably the pinnacle of my success in implementing monitoring systems—came one day when I was asked by a Microsoft SQL Server DBA for Nagios to monitor the tablespace on his database servers. It doesn't sound like much, but then you haven't met my DBAs.

This came shortly after a meeting to determine "why the monitoring system was sending false alarms to the DBA teams." Even given the meeting subject, I was not expecting the hostility that greeted a fellow sysadmin and me when we walked into the conference room. The DBAs had brought pages of specific instances of "false alarms," and I hadn't even brought a laptop. Oops. So I must have looked a bit panic-stricken when the VP of software development passed out a copy of the monitoring system's "errors" and began to discuss with those in attendance how the "obviously flawed" system was sending bogus disk capacity warnings and that if it couldn't even monitor disk capacity correctly, it would need to be replaced. They already had their own monitoring software, so they weren't bucking for budget. Something political was probably going on, but to this day I'm not sure what their actual goal was.

I was only a few words into stammering out something like "I'll have to look into this and get back to you" when my teammate, who hadn't forgotten his laptop, interrupted with "These aren't false alarms." He had generated a Nagios trends graph of the disk service for one of the database servers and was visually correlating this with an RRDtool graph of disk utilization for the same server. He could see at a glance that a disk partition was filling up every few days at a certain time, and then was being emptied shortly thereafter.

Faced with the graphs, the VP turned to the DBA team and asked, "Well, is one of you clearing space on partition X on server Y every Tuesday and Thursday?" A moment of silence and head-shaking ensued followed by a "Huh?" from the corner of the room. It came from a DBA who until this moment had been face down typing furiously into a laptop. The VP repeated his question, to which the DBA replied, "Yeah I clear out a bunch of temp files whenever I get notifications from the monitoring system. It happens a couple times a week. I put in a request for more SAN but haven't heard back yet." Evidently he hadn't paid much attention to the meeting subject line either.

From that day on, the DBA team projected upon the monitoring system a sort of mythical all-knowingness. They often assumed we had data that we didn't (though when they did this we usually quickly added it; there's little difference between presumption and permission IMO). I was in the hallway on the way back to my desk from that meeting when one of the DBAs approached and asked if he could get added to the "table-space notifications." We weren't monitoring the table-space at that point, but needless to say we began that afternoon. Our DBAs as a group were a fiercely protective bunch. I took it as a huge compliment that he had assumed we were monitoring the innards of their precious databases and that he was OK with that.

There are a handful of monitoring technologies that can pack large amounts of very specific, historically relevant data in an easy-to-use, accessible format. These are the tools that give your monitoring system an all-knowing air, which, as I learned from this episode, is a wonderful thing to have on your side. So, this being a security-focused issue, I thought I'd take the opportunity to talk about one of the best of this class of tools: NetFlow. Just as RRDtool made us witch doctors to the DBAs, NetFlow data can make you a mystic to your security and NOC staff.

NetFlow began life as a Cisco proprietary protocol for traffic accounting information. There is a fledgling industry standard called IPFIX [1], which is based on Cisco NetFlow v9. This standard, defined in RFC3917 [2], is generally reverse-compatible with Cisco's proprietary NetFlow protocol without modification, and it has already been implemented by several routing vendors.

Most modern routers from Cisco and Juniper, as well as various open-source implementations such as pfflowd [3] and nprobe [4], can export NetFlow data. It is not, however, supported by Cisco PIX firewalls or Cisco switches. Cisco Layer-3 switches such as the 6000 series can export NetFlow data, but since such switches offload and cache routing decisions to ASICs (Application Specific Integrated Circuits), the flows for packets that traverse the routing processor may be in a different format from those processed by the ASICs. In some cases it may not be possible to export flows for any packets but those that traverse the routing processor, so NetFlow data from these devices may be incomplete.

The overall NetFlow architecture may be superficially thought of in terms of a specialized, task-efficient syslog implementation. Routers or routing sys-

tems emit UDP NetFlow data to one or more centralized NetFlow collectors, where they are aggregated, stored, and possibly processed. There is no transport-layer encryption or signing. NetFlow emitters are called "probes." Probes are generally given the network socket of a listening collector and may be configured with options that change the details of the flow data.

A flow is loosely defined as a series of related, unidirectional packets, which represent half of a two-way conversation between two network entities. NetFlow data contains summary metadata about the connection it represents, and therefore flow data is only exported to the collector once the flow has ended and can be summarized. By default in Ciscoland, a flow begins when the relevant traffic is first detected and ends when one of the following criteria is met:

- For TCP traffic, when the connection is terminated (e.g., an RST or a FIN is encountered)
- When no related traffic has been seen in the last 15 seconds
- When the flow has continued for more than 30 minutes
- When the memory buffer containing the flow has filled up

The IPFIX standard allows for some user-defined criteria for detecting the beginning and end of a flow. In my experience, the Cisco criteria are usually sufficient.

Flow summary data is encapsulated into a flow record. Every flow is unique but may be represented by multiple flow records if, for example, the router's memory buffer fills up before the connection is terminated. Flow records are really great; they contain oodles of info about the connections they represent, including source and destination IP and port numbers, protocol type, type of service (TOS), number of octets and packets transmitted, time/day stamps for the beginning and end of the flow, source and destination AS numbers, input and output interface names, and even a bitmask representing the TCP flags that were set during the connection.

You may have noticed that flow records don't contain any application-layer data, but the network-layer data that's available is more than enough to get great visibility into what's happening on the network as well as detecting some much-hated and historically difficult to diagnose problems and attacks such as DDoS, worms, and viruses. But I'm getting ahead of myself. First, let's talk tools.

There are quite a few NetFlow collectors out there, some of which are commercial, such as Cisco's NetFlow Collector (NFC) [5], and hundreds of open-source tools of all description. When I look for tools in a monitoring context, I tend to optimize for flexibility so that I can easily add the data in question to the existing monitoring interface. I abhor one-off interfaces for every little thing, and for this reason I'm likely to choose lightweight command-line tools that don't have a lot of dependencies and don't make it difficult for me to get at the data.

There are at least a couple of new NetFlow tool papers a year, and I don't keep up with them as well as I probably should, because Ohio State University's flow-tools package [6] is a category killer for me. OSU flow-tools, a collection of small single-purpose tools that are designed to interoperate with each other via pipes, run the gamut of everything you might want to do with NetFlow, including collect data from a probe, "tee" data to real-time analyzers, perform query-based analysis on archived flow records, replay archived flows, and reassemble connections contained in multiple flow records. Judging by the quantity of graphical front-ends for visualizing data from flow-tools, I'm not the only one who considers it a category-killer.

The flow-tools install is a typical ./configure && make && sudo make install. Then, your routers need to be configured to export their NetFlow data. If they are Cisco routers, the following should work:

```
ip cef distributed
ip flow-export version 5 origin-as
ip flow-export destination 1.2.3.4 9800

interface FastEthernet0/1/0
 no ip directed-broadcast
 ip route-cache flow
 ip route-cache distributed
```

Once the data is being exported, the flow-capture tool can collect NetFlow data from the routers and archive it to disk. The flow-capture tool requires only a localip/remoteip/port tuple, and it automatically handles log file naming, compression, and rotation. Command-line options can change most aspects of its behavior, including the -D switch, which forces it not to daemonize so that you can run it under daemontools or the superserver of your choice. It's possible to connect to the flow-capture daemon on a TCP port to receive a real-time data feed, suitable for feeding to your favorite parsing engine as well.

Several tools in the package can do creative things with the flow data as it arrives. Flow-fanout is a tool for redistributing the data via the NetFlow protocol to additional collectors, and flow-mirror and flow-rsync copy the flow logs themselves to backup collectors.

The backbone of NetFlow analysis is the combination of the tools flow-cat, flow-filter, and flow-print. Because flow-tools writes flow data to binary log files, which are also optionally compressed, and each log file has a metadata header, a special cat tool called "flow-cat" is required to concatenate them. Output from flow-cat may be passed directly to flow-print, which outputs the records in a human-readable format, or it may be piped first to flow-filter, which filters the output using the criteria of your choosing.

Before you reach into your bag for cut, sort, awk, and grep, I should mention flow-stat. This tool can sort, summarize, and segment the output from flow-filter. There's a bit of a learning curve, but if you play around with it for a while I think you'll find a few formats you like, and once you have them in your head, the tool will save you a bunch of time. Formats 8, 9, and 10, the IP source/destination-based formats, are the ones I tend to use the most often. It's worth reading the man page to get an idea of what other formats are available.

Several conversion programs exist to move the data to external formats. The flow logs may be exported directly to delimited ASCII formats of various types with flow-export, the headers may be viewed with flow-header, and there are even a couple of tools included for exporting the data directly into RRDtool.

There are several special-purpose analysis tools such as flow-report and flow-dscan. With flow-report you get summary statistics in a predetermined format for a given collection of NetFlow logs. The IDS flow-dscan is intended to spot aberrant behavior in real-time traffic flows. But I'll leave it to you to play with those two; I'd like to focus on the core tools and show you some ways they can help you gain some visibility into your network traffic.

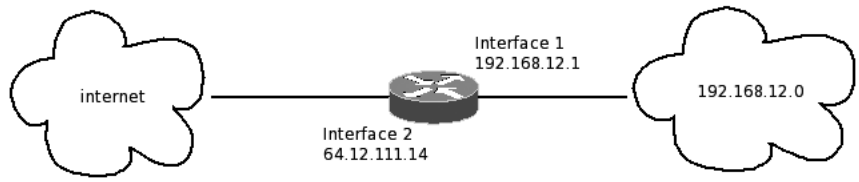


FIGURE 1: A ROUTER SEGMENTING AN RFC 1918 NETWORK FROM THE INTERNET

Let's start with a relatively simple example. The router in Figure 1 segments an internal 192 network from the "internet." Assuming the flow records for this router were in /var/flows/, with the following command you could find all of the internal Web servers that serviced external hosts:

```
flow-cat /var/flows/ | flow-filter -i2 -P80 | flow-stat -f8
```

In pseudocode, that's "filter the flow data for flows coming into interface 2 (-i2) destined for port 80 (-P80) and format the reports using the destination IP format (-f8)." Adding an "-S3" (sort on field 3) to the flow-stat command would have sorted the resulting report by the host that sent the most data.

The "-i" and "-P" switches are reversible via case sensitivity. In other words, had I specified "-I" instead, I would have filtered the data for flows exiting interface 2 instead of entering interface 2, and had I specified "-p80" I would have filtered the data for flows originating on port 80 instead of destined for port 80.

As you can probably imagine, flow-filter can also filter on host IPs and network ranges. To make this a bit simpler on the command line, you can create an ACL file and give macro-style names to IPs and ranges using Cisco-standard ACL syntax. For our network we might create the following macros in a file called my.acls:

```
ip access-list standard inside permit 192.168.12.0 0.0.0.255
ip access-list standard not_inside deny 192.168.12.0 0.0.0.255
```

Then we could, for example, find the top 10 bandwidth users on our network with something like:

```
flow-cat /var/flows | flow-filter -f./my.acls -Sinside | flow-stat -f9 -S3 | grep -v \# |
head -10
```

Perhaps, upon finding that the top bandwidth user was 192.168.12.42, you might want to know to which hosts this user was sending data. After adding

```
ip access-list standard topDude permit host 192.168.12.42
```

to our ACL file, the command

```
flow-cat /var/flows | flow-filter -f./my.acls -StopDude | flow-stat -f8
```

enables us to find out what ports this user is connecting to by using "-f5" in the flow stat command here. Upon finding out that topDude's traffic was destined for port 1434 (which is slammer worm behavior) on various remote hosts, we might search for any other internal hosts exhibiting the same behavior with, for example:

```
flow-cat /var/flows | flow-filter -f./my.acls -Dnot_inside -P1434 | flow-stat -f9
```

Since NetFlow data is in an offline-archived format, this technique can give you answers without requiring that you talk to the router directly, which is sometimes not possible. In scenarios such as DDoS attacks, where the NOC staff will be spending its time waiting at router ssh prompts trying to get a

handle on what is happening, NetFlow makes it trivial to quickly isolate the offending traffic and take action. At the same time it's a powerful capacity planning and forensics tool, putting months or years of detailed traffic information at your fingertips.

The NetFlow/flow-tools combination as a solution is very scriptable, easy to install, and gets along excellently with popular monitoring tools, including RRDtool and Nagios. But its "magic smoke" lies in the means it gives you to answer very specific questions about the network in an "on-demand" fashion. With NetFlow, you can create facts from hunches and make decisions out of options in a few keystrokes—the kind of thing that soothsayer reputations are built upon. If you aren't using NetFlow or something like it currently, you're missing out, and I highly recommend you give it a try.

Take it easy.

REFERENCES

[1] http://tools.ietf.org/wg/ipfix/.

[2] http://tools.ietf.org/html/rfc3917.

[3] http://www.mindrot.org/projects/pfflowd/.

[4] http://www.ntop.org/nProbe.html.

[5] http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/.

[6] http://www.splintered.net/sw/flow-tools/.

HEISON CHAK

Asterisk and LumenVox ASR

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003.

heison@chak.ca

IN THIS ARTICLE I INTRODUCE THE LumenVox Speech Recognition software and how it can be tied into a VoIP platform. Asterisk as an interactive voice response (IVR) system can utilize text-to-speech (TTS) and automatic speech recognition (ASR) technologies to improve system usability and encourage customer interaction. I will use the LumenVox ASR engine as an example.

Text-to-Speech

Text-to-speech technologies allow computer systems to convert text into spoken languages. TTS engines are capable of synthesizing male or female voices in various languages. Quality and accuracy are dictated by the affordability of the products; more expensive products tend to produce easier-to-understand phrases than their less expensive counterparts.

An open-source TTS engine, such as Festival, is suitable for starting a TTS application, but one may soon realize that such development efforts deserve more pleasant and accurate voices. Cepstral (~$30) offers a better synthesizer and is often involved in proof-of-concept of application developments. As we move into more sophisticated products (with a $300 to $3000 price tag), the synthesizers tend to become more accurate—instead of typing the text in a special phonetic way, the effort can be spent on achieving business requirements.

For example, if you want output of a TTS to sound like "Hello, how are you?" you may need to input the following text:

"Hah loow, how arh u" - Festival, open source
"Ha lo, how are you?" - Cepstral, $30 products
"Hello, how are you?" - better commercial products, >$3000

To present to a caller of Asterisk the synthesized audio, the Asterisk Gateway Interface (AGI) is often used. AGI allows external applications (shell, Perl, PHP, C, etc.) to have read and write access to the voice channel. Read access can allow these AGI scripts access to DTMF (or touch-tone) inputs, whereas write access enables the scripts to play out the TTS synthesized audio to the caller, in response to the caller's DTMF input or selection.

Automatic Speech Recognition

We often see the terms *speech recognition* and *voice recognition* used interchangeably. In academic con-

texts, voice recognition identifies the speaker, whereas speech recognition distinguishes the words or phrases spoken by the speaker (i.e., contents of the speech). Speech recognition converts spoken language into readable text, and there are two types of speech recognition systems—speaker-dependent and speaker-independent.

Speaker-dependent systems require prescribed text and phrases to be read by the speaker prior to conducting recognition; these materials are used to train the engine's vocabulary. A typical use for user-dependent ASR is speech transcription—using speech to replace nonspeech input. A properly trained speaker-dependent ASR system can transcribe speech at an average rate and accuracy of 17 words per minute and 84.9%, respectively.

Speaker-independent systems, often seen in telephony applications (e.g., auto attendant and interactive voice response systems), can be classified as ASR, computer-driven transcription of spoken language into readable text in real time; it handles pronunciation discrepancies in words or phrases among users. ASR has been viewed as the bedrock for building next-generation telephony applications. Not only is it a more natural input method—allowing a higher level of engagement from callers—but it also enables menu options to go beyond the limitation of a touch-tone keypad.

For example, imagine a caller who wants to speak with John Smith. On a traditional IVR/auto attendant, the caller may be entering "76484" to spell "Smith" and the system may respond with:

"Press 1 for Alan Smith, 2 for Bob Smith, 3 for Jan Smith, ..., 5 for John Smith."

If there is more than one match, the caller may have to wait for all of the choices before making a selection. With an ASR-capable IVR system, the caller can simply say "John Smith" and conversation can be established much more quickly. In addition to traversing the employee directory more easily by flattening the menus, ASR can also allow more free-form input, such as for country and city names, which are traditionally difficult to implement.

ASR Software for Asterisk

Sphinx is open-source ASR software that runs under Linux. However, Asterisk's handling of audio in 8 kHz samples is not readily compatible with the 16 kHz that Sphinx expects. Up-converting is an option but the end result is far from ideal. Sphinx has been reported to do a fine job on a native 16 kHz sample, just not on up-converted samples.

LumenVox (www.lumenvox.com) addressed the need for an affordable speech solution for the Asterisk community by integrating its speech recognition engine into Asterisk 1.4. Users can choose to purchase a supported platform with packages tailored toward deployment needs (e.g., number of concurrent recognitions and size of the vocabulary). LumenVox supports RedHat Enterprise and Fedora Core; Debian support has been added to the mix recently.

Customers can choose to download Speech Recognition Engine, License Server for their favorite platform, as well as an Asterisk Connector for the specific version of Asterisk they are running. There is also a Wintel-based Tuner, which can be used to fine-tune the speech engine.

In Debian, there are a number of required packages to run the Speech Recognition Engine:

```
# apt-get install libboost-filesystem1.33.1 libboost-program-options1.33.1
```

```
libboost-regex1.33.1 libboost-thread1.33.1
libboost-date-time1.33.1 libxul0d libc6-i686
iceweasel libnspr4-0d
```

After apt-get, the downloaded Debian LumenVox packages can be installed without a hitch:

```
# dpkg -i lumenvoxsre_8.0-106_i386.deb   # Speech Recognition Engine
# dpkg -i lumenvoxmrcpserver_8.0-106_i386.deb # License Server
```

The Asterisk connector contains binary distribution of a module, res_speech_lumenvox.so, that works with Asterisk's Generic Speech Recognition API (res_speech.so). With the API loaded, the LumenVox module can be loaded into Asterisk and is ready to test:

```
[lumenvox-test]
exten => s,1,Answer
exten => s,n,Wait(1)
exten => s,n,SpeechCreate
exten => s,n,SpeechLoadGrammar(yesno|${LUMENVOX_PATH}/ABNFBoolean.gram)
exten => s,n,SpeechActivateGrammar(yesno)
exten => s,n,SpeechBackground(beep)
exten => s,n,Verbose(1,Result was ${SPEECH_TEXT(0)})
exten => s,n,Verbose(1,Confidence was ${SPEECH_SCORE(0)})
```

The sample dial plan creates a speech object and allows a caller to say "Yes" or "No" at the beep. The recognition will be in the form of a Boolean value "true" or "false." The results can be viewed from the Asterisk console with verbosity set to at least 1.

Tuning

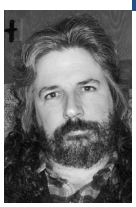
With ASR, tuning becomes essential to providing an intuitive tool for callers to engage in, rather than having to press "0" all the time to reach an operator. Unlike traditional nonspeech input methods (e.g., touch tone), what a caller may say is unpredictable. Tuning is the key to ensure reliability and improve accuracy on an ASR system:

- Dial-plan Tuning—The IVR prompt may have instructed the user to say "Main Menu" to return to the main menu. Yet if the caller says "Go Back," the dial plan may need to incorporate that as a valid command and take appropriate action (e.g., "Go one level up, except if already in Main Menu").
- Speech Database Tuning—Using the "Go Back" example, it is possible for the speech engine to fail in correctly identifying the phrase. In that case, the Windows-based Tuner can be used to correct the misinterpreted text. Using the Tuner, corrections can be made to the engine's database so that the spoken language will be interpreted as "Go Back" rather than "Go Bat." This will require that the original audio be recorded and stored for analysis—a tunable feature in the engine itself.

Some of the worse ASR-capable systems out there are the ones that have both touch-tone and ASR, especially the ones with long explanations about how to use the system followed by a large number of touch-tone options. It makes a caller yearn to press "0" and get a real person on the phone, defeating the purpose of an ASR system.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek who enjoys surfing (the Internet), sashimi (it makes great bait), and long walks (to the coffee machine and back).

rgferrell@gmail.com

THE TIME HAS COME, THE WALRUS said, to address the widespread and wholly erroneous notion that UNIX is immune from malware and therefore needs no vetting for the detection and removal thereof. Although it's true that certain other operating systems (which shall remain nameless but which rhyme with "ten toes") are far more attractive targets for exploitation and therefore receive the lion's share of attention from malware authors, "security through lesser market penetration" is hardly what one might term a robust information protection strategy.

If at this point, dear reader, you're expecting me to take you on a myopic retrospective through the history of UNIX viruses, trojans, and worms (oh my), you've once again failed to reckon with my formidable aversion to doing actual research. Looking at the big picture, I've come to the conclusion that at least part of the reason UNIX hasn't suffered more at the pale fingers of the bad guys is the habit of antivirus companies and the media of labeling newly discovered malware with downright silly names. Lupper, Scalper, and Slapper spring to mind, as do Pooper, Gasser, Ripper, and Stupor. Well, I probably owe those last few examples to a late-evening peanut butter and roasted habanero sandwich rather than any actual virus alert, but you get my drift.

Think about it: Who really wants to spend long hours slaving over a hot keyboard only to have the resultant glittering black pearl of slithery digital evil referred to by some pansy malware-tracking site as the "Foofer" worm? Indignity of indignities. Sure, "Win32/DEL.100907.ZZA" isn't exactly a coolly ominous moniker, either, but at least it gives the impression that someone's taking your work semi-seriously, providing a light at the end of the carpal tunnel, as it were.

Most modern malware contains two broad functional elements: the operational component, which takes care of infection, propagation, sanitization, amortization, and defenestration, and the payload (where the peanuts and nougat are found), also known as the business end. Even though it is the operational code that exhibits the real innovations in areas such as antiviral avoidance, stealth, firewall-dodging, polymorphism, consumer confidence index, and so on, the payload is where the

rubber meets the road so far as your hapless data is concerned. Payloads range from whimsical taunting ("You've been pwned!!!!") to the downright vicious (rm -rf *).

Pretty much everyone and their sentient canine has heard of the major classes of malware (i.e., viruses, trojans, spyware, worms, eels, and hagfish); the attack mechanisms of the most famous and widespread are well documented. There are many thousands of lesser pathogens out there, however, the chewy cream centers of which remain shrouded in palate-adhering gooey mystery. As a public service I've prepared a handy pocket reference to a few of these chigger bites in the picnic of computing, crafted with the same careful attention to detail and accuracy as is your local commuter train schedule. Antihistamines and anesthetic available on request from the front desk.

#*$!: Reverses "copy" and "delete" hotkeys; disables "undo."

Bunion: Hobbles snmpwalk.

Compost: Forces every running process to dump core; disables garbage collection; resides entirely in the heap.

Creosote: Downloads and installs bloatware until all disks are full, then crash dumps at the first user command input.

LiteSabr: Publishes any video files it finds on the system to YouTube—all of them.

Lumbago: Causes disks to spin out of control until they slip and rupture.

Nronn: Multiplies all arithmetic operations by 10.

Odie: Reboots whenever a user types "cat."

QRN: Replaces all semaphores with Morse code.

Qu'vatlh: Translates every text file on the system into Klingon.

Reverse Engineer: Plays all your .mp3, .ogg, and .wav files backward, then replaces them with a scratchy recording of "The Wabash Cannonball" (in one particularly virulent variant, on zither).

Roid: Drops the network connection and fills system logs until they push painfully against their partitions.

SafeSeks: Loads itself into the kernel, puts network interface into promiscuous mode, then hits random adult sites; will not unload until it encounters a trojan.

Scooter: Mails /etc/hosts to everyone in $HOME/.addressbook.

Upgr8: Renames /usr/bin/cd to /usr/bin/dvd.

Windozer: Disguises itself as a useful application but halts the system when invoked; at next reboot it blames the user, modifies the interface and application icon slightly, and repeats the process.

Xtinkt: Deletes awk.

ZZZZ: Deletes all rc files and replaces them with `perl -e 'sleep'`.

Don't scratch; it just makes it worse.

book reviews

ELIZABETH ZWICKY, SAM STOVER, AND MING CHOW

THE PRACTICE OF SYSTEM AND NETWORK ADMINISTRATION, 2ND EDITION

Thomas A. Limoncelli, Christine J. Hogan, and Strata R. Chalup

Addison-Wesley, 2007. 938 pages.

ISBN 978-0-321-49266-1

The first edition of this book was also the first book that talked about system administration conceptually, not focusing on theory or nitty-gritty instructions, but delving into information about how system administrators—serious, practicing system administrators, not any old bozo who happened to be saddled with the title—think about the operating system–independent issues involved in the job. It was a ground-breaking book and rightfully generated a lot of excitement, which makes it a hard act to follow.

However, the first edition was by no means a perfect book. The second edition isn't either, but it's a nice improvement. This is the book you need if you're looking for help in being a system administrator: not knowing what command to run, but knowing where to start in dealing with the people and machines around you. I don't agree with everything in it, I would have ordered it differently, and I really wish it had been better edited. But these are all quibbles. The second edition covers more ground, and covers it better, than the first edition.

RELEASE IT! DESIGN AND DEPLOY PRODUCTION-READY SOFTWARE

Michael T. Nygard

Pragmatic Bookshelf, 2007. 333 pages.

ISBN 978-0-9787392-1-8

This is a great book with a bad title and worse back-cover copy. I thought it was going to be about shipping software, a vague concept based on the title and strengthened by a back cover that starts, "Congratulations! Your project is finally finished and ready to ship . . . or is it?" In the terms I'm accustomed to, this book is about building services, not software. Software comes in a box, or on a CD, or you download it; it is shipped. Services you connect to; they are deployed. Yes, the title gets it right, but I was confused, nonetheless, since both get released.

Anyway, this is the book to consult if somebody bops up to you and says, "So, welcome to our new project. In three months, this site is going to go live on the Internet, and when anything goes wrong, your pager is going to go off." In case you don't know, there are two likely outcomes when the site goes live: Either nothing happens, or the site melts down. If nothing happens, either eventually the site melts down, or very, very bad financial things happen and it quietly disappears. You will note that all of these likely outcomes are bad. (They are also not mutually exclusive; probably the most likely outcome involves melting down immediately, melting down again later, and then disappearing eventually.) If you have a pager, your primary goal is to reduce the number of times the site melts down. Your secondary goal is to reduce the amount of time it takes to get the site out of meltdown.

This book will help you with both. It covers the most common mistakes people make in trying to build scalable, reliable sites and describes the ways in which you can add in tolerance, scalability, and manageability. It does so in the unmistakable voice of somebody who has been woken up in the middle of the night to save an ailing computer more times than can be counted.

THE ART OF SOFTWARE SECURITY ASSESSMENT

Mark Dowd, John McDonald, and Justin Schuh

Addison-Wesley, 2006. 1200 pages.

ISBN-10: 0321444426; ISBN-13: 978-0321444424

REVIEWED BY SAM STOVER

In the 1997 movie *Event Horizon,* Laurence Fishburne's character utters the line, "This place is a tomb." I had a very similar tone in my voice as I said, "This book is a tome" when I first laid eyes on *The Art of Software Security Assessment*. This book weighs in at a whopping 1174 pages (including the index); you *know* you're reading a book when you have this juggernaut in your hands. But even more impressive than the size is the content. This book has more value per page than a lot of other vulnerability books I've read, and that is saying something.

The first thing I noticed was that the majority of this book is over my head. Not being well versed in C, I had to struggle through the code to follow what was going on. That said, I found the chapter on Memory Corruption very readable. The chapters on Network Protocols, Firewalls, and Network Application Protocols (14–16, respectively) were especially interesting. Not only do the authors discuss different protocols in-depth, but they also highlight security issues within the protocols. For example, they provide a very complete explanation of how different operating systems treat overlapping IP fragments. This is not a new security flaw, but being walked through the protocol at a low level will definitely give the budding vulnerability researcher something to learn from.

The book is divided into three sections: Introduction to Software Security Assessment, Software Vulnerabilities, and Software Vulnerabilities in Practice. Each section builds on the previous, as the book is designed to be read cover to cover. However, as is always the case, you can feel free to jump around to your areas of need. In the first chapter, you'll read about fundamentals and terms to be used throughout the book. After that, Chapters 2–4 walk through the Design, Operational, and Application Review processes. Once you have a firm grip on how software development flows, the Software Vulnerabilities section jumps into all the problems that can arise during each of the development stages. Tons of sample code and great explanations abound, for both UNIX and Windows environments.

The final section deals with network and application vulnerabilities, and this is where I could really dig in. Understanding how the protocols worked made it a lot easier for me to understand the code samples, but there is still much I don't know, and this book is a great source to learn from. I was continually amazed at how much I learned by reading the individual sections, and yet I'm excited that there is still so much more to discover. I've read and reviewed too many books that left me wanting more. When I walk away from this book wanting more, I'll go tackle something simple like brain surgery.

In summation, I have no complaints at all about this book. Not a single one. I found it challenging, but accessible, and intimidating, but educational. What more could I ask for? I think you'll find that once you tackle this book, many of the other vulnerability development books will fall by the wayside. With this one, you won't need much else—which is good, because if I have to carry any other books besides this tome, I'll have to buy a heftier backpack.

EXPLOITING ONLINE GAMES: CHEATING MASSIVELY DISTRIBUTED SYSTEMS

Greg Hoglund and Gary McGraw

Addison-Wesley, 2007. 384 pp.

ISBN-10: 0132271915; ISBN-13: 978-0132271912

REVIEWED BY MING CHOW

The computer and video gaming industry is a multibillion-dollar-per-year industry. Since its inception, one problem has plagued both players and the industry: cheating. The spawn of massively multiplayer online role-playing games (MMORPGs) has introduced sophisticated cheating techniques, affecting the fun of games and also the bottom line of game companies. Greg Hoglund and Gary McGraw's latest book, *Exploiting Online Games: Cheating Massively Distributed Systems,* discusses how to cheat and break online games, how to develop some cool gaming hacks, and privacy and legal issues surrounding all the activities.

I was pleasantly surprised at the depth of the book. Many "dark side" topics of game development from modding to building bots were included. Before reading this book, I was only aware of one of the more publicized issues affecting MMORPGs: the sale of virtual goods for real money. There is a chapter in the book dedicated to the issues of money, virtual economies, and criminal activities. There are various technical techniques to exploit a game, including hacking the game's client and user interface, manipulating memory, modifying a game's state, and even using a debugger. The first half of the book (first five chapters) discusses basic issues in online games of which all players and game developers must be aware. The second half of the book (Chapters 6 to 10) discusses hacking various aspects of a game.

The first part of the book provided insightful overviews of (1) why players cheat, (2) relatively low-tech and high-tech ways of cheating in games, and (3) the gaming business and how far companies have gone to prevent piracy of their software. One of the most chilling parts of the book was presented in the second chapter, where the World of Warcraft's Warden, software to combat cheaters but comparable to spyware, was dissected. Hoglund presented a program he wrote called "The Governor" to identify the activities of Warden, including the reading of all open programs, processes, window names, and even memory locations on a player's machine. The book details new ways of cheating in online games, including gold duplication, traveling and respawning via taking advantage of the game client's bugs, aim and combat bots for first-person shooter games, and bots for online

poker games. Legal issues with game hacking were also presented, and several popular end user license agreements were examined. To understand the second part of the book, knowledge of C/C++ and assembly language is absolutely necessary. Hoglund and McGraw delved into hacking game clients, reverse engineering, and bot building. Hacking the game client is not just about controlling the game's user interface; it also involves manipulating graphical rendering information and injecting new code into the client via DLL injection. The details on reversing were rather basic, focusing on searching for strings and identifying assembly code patterns. Not everything in the book pertained to attacking online games in the black hat sense: there was also a chapter on what gamers know as modding, creating characters and new maps. The book concluded by stressing the need for security in the design of online games and urging players to pay attention to these issues with online games.

Most of the examples presented in the book were confined to arguably the most popular MMORPG of our time, World of Warcraft. I also found that the book constantly referred to two previous books in the Addison-Wesley Software Security series: *Exploiting Software: How to Break Code* and *Software Security: Building Security In*. You are almost required to read them before delving into this one, considering the number of references to the software security touchpoints, reversing, and debugging, which are covered more deeply in the previous two books. However, if you have read them already, then this is an outstanding book to apply everything learned to a practical topic.

This is a seminal book. Computer and video gaming is an integral part of our socioeconomics and culture, and the industry is booming at an alarming rate. It is important for both gamers and developers to understand that there are serious security, privacy, and legal pitfalls in online games. The problems are very real, and Hoglund and McGraw did a great job conveying the message that online games cannot be deployed or played naively.

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/.

ACCESS TO PAPERS from USENIX conferences online: www.usenix.org/publications/library/proceedings/

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

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.

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

Communicate directly with the USENIX Board of Directors by writing to board@usenix.org.

PRESIDENT
Michael B. Jones, *mike@usenix.org*

VICE PRESIDENT
Clem Cole, *clem@usenix.org*

SECRETARY
Alva Couch, *alva@usenix.org*

TREASURER
Theodore Ts'o, *ted@usenix.org*

DIRECTORS
Matt Blaze, *matt@usenix.org*

Rémy Evard, *remy@usenix.org*

Niels Provos, *niels@usenix.org*

Margo Seltzer, *margo@usenix.org*

2008 ELECTION OF THE USENIX BOARD OF DIRECTORS

ELLIE YOUNG
USENIX Executive Director

The biennial election for officers and directors of the Association will be held in the spring of 2008. A report from the Nominating Committee will be emailed to USENIX members and posted to the USENIX Web site in December 2007 and will be published in the February 2008 issue of *;login:*.

Nominations from the membership are open until January 9, 2008. To nominate an individual, send a written statement of nomination signed by at least five (5) members in good standing,or five separately signed nominations for the same person, to the Executive Director at the Association offices, to be received by noon PST, January 9, 2008. Please prepare a plain-text Candidate's Statement and send both the statement and a 600 dpi photograph to jel@usenix.org, to be included in the ballots.

Ballots will be mailed to all paid-up members in mid-February 2008. Ballots must be received in the USENIX offices by March 19, 2008. The results of the election will be announced on the USENIX Web site by April 1 and will be published in the June issue of *;login:*.

The Board consists of eight directors, four of whom are "at large." The others are the president, vice president, secretary, and treasurer. The balloting is preferential: those candidates with the largest numbers of votes are elected. Ties in elections for directors shall result in run-off elections, the results of which shall be determined by a majority of the votes cast. Newly elected directors will take office at the conclusion of the first regularly scheduled meeting following the election, or on July 1, 2008, whichever comes earlier.

THANKS TO OUR VOLUNTEERS

ELLIE YOUNG
USENIX Executive Director

As many of our members know, the success of USENIX 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 2007.

The program chairs of our 2007 conferences:
Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau, FAST '07

Ric Wheeler, 2007 Linux Storage & Filesystem Workshop

Brian Cooper and Nick Feamster, NetDB '07

Jeff Chase and Ira Cohen, SysML07

Niels Provos, HotBots '07

Hari Balakrishnan and Peter Druschel, NSDI '07

Galen Hunt, HotOS XI

Steven M. Bellovin, SRUTI '07

Jeff Chase and Srinivasan Seshan, 2007 USENIX Annual Technical Conference

Dan Boneh, Tal Garfinkel, and Dug Song, WOOT '07

Ray Martinez and David Wagner, EVT '07

Terry V. Benzel and George Kesidis, DETER 2007

Niels Provos, 16th USENIX Security Symposium

Betsey Nichols and Gunnar Peterson, MetriCon 2.0

Trent Jaeger, HotSec '07

The 2007 Linux Kernel Summit Committee members

Paul Anderson, LISA '07

Invited Talks/special track chairs:
Bill Aiello, Angelos Keromytis, and Gary McGraw, Invited Talks Coordinators for USENIX Security '07

Rudi Van Drunen and Doug Hughes, Invited Talks Coordinators for LISA '07

Philip Kizer, Guru Is In Coordinator for LISA '07

Adam Moskowitz, Hit the Ground Running Track Coordinator for LISA '07

Lee Damon, Workshops Coordinator for LISA '07

Brent Hoon Kang, WiPs and Posters Coordinator for LISA '07

Some other major contributors:
Balachander Krishnamurthy for his continued efforts in obtaining sponsorships and providing guidance for SRUTI

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

Avi Rubin, Dan Wallach, and ACCURATE for helping organize the Electronic Voting Technology Workshop

Dan Geer for steering the Workshop on Security Metrics

Peter Honeyman for his efforts in outreach to the international community, e.g., the SANE and Middleware conferences

Michael B. Jones for serving as liaison to the Computing Research Association

Matt Blaze, Clem Cole, Alva Couch, Rémy Evard, Michael B. Jones, Niels Provos, Margo Seltzer, and Theodore Ts'o for their service on the USENIX Board of Directors in 2007

Mike Jones and Dan Geer for serving on the USENIX Nominating Committee

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

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

Mark Burgess, Richard Chycoski, Esther Filderman, Cat Okita, and Dustin Puryear for serving on the SAGE Awards Committee

USACO UPDATE

ROB KOLSTAD
USACO Head Coach

USENIX is the premier sponsor of the USA Computing Olympiad, one of six prominent international Olympiads for pre-college students, a set that also includes Mathematics, Physics, Chemistry, Biology, and Astronomy. (Others, such as Computational Linguistics, Geography, and Philosophy are as yet in their infancy.)

The USA Computing Olympiad has four major goals: (1) to provide pre-college students around the world with opportunities to sharpen their computer programming skills to enable them to compete successfully at the international level; (2) to enhance the quality of pre-college computer education by providing students and teachers with challenging problems, training materials, and competitions that emphasize algorithm development and problem-solving skills; (3) to recognize those students with outstanding skills in computer science and encourage them to pursue further opportunities in the profession; and (4) to provide educational, motivational, and competitive materials in the form of programming competitions and Web-based training to students in the United States and over 75 other countries.

Most of these goals one might guess a high-school level computing organization might provide. Probably the most interesting point is the international flavor of the USACO. We, almost alone in the world, believe that the Internet's ubiquity enables us to share our ideas and work with students throughout the globe rather than just those in our own country. This shows up very clearly in our contest results. (See Figure 1.) U.S. students comprise only 21.2% of participants. It is worth noting that this is also a consequence of U.S. students' current lack of interest in pursuing careers in computer science.

Note the participation of Iran (29 contestants) and even Cuba (2 contestants). China, of course, is the big pow-

erhouse. They win four gold medals at the international competition almost every year. In countries like China and Poland, becoming a computer programming expert is often the ticket to a very successful career; they are less concerned than U.S. students about the job market.

FIGURE 1. Participation by Country

225	USA	10	ZAF	2	CUB
199	CHN	9	LTU	2	GRC
78	ROM	8	MEX	2	KOR
50	BLR	7	BRA	2	MDA
47	BGR	7	SVK	2	POR
37	CAN	7	YUG	2	SYR
33	GEO	6	ARM	2	THA
29	IRN	6	FRA	1	AZE
29	POL	5	LVA	1	BIH
24	IDN	5	SVN	1	CHE
24	VNM	5	TKM	1	CZE
19	UKR	4	ARE	1	FIN
17	EGY	4	ARG	1	HKG
17	TWN	4	ESP	1	MAC
16	IND	4	NED	1	MAR
15	DEU	4	SGP	1	MUS
14	TUR	3	AUT	1	MYS
13	HRV	3	EST	1	PAK
12	KAZ	3	MNG	1	SWE
12	RUS	3	PHL	1	TTO
11	BGD	2	AUS		

Online contests are offered monthly through the school year, culminating in the U.S. Open in April. Students compete for recognition (on the competition results), satisfaction, and to join the USA Invitational Computing Olympiad (more on that later). We also offer guided training (http://train.usaco.org), forcing students to complete tasks before moving on to the next section. This eliminates the observed problem of students skipping the more challenging tasks. Supplemented by a set of chapters on algorithmic programming, the 97 exercises provide perhaps 200 hours of online training. Another set of 300+ tasks is in final preparation for release as an enhanced training activity.

Perhaps surprisingly, the USACO does not train those who have no background in programming. Rectifying this omission is high on my priority list, but it will probably require a different set of volunteers.

The USA Invitational Computing Olympiad is a primary motivator for the 50 or so U.S. students who compete in the Gold division. About 15 students are invited to a college campus for a week of competitions, training, and fellowship. The 22 hours of competition determine the four members of the USA international traveling team.

Lately, the USACO has sent students only to the main international competition, the UNESCO-fostered International Olympiad on Informatics. This year's IOI was held in Zagreb, Croatia. I presented a summary of the USACO at the mini-conference there and will be in charge of submission grading for 2008's IOI in Egypt.

Two of our competitors, John Pardon and Matt McCutchen, earned gold medals, with John placing fifth in the world and Matt placing ninth. Ye Wang, exchange student from China, earned a silver medal, and junior David Benjamin (with two more years of eligibility) won a bronze medal. John has matriculated at Princeton; Matt is at the University of Virginia. Matt points out that "USACO training enabled me to skip the algorithmic part of undergraduate school and move straight into graduate courses during my first year." Matt is a talented programmer who also works in the filesystem code of the Linux kernel, testing new ideas on file ownership and access control.

All of these activities require a tremendous amount of manpower. Happily, former competitors assist Director Don Piele (emeritus professor at U. Wisconsin—Parkside) and myself as Head Coach. These top-notch coaches including Brian Dean, who recently finished his Ph.D. at MIT and is now a faculty member at Clemson. Percy Liang completed his Master's Degree at MIT and has journeyed to U.C. Berkeley for his Ph.D. Alex Schwendner joined us in Colorado after finishing his second year at MIT. Canada's 19-year-old Richard Peng created all the contests for the camp; he has returned for his second year at U. Waterloo. Another dozen associate coaches help keep the USACO contests running smoothly, including coaches at U. Waterloo in Canada and the University of Cape Town in South Africa. Many others contribute to creating and vetting the huge number of contest tasks the USACO consumes each year (54 tasks for the regular season; another 25 to 30 for the invitational camp).

We could not run the USACO without USENIX's sponsorship. Many thanks!

Ye Wang, David Benjamin, John Pardon, and Matt McCutchen show off their medals from the International Computing Olympiad in Zagreb, Croatia

conference reports

THANKS TO OUR SUMMARIZERS

16th USENIX Security Symposium

Boston, MA
August 6–10, 2007

KEYNOTE ADDRESS

■ ***How the iPod Shuffled the World as We Know It***
Steven Levy, Senior Editor and Columnist, Newsweek
Summarized by Kevin Butler (butler@cse.psu.edu)

Steven Levy used the shuffle feature of the Apple iPod as a metaphor for the digital age. Newspapers used to be the medium for getting news in one physical bundle, but today people can "shuffle" between news sites. Similarly, thanks to the Internet, shoppers can shuffle between stores, no longer constrained by what is in a mall. Levy postulated that even Apple did not realize the transformative nature of this facet of the iPod when it was first introduced. Despite the naysayers who considered Apple to be out of its depth dealing with consumer electronics, and despite the inauspicious timing of the product announcement, just a few weeks after 9/11, the iPod has become extremely popular, particularly since Apple stopped the strategy of tying the device only to the Mac and opened it up to the Windows market.

Levy said the random shuffle feature of the iPod didn't seem particularly random, with different people reporting to him how their devices seemed to play favorites. Although some would criticize the iPod for creating a "nation of zombies," Levy pointed to similar criticism mounted at the Sony Walkman when it was created. Competitors to the iPod may have better features (e.g., Microsoft's Zune can "squirt" songs between devices through their wireless interfaces), but the models are often botched by DRM that hampers the user experience in unacceptable ways. For example, the Zune will only allow playing of a song squirted to it three times or for three days, whichever is less.

Levy talked about the iPhone, described by Steve Jobs as the best iPod ever. In contrast to the low-key launch of the iPod, the iPhone's launch was a spectacle, and Jobs has been quoted as expecting it to sell 10 million units by the end of 2008. The iPhone, however, will not be as significant as the iPod, reasoned Levy, as the iPod was a once-in-a-lifetime device that symbolizes the digital age and represented our connection to our music. No later device will give the visceral charge users got from their iPods.

Bill Cheswick pointed out that nonrandomness in random generators was well known at AT&T Labs, given Ken Thompson's generator. The predecessor to

the iPod was developed at DEC research, which was subsequently bought by HP, but HP co-branded iPods with Apple, and even Carly Fiorina probably never realized that HP had the patents for many of these digital players. Niels Provos asked about convergence and interoperability between devices, and Levy answered that companies are too invested in rights management, preventing seamless interoperation. Consumers want a revolution, but given how much money they pay to companies such as cable and telephone for service, there will likely not be any revolutionary changes until high-speed Internet connectivity brings streaming content directly to end users.

WWW SECURITY

Summarized by Patrick Traynor (traynor@cse.psu.edu)

- ***SIF: Enforcing Confidentiality and Integrity in Web Applications***

  Stephen Chong, K. Vikram, and Andrew C. Myers, Cornell University

According to a recently released report by Symantec, Web applications account for more than two-thirds of all Internet vulnerabilities. These vulnerabilities are largely caused by inappropriate information flows within applications. To address this problem, Stephen Chong and his co-authors proposed the construction of the Servlet Information Flow (SIF) framework. Built using the Java Information Flow (Jif) language, SIF ensures at compile time that Web applications provide and respect confidentiality and integrity constraints. More specifically, SIF ensures that expressive policies dictating the flow of information can be enforced before, during, and after it is handled by an application.

Although the compile-time checks of the Jif programming language provide a number of benefits, they are insufficient to support highly dynamic operations. For instance, a user may want to specify policy during runtime. To address this concern, the authors significantly extended the language such that applications can dynamically delegate authority to "principals." By allowing an application to dynamically constrain the privilege associated with each specific instance, SIF provides significantly increased assurance over previous application frameworks. To demonstrate the utility of SIF, the authors created two sample applications. The first, a cross-domain information sharing tool, uses a mail-like interface. Information is subjected to a mandatory review before crossing domains. The second application, an online calendar, employs dynamic policies to limit a user's view of events. As situations evolve, users can appropriately change their information flow policies.

A number of attendees inquired about the difficulty associated with programming in Jif. Others were curious about information flow policy false positives generated during the compilation process. Whereas the first is certainly nontrivial, Stephen told the audience that the second was largely not a problem. Finally, in terms of performance, SIF causes no noticeable degradation to the system; however, a full set of microbenchmarks was not created.

- ***Combating Click Fraud via Premium Clicks***

  Ari Juels, RSA Laboratories; Sid Stamm, Indiana University, Bloomington; Markus Jakobsson, Indiana University, Bloomington, and RavenWhite Inc.

Advertisers have long had problems determining whether publishers actually deliver content to a targeted audience or simply claim their fees without doing the agreed work. As advertisers move significant portions of their efforts to the Internet, such fraudulent behavior is increasingly affecting the industry. Specifically, click fraud, or dishonestly reporting the frequency with which consumers are viewing advertising, has become the best-known vehicle of advertisement embezzlement. Unchecked, such behavior could not only be used to dishonestly gain revenue but also to maliciously drain the advertising budget of rival companies.

Recognizing the magnitude of this problem, Ari Juels and his co-authors propose a change to the online advertisement revenue model. Instead of attempting to filter out "bad" clicks, this group of researchers proposes that "good" clicks be used as the basis of advertising reimbursement. Such a technique need not be the result of a change from the "pay-per-click" to a "pay-per-click-creating-a-sale" model. Clients instead receive tokens in response to performing beneficial operations, such as making purchases at an online store. As clients with tokens visit a Web site, their visit is billed at a higher rate because they are more likely to be a real customer (as opposed to a bot performing millions of clicks). Although such a solution certainly would work for individual domains, Ari discussed how cross-domain use of tokens and user privacy (whether the token contains a list of items purchased or a single "premium" bit) presents challenges for future work.

One of the participants asked about how tokens fundamentally differed from cookies, which can also be used to track user behavior across Web sites if set by third parties. Another member of the audience questioned the benefit to the consumer of willfully providing such tokens to every site they visit. Ari responded that in addition to reducing the impact of fraud on prices, customers might be offered discounts to provide such information.

- ***SpyProxy: Execution-based Detection of Malicious Web Content***

  Alexander Moshchuk, Tanya Bragin, Damien Deville, Steven D. Gribble, and Henry M. Levy, University of Washington

Malicious Web content is one of the most serious security threats facing systems. As the escalating arms race between adversaries and providers shrinks the window between exploit and patching, new solutions must be provided to protect commodity computing devices. In response, Alex Moshchuk and his co-authors have created SpyProxy, a fil-

tering mechanism sitting between the Web and a user's browser. SpyProxy uses a virtual machine (VM) to sandbox and pre-executes incoming traffic in order to determine whether or not it is malicious. Should such traffic be benign, it is allowed to pass to the user. Malicious code, however, is not allowed past the sandbox.

The determination of whether incoming traffic is malicious relies upon three simple but effective signals. Because normal Web traffic should not cause new processes to spawn, suspicious files to be created, and the registry to be modified, the presence of any such change is used as the indicator of maliciousness. To test their hypothesis, the authors gathered 100 sites known to host malicious content (browser exploits and spontaneous downloads) and ran them through SpyProxy. Whereas commercial services such as McAfee SiteAdvisor were only able to block 80% of these pages from loading, the SpyProxy approach prevented all 100 pages from infecting the client's browser. In spite of its success, Alex noted that there were a number of challenges facing SpyProxy. For instance, a naive implementation introduces significant delays to the rendering of benign pages in the user's browser. Accordingly, incremental rendering was implemented to make the presence of a virtual machine transparent. All 100 pages analyzed by this work also contained largely static content. However, a significant portion of dynamic content, such as advertisements, can be made deterministic to the system through the use of caching.

One attendee wanted to know how encrypted traffic over SSL and streaming content would be handled. Alex acknowledged that both issues presented challenges and would be investigated in future work. Other audience members were curious about the false-positive rate observed during the group's experiments. Alex noted that false positives were extremely low (4 out of 2000 known safe pages) and were caused by the download of browser plug-ins. Such problems could be minimized by installing the most popular plug-ins into the VMs.

INVITED TALK

■ *The Human Factor in Online Fraud*

Markus Jakobsson, Indiana University

Summarized by Adrienne Felt (felt@virginia.edu)

Markus Jakobsson spoke about the importance of considering both human and technical factors when designing for security. He said that the perfect technical solution is not enough and that people cannot be treated like machines. Realistic experiments need to be conducted. Public education is required to successfully combat online fraud, with an emphasis on understanding why certain things are dangerous; for example, people who know not to click on links in email may instead copy and paste the link into their browser, because they do not understand the reason for the warning.

Jakobsson cited improper configuration, neglect, and deceit as three major human factors that lead to tricked and compromised users. Improper configuration includes issues such as weak passwords on access points, and neglect occurs when trusted companies use bad practices. "Spear phishing" is a highly sophisticated form of deceit that combines traditional phishing methods with data mining. He called these "context-aware attacks," in which a phishing email could use the correct bank or mail service name for a given user. These powerful attacks are spurred by the increasing amount of information available in public databases and social networking sites. Alternately, they may take advantage of a technical security hole such as the use of CSS to access a browser's history.

He promoted user experiments as a way to gauge the effectiveness of fraud prevention techniques and predict future trends. He said that they should not be done in the lab, because users have artificially heightened awareness when they know they are being tested. As an example of commonly misunderstood user behavior, he explained that users tend to look for negative signs of danger but not be alarmed by the lack of positive signs of security. This renders commonly used positive reinforcement techniques, such as small lock logos, ineffectual in many cases. Nonlab experiments are difficult to do because of ethical concerns and the lack of debriefing, but Jakobsson asserted that they can be done.

To address the need for public education, Jakobsson said that users need to be taught a basic understanding of security principles. He suggested the avoidance of specific examples (which often cannot be sufficiently generalized) or bullet lists. He runs a Web site, www.securitycartoon.com, that provides security-themed cartoons that are intended to teach accessible security lessons. The site http://www.human-factor.org contains his annotated talk slides and links to specific studies he mentioned during his talk.

PRIVACY

Summarized by Kevin Butler (butler@cse.psu.edu)

■ *Language Identification of Encrypted VoIP Traffic: Alejandro y Roberto or Alice and Bob?*

Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson, Johns Hopkins University

Charles Wright presented his work on determining information about anonymous calls made through VoIP networks. Although anonymous VoIP calls encrypt the data associated with voice communication, it is possible to infer who is speaking. This "perfect storm" of factors includes the use of variable bit-rate (VBR) codecs and length-preserving stream ciphers; these are good for compressing information but allow information leakage because of the different-sized packets found with VBR, which carry information about the original waveform. Additionally, by look-

ing at a spectrogram, it is easy to see different energy levels associated with frequencies.

By looking at speech data from a corpus of 21 different languages, Wright and his co-authors found that each language had its own fingerprint that could be compared against. By using *n*-gram probability distributions and chi-squared tests, the group was able to determine the language a caller was using with up to 40% accuracy. Arabic, the most difficult language tested, was detected with 30% accuracy after three tests. With binary classification, determining whether the caller was speaking English vs. Farsi was 98% correct, whereas determining Czech vs. Hungarian (the most difficult binary classification) was about 86% correct. Wright discussed countermeasures, which involved the need to decouple packet sizes from contents. With extra padding, the accuracy of determining the language can be reduced at the cost of extra overhead, making this less beneficial for VBR schemes. For a quick fix, a user may as well use constant bit rate (CBR) schemes and use the padding for improved sound quality; this will mean that all packets are large, however. There is a tension between efficiency and privacy, and an ongoing research question is how much more information is really leaked.

Nikita Borosov (UIUC) asked whether use of multilayer classifiers that had individual strengths was possible. Wright said that it was and that even a decision-tree approach could be used. Perry Metzger asked how many speakers were in the original corpus and, when doing extraction, whether elements of the original data set were recognized. Wright replied that between 70 and 100 speakers were in the corpus and that cross-validation was done, leaving one user out for statistics generation and testing against that user for validation. It was noted that patterns exist in these languages. Another question was about non-native speakers; Wright conceded that they may throw off the distributions. Jeff Donnelley (LBNL) pointed out that the tradeoff of bandwidth versus privacy was discouraging, and he wondered whether there are other parameters such as jitter or latency that could be traded off. Wright said this hadn't been thought of but could be nice. Lucas Ballard (Johns Hopkins) mentioned that jitter will not affect accuracy of this method. Paul van Oorschot (Carleton) asked about dialects within a language. Wright wasn't sure how it would play out but said it would be fun to look at.

- ***Devices That Tell on You: Privacy Trends in Consumer Ubiquitous Computing***

  T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno, University of Washington

Scott Saponas said that privacy was important, but people are often unaware of how much information they are leaking . For example, most people have no idea that their employers can read their online information. Similarly, loyalty cards allow grocers to get information about what we buy, but that information is also being used for other things, such as in lawsuits and for divorce cases. New technology in particular has unknown privacy issues, but there is plenty of potential information to be mined from us.

Saponas and his group looked at the Nike + iPod sport kit, the Microsoft Zune, and the Slingbox to determine what information was being leaked. The Nike kit lets the user see their speed and whether workout goals are being met. GPS is not used and, according to Nike, movement cannot be tracked. However, each device has a unique identifier that is often left in the shoe. Saponas et al. wired sensors around their campus and built a Google Maps application that allowed them to determine the whereabouts of people with the device in their shoes. The Microsoft Zune also had privacy issues, since user Bob can "squirt" content to user Alice. Although Alice can block Bob, the blocking feature is only based on the MAC identifier, so after Bob is blocked, he can turn off his Zune, spoof his MAC identifier, and send more unwanted content. The majority of their investigation, however, was into the Slingbox, a streaming-media server that uses VBR encoding and transmits the difference between frames rather than the frame itself. Saponas et al. grabbed 26 movies and captured the encrypted packets they streamed from the Slingbox to gain a fingerprint of the movie. By querying a 10-minute window, they were able to identify a movie with 62% accuracy; looking at 40 minutes allowed identification of the content with 77% accuracy. All the streaming was done in a lab environment, so it is unclear how well it would work over longer distances or over the Internet, but this information could be a useful way of determining whether movies are being transferred online.

One questioner asked about the range that was possible with the Nike + iPod kit. Saponas answered that detection could be done from about 40 feet away, and the shoe broadcasts about every second. By contrast to an RFID, the shoe contains an active sensor. Paul van Oorschot (Carleton) asked about the first two attacks presented and weak versus strong identifiers—is this privacy versus denial of service? Saponas relied that this may be a tradeoff between privacy and performance, and sometimes privacy vs. user experience (e.g., the sensor in a shoe could change its identifier every time, but it becomes difficult to differentiate the user from the recipient, making tracking progress problematic). Marcus DeShaun asked why the ID device in the Nike shoe was so strong. Saponas answered that it is persistent and tied to the device from the time of manufacture. Paul van Oorschot asked about the confusion matrix presented in the talk. Saponas replied that this is useful for giving hints about correlation.

- ***Web-Based Inference Detection***

  Jessica Staddon and Philippe Golle, Palo Alto Research Center; Bryce Zimny, University of Waterloo

Philippe Golle discussed the inference problem, giving the example that medical databases combined with voter registration databases can expose lots of information about a person. For example, the military set up a Web site to

make a public archive, but it did not realize that documents existed on the site that, according to experts, showed previously undisclosed ways to make nuclear bombs. AOL published a series of half a million anonymous queries, but it was easy to correlate these by the anonymous user ID used. As a result, it was possible to determine the entire personal life of someone through these queries, and the *New York Times* was able to track an individual user down by looking at this information. Inference detection is a general method of identifying potentially sensitive knowledge that may be derived when new information is combined with reference information. The closest related work is manual expert review of information to redact, as is performed in industries such as healthcare and litigation; however, these processes are expensive and error-prone.

Golle suggested using the Web to proactively detect inferences. The Web provides an up-to-date corpus of public information. It can act as a proxy for reference knowledge and allows for efficient automatic inference detection. By combining inferences and extracting keywords from documents, queries may be issued to a search engine and results determined. For example, most pages that reference the terms *Saudi, magnate*, and *sibling* also contain *Bin Laden*, so if these are found in a document, with a high probability Osama bin Laden is the inferred subject of this document. The algorithm presented was data-intensive but simple and effective, with no formal representation or NLP required. The Web is imperfect for reference knowledge, as the less famous people are, the less information appears about them on the Web. Also, whereas inferences may be detected by co-occurrences, it does not explain some of them (e.g., Madonna is correlated with "gay": although she herself is not gay, she has a large gay audience). These techniques could be used to find the identity of anonymous bloggers and also to find sensitive inferences between keywords and topic. For example, the words "transmit" and "infected" often inferred "HIV"; "transmit" and "mucous" together were an inference for "herpes." A potential use of this scheme is for redaction of documents to be declassified. There are billions of pages of data that need to be redacted but the rules are extremely long and complex for what to release; this line of inference of sensitivity could be very valuable as a declassification mechanism. Should there be a privacy firewall for redacted information? Individuals are aware of privacy needs but are not good at protecting their privacy, and this trend will likely only continue as information gets mined more and more while sources of information leakage continue to accrue.

One questioner asked whether redacting certain information may lead to an observer seeing that this information is gone and wondering what is being hidden. Golle replied that this was a good question and an area of research. Another questioner said that this was just inference detection but not control. Corman Deley (Microsoft Research) pointed out that the three keywords for Osama were found because he is famous and the information about him is well known; what about regular people? Golle answered that some information is more easily inferable than other information. Only the top three results were looked at before the USENIX deadline, but more terms were also looked at and indices created for them, and this yielded valuable new information. Additionally, companies may create internal indices and use these for redaction to not let people know that they are being looked at. In response to the question of what to do about this inference and whether it would be better to drown the signal emitted with high-quality disinformation (a potential anti-Google), Golle said that this was one way to go and that some companies specialize in removing a user's Web presence.

INVITED TALK

■ *Windows Vista Content Protection*

Peter Gutmann, University of Auckland, New Zealand
Summarized by Stefan Kelm (stefan.kelm@secorvo.de)

After being introduced as a "self-proclaimed hippy," Peter began one of his jam-packed, highly entertaining, high-speed talks about the new content protection "features" as introduced by Windows Vista. It all started with a posting made by Peter in late 2006 to the crypto list, which soon ended up in discussions on slashdot as well. The posting was about "A Cost Analysis of Windows Vista Content Protection" and slightly stirred up organizations such as Microsoft.

To summarize the contents of both the paper and his talk: Windows Vista's content protection mechanism introduces an SSL-like end-to-end encryption of all content paths within the PC. The software is protected by OS mechanisms such that, in theory, if there's any break at all the content quality gets degraded by the system. In effect, it looks like an attempt to turn a general-purpose PC into a sealed audio/video jukebox.

Peter described at great length the many problems with this approach. He mainly did so by reading the available specifications and interpreting them into general language.

The main issue is that general functionality gets disabled by Vista once anything looks suspicious. Because this affects not only commercial HD content blocking but the user's own content as well, this could lead to "premium silence." A key issue is that of driver problems, especially when using signed drivers. Already, Microsoft had (for no apparent reason) revoked a driver signature, which effectively led to denial-of-service via driver revocation.

Peter mentioned a number of other odd things in the spec, such as the so-called tilt bits ("you have to be insane to actually implement this stuff"), which need to be set by any device that detects anything unusual, whatever that means. Of course, this all leads to increased hardware and software costs as well as additional CPU consumption, espe-

cially because of all the (128-bit AES) encryption going on. Laptops, for example, cannot enter power-saving mode when content protection is active. However, one might argue about Peter's claim that Windows Vista is thus causing global warming . . .

In his closing thoughts he tried to answer the question, "Why did they do it?" The intent, Peter said, was not to protect HD content. In fact, if there was any threat model at all it was pretty badly done. The content protection "features" were likely being added because of requirements driven by lawyers.

One attendee claimed that what Peter described might be regarded as generally good since Microsoft is "raising the bar." Peter disagreed, stating that, rather than raising the bar, the company is annoying consumers, encouraging them to buy cheap PCs in order to view their own content. Another attendee asked for advice on how to build his own home cinema. Peter's recommendation was to not go near a Windows PC, but buy one of those cheap Asian-made media players using component video instead of HDMI and the like. Finally, Peter replied to another question that this'll never stop commercial pirates, since they "can just walk around it."

AUTHENTICATION

- ***Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks***

  Saar Drimer and Steven J. Murdoch, Computer Laboratory, University of Cambridge

  Awarded Best Student Paper!

  Summarized by Patrick Traynor (traynor@cse.psu.edu)

The average American consumer is unfamiliar with "Chip & Pin" readers, but customers throughout the rest of the world recognize these portable credit card readers as the product of a concerted effort to thwart fraud. Instead of allowing a waiter or waitress to make arbitrary charges to a card, t smart cards combined with PIN technology strives to force customers to become active participants in all transactions. Although potentially promising, such systems suffer a number of significant vulnerabilities. Most critically, Chip & Pin systems provide no means for a customer to verify device authenticity or correctness.

Saar Drimer demonstrated this point in as direct a manner as possible: by converting a seemingly normal Chip & Pin system into a Tetris-playing handheld device. Having demonstrated the ease with which such a device could be subverted and reprogrammed, Saar discussed a more critical threat. Specifically, card and PIN information can potentially be transmitted to a remote location and used by an adversary. Current systems attempt to bound such interactions by setting an upper bound on transaction length, but the allotted time of such interactions is sufficiently large to allow remote use of a card. To prevent such an attack from happening, the authors presented the first implementation of distance bounding protocol. Using the Hancke-Kuhn protocol, the authors sought to bound the latency between challenge and response to within a reasonable physical distance between card and reader. Because signals cannot travel faster than the speed of light, such a protocol can drastically reduce the area in which an adversary could launch such an attack (from around the world to within a room).

A number of attendees wondered whether the fraud caused by such attacks could more easily be tracked if customers were simply to retain all of their receipts. Saar agreed that such an approach would certainly make recognizing such incidents easier, but that in practice it would be difficult to enforce such actions.

- ***Human-Seeded Attacks and Exploiting Hot-Spots in Graphical Passwords***

  Julie Thorpe and P.C. van Oorschot, Carleton University

The security of passwords seems to be constantly in peril. Because most users are unable to remember strings that are both long and pseudo-random, accounts are "protected" by short and/or dictionary-word passwords. In response to this problem, new schemes including graphical passwords have been created. Passwords are represented as sets of "points of interest" selected by the user. By entering the correct points in the correct order, a client can authenticate to the system.

Although such systems are in many ways more usable than traditional password schemes, previous research did not investigate the security of this new approach. Specifically, because humans are known to pick distinctively nonrandom passwords, it is highly likely that their selection of points in an image will be similarly nonrandom. The authors performed two tests. In the first, 43 university students were asked to create click-based graphical passwords for 17 different images. In the second, 223 users created passwords on one of two images. These experiments were used to determine whether an adversary-seeded set of points could be used to reduce the password dictionary size. When using all of the user points from the first set of tests, the password program was able to guess 20% and 36%, respectively, of the passwords for each of the two images used in the larger study. Moreover, a significant number of those passwords (11% and 8%) were recovered in only three attempts. Julie then discussed the use of techniques from computer vision that automatically recover points of interest and thus do not require intervention by the adversary. This approach, while able to recover some passwords, will likely require additional tuning to increase its success rate.

One attendee wondered whether preventing users from clicking obvious hotspots via filtering would increase security. Julie agreed that such an approach would be possible, but potentially at the cost of usability. Others discussed the

discrepancies between lab and field tests and wondered whether the incentive of the lab participants was sufficient to create good passwords. Audience members were interested in whether the people picking good text passwords were also those picking more robust click-based passwords.

■ ***Halting Password Puzzles: Hard-to-break Encryption from Human-memorable Keys***

Xavier Boyen, Voltage Security, Inc.

A number of techniques have been used to dampen attacks on passwords. For instance, attacks on passwords that are only accessible through an online interface are easily detectable and often subjected to "three strikes" lockdown policies. Offline attacks are slowed by forcing adversaries to perform an expensive cryptographic operation a large number of times. The difficulty with this approach, however, is that this workload is client invariant. For instance, a user logging into an account from a low-power device (cell phone) will have to perform the same expensive operations. Moreover, a fixed number of iterations of a protocol fails to take into account advances in technology. Whereas 1,000 HMAC operations may be expensive today, technology may advance sufficiently in the coming years to reduce this burden on the attacker.

In response, Xavier Boyen argued that the number of iterations used by the transformation algorithm should not be fixed. Instead, the number of iterations "t" should be selected by the user and become part of the secret itself. An adversary with the correct password would not know whether it was correct without generating the results of all possible iterations. To execute this protocol, the user simply enters his or her password when prompted and then presses the start button. When the counter reaches the user's secret value of "t," the user presses the start button again. The transformed value of the password is then checked to determine correctness.

Several attendees noted that such a scheme was similar to those that use two passwords and asked why "t" was not simply appended or prepended to the password itself. Xavier noted that this approach allowed users to gauge their own security needs and tailor performance to their expected needs. Another member of the audience asked whether the selection of "t" could be included in password strength analysis applications. Although such an approach seemed possible to Xavier, he noted that he did not consider such a use in his work.

INVITED TALK

■ ***How to Obtain and Assert Composable Security***

Ran Canetti, IBM Research

Summarized by Kevin Butler (butler@cse.psu.edu)

Bill Aiello introduced Ran Canetti to the audience as one of the co-authors of HMAC, a contributor to IETF efforts such as IKE v2, TLS, and other protocols, and a co-chair of the multicast and cryptography groups within the IETF, but whose work has also garnered fundamental cryptographic results. Canetti has been instrumental in helping to formalize definitions of privacy and confidentiality.

Canetti began with a description of the millionaires problem, where two millionaires want to figure out which one lost more money in a stock crash without telling each other how much they lost. Each has a private input, which is then securely evaluated. This is an example of a cryptographic task, where two or more parties want to perform some joint computation while guaranteeing "security" versus "adversarial behavior." This has many applications such as secure communication, secure storage, e-commerce, and database security. Basic cryptographic building blocks have been established, such as key exchange, contract signing and fair exchange, coin tossing, zero knowledge, commitment, oblivious transfer, and secret sharing. Many cryptographic protocols have been developed over the years for obtaining authenticated and secure communication, with both general solutions and more efficient constructions for specific problems.

What does "security" mean? Some of the concerns include correctness of local outputs, and distributional and unpredictability guarantees. For example, in a gambling scenario, one wants to ensure that nobody knows the output beforehand, tallies are performed correctly, input is independent, and there is unbiased randomness in the output. Other desirable traits include the secrecy of local data and inputs, privacy, fairness, accountability, and availability against denial of service attacks. Rigorously capturing an intuitive notion of security is tricky. Security can only hold against computationally bounded adversaries and only in the probabilistic sense. There are unexpected interdependencies between security requirements and computational assumptions made (e.g., the difficulty of finding discrete logarithms in a finite field), and situations such as secrecy depend on correctness while correctness may depend on secrecy. Canetti's goal was to describe a paradigm for formulating definitions of security in a way that guarantees it in any given environment.

Canetti introduced a simple, insecure protocol combination, where each protocol was secure when run alone but when the protocols were run together they became completely insecure, because they used joint secret information in an uncoordinated way. This is a problem with the Needham-Schroeder-Lowe protocol, as, after authentication, the key N is XORed with the message it protects.

Now an attacker in the middle between A and B can intercept the message C, if it knows something about parties A and B. Thus, a message C = N XOR M, where M is the text to be delivered. Now if an attacker in the middle knows the message is either "buy" or "sell," that attacker can capture C and use B as an oracle to recover the key (because if

B sells, the attacker finds that C'=(N XOR "sell") XOR "sell" = N) to determine whether it has the right key. The weakness only comes into play in conjunction with another protocol that gives the adversary two possible candidates for the key; consequently, there is a need to explicitly incorporate the encryption protocol in the analysis of the key exchange protocol. Canetti also gave examples of the insufficiency of stand-alone security by showing the malleability of commitment protocols and weaknesses in the original zero-knowledge protocols, demonstrated through an auction example. He reiterated the need for a general framework for representing security concerns and requirements from protocols.

Universally composable (UC) security is a framework that allows any set of concerns and requirements for any cryptographic task, such as authenticity, secrecy, and anonymity. The basic paradigm is that a protocol is secure if it can emulate the ideal operating case, where two parties hand their information to a trusted third party (a criterion expressed by Goldreich, Micali, and Wigderson). This is an intuitively attractive definition, but it is difficult to formalize. Three necessary steps are formalizing the process of protocol execution in the presence of an adversary, formalizing the ideal process for realizing functionality, and formalizing the notion of a protocol emulating ideal operation. For example, in the ideal case for the millionaires problem, parties A and B would send their input x and y to a trusted third party, who would compute $b = x > y$ and send the result to A and B. Each party is then assured that its own output is correct based on the other's input, the inputs are independent, and its own input is secret except for the function value. Examples of ideal key exchange and commitment functionality were also given. Having these in place, it is possible to start with a protocol R that uses ideal calls to functionality F and to have a protocol P that securely realizes F. The protocol R^P can then be composed with each call to F replaced by an invocation of P, and each value returned from P treated as if it came from F. This allows for security complex environments and for modular system design and analysis, as decomposition into small protocols is possible and each can be analyzed individually. The security of the system composed of these protocols can then be deduced. Already there are many known protocols for secure communication primitives that are UC-secure, such as IKE, TLS, and NSL, and some notions are equivalent to traditional ones (e.g., digital signatures and CCA-secure encryption). Multiparty computation is also UC-secure if there is an honest majority; the case is more complex if only a minority is honest.

One problem with UC security is that it applies only to instances that do not share any local state, but this occurs often in reality. UC security with joint state is an attempt to solve this problem. Canetti reflected on this work by mentioning that although components were separately analyzed and the analysis only dealt with a single session, it was still possible to assert security for the entire system, with guarantees holding within any external context for the application. He concluded by briefly discussing pros and cons of some formal methods for security analysis and presenting further research topics such as making security analysis ubiquitous, finding better protocols that guarantee UC security, and potentially applying composition to other scenarios such as program obfuscation.

POSTER SESSION

Summarized by Charles V. Wright (cvwright@jhu.edu)

The poster session Wednesday evening featured posters on a wide range of interesting topics in security, from cryptography and authentication protocols, to botnets and network security, voting systems, and the business aspects of successful security products.

Adrienne Felt from the University of Virginia presented "Defacing Facebook: A Web 2.0 Case Study." Her poster described the system used by the social networking Web site Facebook to allow users to write and run custom applications, and how a single cross-site scripting vulnerability opens up the entire site to attack.

Alok Tongaonkar and R.C. Sekar from Stony Brook presented "Fast Packet Classification Using Condition Factorization." This work focuses on finding efficient ways to match packets to rules, such as in a firewall or Berkeley packet filter implementation. Using a novel technique for building rules into automata, they can achieve polynomial size and near-optimal runtime performance.

David Botta, Rodrigo Werlinger, Andre Gagne Konstantin Beznosov, Lee Iverson, Brian Fisher, and Sidney Fels, from the University of British Columbia, presented "Towards Understanding IT Security Professionals and Their Tools." They are working on learning about and understanding the human aspect of network operations and security, and how it affects the ad hoc communication systems that evolve among admins. The Web site for this study is http://hotadmin.org/.

Andrew Blaich, Qi Liao, Aaron Striegel, and Douglas Thain, from the University of Notre Dame, presented "Lockdown: Distributed Policy Analysis and Enforcement within the Enterprise Network." This work uses a wide array of tools—agents, visualization, Linux security modules—to construct a comprehensive framework for creating, verifying, and enforcing security policies throughout a modern enterprise network.

Chaochang Chiu and Yu-Ching Tung, from Yuan Ze University, and Chi-I Hsu, from Kainan University, presented "The Study of Identifying Critical Success Factors in Predicting PKI Implementation Success: A Bayesian Classifier Approach." They studied several public key infrastructure (PKI) products funded by the Taiwanese government and evaluated several different factors to determine what makes a successful product.

Chris Nunnery, Brent ByungHoon Kang, and Vikram Sharma, from the University of North Carolina at Charlotte, and Julian Grizzard, from the Johns Hopkins University Applied Physics Laboratory, presented a poster on "Locating Zombie Nodes and Botmasters in Decentralized Peer-to-Peer Botnets." They monitored P2P-based botnets and found that zombies are characterized by searching behavior, whereas botmasters are often the ones providing files that the zombies search for.

Janne Lindqvist, from the Helsinki University of Technology and the International Computer Science Institute, presented "Privacy-Preserving WLAN Access Point Discovery." His work, now in the exploratory stage, focuses on finding a way to achieve both ease of use and security in wireless network protocols.

George I. Davida and Jeremy A. Hansen, from the University of Wisconsin—Milwaukee, presented "A Four-Component Framework for Designing and Analyzing Cryptographic Hash Algorithms." By studying each of the constituent pieces of a typical hash function in isolation, they hope to build a component framework so that if one component (for example, the compression function) is found to be insecure or too slow, it can simply be removed and another plugged in to replace it.

Manigandan Radhakrishnan and Jon A. Solworth, from the University of Illinois at Chicago, presented a poster on "KernelSecNet," a framework for building secure applications. In this system, security-related functions such as authentication, authorization, audit, cryptography, and process creation are moved out of application programs and into an application proxy or even into the OS itself, to minimize the risk of vulnerabilities in application code.

Micha Moffie and David Kaeli, from Northeastern University, and Winnie Cheng, from the Massachusetts Institute of Technology, presented "TRACKS: A Behavior Based Policy System to Detect Hidden Malware." TRACKS is an intrusion detection system that analyzes program events to detect malware and enforce security policy. It learns malware behaviors from real trojan programs, and it detects attacks using binary analysis similar to taint detection.

Peter Williams, Radu Sion, and Erez Zadok, from Stony Brook, presented "NS3: Networked Secure Searchable Storage with Privacy and Correctness." The goal of this work is to develop techniques such as personal information retrieval, providing not only confidentiality for the stored data, but also privacy of the user's searches and query correctness.

Robert Beverly and Steven Bauer, from MIT, presented "Tracefilter: A Tool for Locating Network Source Address Validation Filters." They are conducting an ongoing measurement study on the Internet to determine where network administrators place firewall rules to filter out invalid source IPs. Usually, most filtering occurs at either the first or the second outbound hop.

Ryan Gardner, Sujata Garera, and Aviel D. Rubin, from Johns Hopkins, presented "Dynamically Establishing Trust: Can It Be Done Reliably?" Their poster deals with the difficult problem of using software on an electronic voting machine to authenticate itself to a human operator.

Thomas J. Holt, Lyudmila Leslie, Joshua Soles, and Brittany Spaulding, from the University of North Carolina at Charlotte, presented "Exploring the Social and Technical Aspects of Political Conflict On-Line." This work details the cyber attacks on Estonia, launched earlier this year apparently from sources in Russia. The poster gives a timeline for the attacks and Estonia's response, with analysis of the major events of the conflict.

Zhiyao Liang and Rakesh M. Verma, from the University of Houston, presented "Authentication Considering a Dishonest Insider." In this work, they seek to extend classical work on authentication protocols such as Needham-Schroder, to allow for a more malicious insider threat.

THREATS

Summarized by Charles V. Wright (cvwright@jhu.edu)

■ *Spamscatter: Characterizing Internet Scam Hosting Infrastructure*

David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego

Chris Fleizach presented a study of the Web server infrastructure used by spammers and other scammers to host their malicious Web sites. In this work, the authors sought to determine (i) how scams are distributed across servers, (ii) whether multiple scams might share servers, (iii) how long scam sites stay up, and (iv) where (geographically) these sites tend to be hosted.

First, the authors had to develop a method for determining to which particular scam each spam mail or fraudulent Web site belongs. For this, the authors developed a novel "image shingling" algorithm, whereby a screenshot of the scam page is taken, then broken up into several smaller blocks. The blocks, or "shingles," are then hashed, and the similarity of two Web pages can be computed as the fraction of shingles they share. This technique cleverly sidesteps the noisy data and ambiguity in the scam URLs, spam emails, and even the HTML content of the scam pages themselves.

The authors found that although scammers use many machines to send out spam messages, they typically use a much smaller number of Web servers to host the scam page itself. For example, one common scam used only three Web servers (one in Russia and two in China). Overall, it is more typical for the scam site to be hosted in the U.S. (over 60%), perhaps to be closer to the victims, to facilitate payment processing, or to cultivate the appearance of legitimacy. Spam relays, in contrast, are much more widely distributed around the world, with only 15% lo-

cated in the U.S. Most of the scams were not distributed across multiple servers, but a small number of sites were observed to use more than 20, and 40% of the scams were hosted on a server that they shared with at least one other scam.

A lively Q&A session followed the talk. Two early questions dealt with discrepancies between trends in the UCSD group's data and what audience members have observed in their own networks. Chris explained that spam changes constantly and can vary considerably between networks; for example, rapidly changing DNS names might be used by one group of spammers and not by another, and spam collected in other countries might differ considerably from that targeted at the U.S. Other questions concerned the nature of the spammers and scammers themselves: whether the UCSD group has made any effort to contact them personally; where the money from the scams is going; and whether the scammers could really be making much money from short-lived scams. Chris replied that, although they have not examined this aspect of spamming and scams yet, they plan to delve deeper and learn more about this "underground economy" in future work.

■ ***Exploiting Network Structure for Proactive Spam Mitigation***

Shobha Venkataraman, Carnegie Mellon University; Subhabrata Sen, Oliver Spatscheck, and Patrick Haffner, AT&T Research; Dawn Song, Carnegie Mellon University

Shoba Venkataraman presented a measurement study of the IP addresses that send mostly spam and those that send mostly legitimate mail. Based on this study, she proposed a new solution to help overloaded mail servers prioritize legitimate messages. Because existing spam-filtering techniques use content-based analysis, the filtering happens fairly late (after the mail has been accepted for delivery) and can be computationally expensive. In contrast, by filtering based on the sender's IP address, the new scheme is computationally efficient and can weed out most spam before it can even be transmitted.

To show that such a technique would be effective at stopping spam with only a minimal impact on nonspam mail, the authors examined traffic logs from a busy mail server. They calculated the "spam ratio" (i.e., the fraction of all mail from an IP that is flagged as spam) for each IP address that sent mail to its server and found that most IPs had either very low or very high ratios. Moreover, IP addresses that are present in the logs on most days tend to send a lot of legitimate mail and very little spam. Most of the spam comes from transient IP addresses, which show up only rarely in the logs, but the authors found that these spammer IPs tend to cluster together in blocks of address space and that the spam-sending blocks tend to be long-lived. (In fact, 90% of them persist for longer than 60 days.) Therefore, when the receiving mail server is very busy, it can prioritize connections from IP addresses from which it has recently received mostly nonspam mail, and it can drop connections from IPs in the recent spam blocks.

There were several thoughtful questions after Shoba's talk. One audience member asked whether spam really does tend to overload mail servers, preventing the delivery of nonspam mail (a behavior he had personally never observed); another asked about the potential effects of deploying the proposed system in different places around the world. Shoba replied that spam behavior can vary from network to network, so the blocks of IP addresses flagged as spammers might be specific to the location of the defender's network. Likewise, the intensity of spam varies, and it does in fact sometimes prevent delivery of legitimate mail on the network where the authors collected their data. There were other questions on whether the clusters of "bad" IPs identified here are correlated with hijacked address space, and how the results of the current study compare to earlier work by Feamster et al. Shoba explained that although the earlier work did not consider legitimate mail and that for the current study they didn't consider hijacked address space, the new results agree with Feamster et al.'s findings about the sources of spam traffic.

■ ***BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation***

Guofei Gu, Georgia Institute of Technology; Phillip Porras, Vinod Yegneswaran, and Martin Fong, SRI International; Wenke Lee, Georgia Institute of Technology

Guofei Gu presented a new approach for detecting botnet infections in a local area network. The system, called BotHunter and available for download on the Web at http://www.cyber-ta.org/BotHunter/, examines two-way communication patterns between hosts in the local network and those external to it, and it detects successful bot intrusions by correlating intrusion alarms on inbound traffic with scanning patterns in subsequent outbound connections. The authors defined a five-step model of botnet propagation: inbound scanning, exploit, download of bot code, communication with the botnet command & control, and outbound scanning. They use this model in BotHunter to improve the accuracy of the network intrusion detection system over what is possible when looking for individual bot-like behaviors in isolation. For example, Guofei explained that inbound infection attempts are not enough evidence of an infection to warrant raising an alarm, because zombie machines on the Internet are continuously scanning for victims, and most of their attempts are unsuccessful. However, when the IDS also sees scanning or other bot-like behavior from machines in the local network, it's much more likely that there really is a problem.

BotHunter is built on the popular open-source Snort intrusion detection system. In addition to using many of the standard Snort filters, the authors also developed two new extension modules: the Statistical sCan Anomaly Detection Engine (SCADE) for detecting inbound and outbound scanning and the Statistical payLoad Anomaly Detection Engine (SLADE), which performs *n*-gram analysis on packet payloads to detect malware. SLADE compares fa-

vorably to the earlier network intrusion detection system PAYL, offering more fine-grained *n*-gram pattern matching, better runtime performance, and improved detection rates. The authors evaluated BotHunter's detection ability in a virtual network in a lab, in a honeynet, and finally in live network settings at Georgia Tech and SRI, finding that it produced good detection rates and a low number of false positives.

The first of several questions dealt with detecting bots that do no inbound scanning, propagating instead via trojan horse, email, etc. Guofei explained that BotHunter doesn't always have to observe all five steps of infection in order to raise an alarm; the model could even be tailored to different types of spreading behavior. Another question concerned whether an attacker might try to avoid detection by going very slowly, so that BotHunter might fail to correlate an inbound scan with an outbound scan happening much later. Guofei replied that, yes, in fact all schemes based on time windowing are vulnerable to such attacks, but the defender could randomize the window size to make the attack more difficult.

INVITED TALK

■ *Exploiting Online Games*

Gary McGraw, Cigital

Summarized by T. Scott Saponas (ssaponas@cs.washington.edu)

Gary McGraw began by suggesting that among the eight million users of online multiplayer games there are bound to be many dishonest or malicious users. In this talk, McGraw focused on how and why some of these users cheat in online games. He said he chose online games because they are a bellwether of things to come.

Online games include the trinity of trouble: connectivity, complexity, and extensibility. Originally, these games only had an online component to help prevent piracy of games; however, connecting to many other users online is now an essential part of many games. Blizzard's World of Warcraft (WoW), for example, has 500,000 simultaneous users on six continents and represents a large distributed system with a fat client pushing the limits of computing systems.

One of the major reasons for cheating is that there is big money to be made. Some users do not have the time or desire to earn achievements in the game, such as collecting gold (virtual wealth) or attaining a certain level. As a result, there is a world market for virtual gold where people will pay real money through online auctions and intermediaries. McGraw cited that in some places, such as China, one can make more money playing video games and selling virtual wealth than through traditional jobs (http://youtube.com/watch?v=ho5Yxe6UVv4).

Many people cheat at WoW by manipulating game state stored locally in memory. They look through memory and find where information such as their virtual location or how much virtual gold they have is located and change this information. Modifying such state allows cheats such as teleporting and duplicating gold. These hacks get distributed on the Internet as unauthorized add-ons to the game.

Blizzard's first approach to thwarting this problem was to look all over one's computer for files related to cheating hacks. This Big Brother approach results in users getting banned when they have downloaded a cheat but not installed or used it. Blizzard also added the Warden software to WoW to watch for interference with the memory or execution of the game.

McGraw described how recently these defense mechanisms have been defeated with more sophisticated techniques attacking from the kernel or video card. From the kernel, it is possible to manipulate memory without the Warden being able to detect changes. From the video card, there have been hacks to change how the virtual 3D world is rendered to give users advantages over their enemies. For example, it is possible not to render walls or to render enemies as large orange objects.

McGraw concluded by suggesting that online games require a new model for software design and security in which distinguishing insiders from outsiders is less clear. For more information, see the book Gary McGraw has written with Greg Hoglund on this topic, *Exploiting Online Games*.

ANALYSIS

Summarized by Stefan Kelm (stefan.kelm@secorvo.de)

■ *Integrity Checking in Cryptographic File Systems with Constant Trusted Storage*

Alina Oprea and Michael K. Reiter, Carnegie Mellon University

Alina Oprea's talk on integrity protection in encrypting file systems was motivated by the many companies that nowadays outsource their data. There already are a number of different protocols and services available. However, do the users actually trust the remote servers?

Alina proposed an end-to-end security architecture in which the overall security is only maintained by the clients through the usage of so-called trusted key storage. In her work she has been looking at integrity in cryptographic file systems (CFS). These systems usually divide files into fixed-lengths blocks, with each block encrypted individually. The integrity of these file blocks needs to be protected by using a constant amount of trusted storage per file, usually at a size of only several hundred bytes.

She proposed two new integrity algorithms, RAND-EINT and COMP-EINT, which utilize two properties of many common file systems: (1) a low entropy of files, and (2) the sequentiality of file writes. After briefly discussing Merkle trees, Alina introduced the two new algorithms.

The entropy integrity algorithm (RAND-EINT) does not initially protect against replay attacks, thus counters are being introduced. The compression integrity algorithm (COMP-EINT), however, is based on Merkle trees but is much improved compared to previous approaches (including one by the author herself).

A main part of her work was to actually implement the algorithms in EncFS (which does not provide for integrity) and evaluate the algorithms in terms of performance. To do this, she described four metrics that were being used to test the performance of both algorithms and suggested which algorithm would be best for low-entropy files vs. high-entropy files, as well as read-only access vs. other access. Alina recommended implementing both new algorithms and letting the corresponding application choose which one to use based on its typical workload.

One question asked was whether the solutions proposed may be used for pipelining in network file systems. You can't do that for integrity, Alina replied, except for HMAC algorithms.

For more information see http://www.cs.cmu.edu/~alina/.

- ***Discoverer: Automatic Protocol Reverse Engineering from Network Traces***

  Weidong Cui, Microsoft Research; Jayanthkumar Kannan, University of California, Berkeley; Helen J. Wang, Microsoft Research

In this talk Weidong Cui described the development of a new tool called Discoverer. The motivation for this work was to automate the process of reverse engineering network protocols, a process that today is performed manually. The goal is to improve security techniques such as IDS/IPS, penetration testing, and generally identifying protocols. The challenges of automatically reverse-engineering these protocols are that they greatly differ from each other and that many message formats are fairly complex, consisting of text and binary fields.

Weidung described the key design goals of their tool as protocol-independence (since they did not want application-specific customization), correctness, and, primarily, automation such that no manual intervention is necessary when examining network flows. The basic idea is to infer protocol idioms, two of which he then described: the message format, usually considered as a sequence of fields, and their semantics (lengths, offsets, cookies, etc.). Moreover, most protocols use what Weidung calls format distinguisher (FD) fields to differentiate the format of subsequent message parts.

Discoverer's architecture can be divided into four phases: (1) during the tokenization phase, field boundaries are being identified; (2) during initial clustering, each message gets dissected into different tokens belonging to either the binary token class or the text token class; (3) during the recursive clustering phase, the initial clusters are being further divided, because initial clustering may be too coarse-grained; and (4) the merging phase merges clusters of the same format by using type-based (instead of byte-based) sequence alignment.

He then went on to describe the prototype they've implemented and the evaluation conducted. The following metrics were of importance during the evaluation: correctness, conciseness, and coverage. Two binary protocols (CIFS/SMB and RPC), as well as one text protocol (HTTP), were evaluated with Discoverer, with network flows having been collected in an enterprise network. Weidung concluded that more than 90% of their inferred messages actually correspond to "real" network flows (as obtained from Ethereal).

One questioner asked about the actual goal of this work. Weidung reiterated that this is all about automatically reverse-engineering network flows. On how resistant this scheme would be against an adversary garbling the messages, Weidung replied that you can't do anything at all about encryption or obfuscation of messages.

For more information, see http://research.microsoft.com/~helenw/pub.html.

- ***Towards Automatic Discovery of Deviations in Binary Implementations with Applications to Error Detection and Fingerprint Generation***

  David Brumley, Juan Caballero, Zhenkai Liang, James Newsome, and Dawn Song, Carnegie Mellon University

  Awarded Best Paper!

The final talk of this session was on detecting differences in how multiple implementations handle the same protocol. Zhenkai Liang began by explaining that many different implementations usually exist for the same protocol but that two implementations often do not interpret the same input alike, often as a result of implementation errors or because the implementer chose to implement only a subset of the protocol.

In their work, the authors propose a new approach to detecting those deviations for error detection or fingerprint generation. This approach is based on behavior-related deviations instead of checking minor output details as provided by any application. The main problem they face is the question, "How do I find input in order to demonstrate that two implementations behave differently?"

The key concept of their work is use of symbolic formulas. Their approach is an iterative one, consisting of three phases: (1) during the formula extraction phase, x86 instructions of a particular implementation are first transformed into an intermediate language, which in turn is transformed into symbolic formulas; (2) the deviation detection phase constructs queries from those formulas which are sent to application servers using different inputs; (3) finally, the validation phase checks whether or not two different implementations really reach different protocol states.

Zhenkai then discussed their prototype implementation. They evaluated their approach on NTP and HTTP, testing Ntpd, NetTime, Apache, Miniweb, and Savant. He concluded that their prototype already is pretty fast in detecting deviations, but that they need to improve the prototype in order to be able to explore different program paths that might lead to the very same protocol state.

One attendee wanted to know how this approach would be superior to static analysis of applications. Zhenkai answered that, with static analysis, one might not know where the program will jump to.

For more information see http://www.andrew.cmu.edu/user/liangzk/.

INVITED TALK

Computer Security in a Large Enterprise

Jerry Brady, Morgan Stanley

Summarized by T. Scott Saponas (ssaponas@cs.washington.edu)

Jerry Brady's invited talk discussed the present challenges for managing computer security at a multinational financial organization. He began by explaining that the security priorities of a large organization vary by location. In some countries, a company is concerned about employee safety or physical theft. However, in the United States, his company is primarily concerned about electronic security.

Electronic security is made challenging by a diversity of technology platforms and processes, as well as global operational requirements such as "no off hours" and limited patch windows. Brady said that upgrading a firewall with an important patch can take as long as a year because of the planning required to identify what will be impacted by the change and the downtime.

Brady identified risk management as the main complicating factor in computer security. Ultimately, risk management has to be the main priority of an organization such as his. However, it is often at odds with security. For example, when there is a complex interdependency of applications and platforms, sometimes the least risky option is not to make an update or change to the system.

Security is a careful balancing act of risk, cost, regulatory obligations, priorities, and risk tolerance. Different business units within a firm have different needs and expect several options. As a result, security must be tiered and responsibility has to be distributed. One challenge of distributed responsibility is that it is not always clear who accepts the risk: the application, the business unit, or the entire company's brand.

Five years ago, exchanges among banks or law enforcement agencies about security incidents or intelligence about computer security threats did not happen or were not useful. Brady said that now when fraud or attacks happen everyone shares knowledge with everyone else so that everyone can fight these attacks.

Brady concluded his talk by discussing the challenge of finding and training good security staff. He said that the security field needs academia's support. The IT culture must be changed. He said we need to weave risk and security into the technology curriculum.

PANEL

Cellular Network Security

Panelists: Ron Buskey, Motorola; John Larson, Sprint Labs; Simon Mizikovsky, Alcatel-Lucent, Wireless Emergency Response Team; Hao Chen, University of California, Davis; Thomas La Porta, The Pennsylvania State University, Bell Labs Fellow; Patrick Traynor, The Pennsylvania State University

Summarized by William Enck (enck@cse.psu.edu)

The recent surge of academic interest in cellular network security has caught the attention of academic and industrial researchers alike. USENIX Security organizers put together a panel to bring together industry and academia to expose the community to the concerns and issues as seen by both sides. Radu Sion introduced the panelists and prompted each to give a short presentation.

Ron Buskey began addressing questions he was sent by the panel committee. Will telecom networks independent of the Internet continue to exist in 25 years? Will providers (wired or wireless) move their services to the general "public infrastructure" of the Internet, or will networks dedicated to voice traffic and its specific requirements remain? Ron said that technology has changed so fast that it is hard to foresee the future. However, the trend has been to "bring the Internet to the user." This means there will be more equipment closer to users, and we must be very careful of the security of these devices. With migration toward an IMS (IP Multimedia Subsystem) core, how can members of the security community become more active in the security of telecommunications networks? Ron believes a major issue will be the widespread adoption of SIP-based services. Specifically, the ability to authenticate and integrity must be clean and easy to manage. Finally, billing has always been, and will remain, important.

Patrick Traynor looked to the audience and boldly claimed that *they* are the next big threat to telecommunications security. The design of future systems will play an important role. Current designs rely heavily on traditional abstraction. For example, devices simply assume the network is there. However, in the cellular network, each action requires a significant amount of work, and a device not sensitive to the network reaction will unintentionally (or intentionally) cause disruption.

Patrick divided future problematic software into three classes: ported, buggy, and malicious. Ported software assumes the cellular network acts as a packet-switched net-

work. However, applications that have long periods of silence (e.g., Skype, IM, and SSH) require costly setup messages for every transaction. Buggy software will, for example, contain timing problems, and refresh connections too late, resulting in many superfluous connection setups. Finally, malicious software will act like buggy software, but will do so on purpose. Patrick concluded by stating that there are over two billion cellular users, whereas the Internet is estimated at only one billion. More sophisticated devices are beginning to penetrate the cellular user base, and systems design must proceed with care, or else there will be serious impacts on the core of the network.

John Larson focused his presentation on WiMAX security and 4G networks. There are many well-known wireless threats, including malware, intrusion, eavesdropping, DoS attacks, and rogue base stations. There are also less-known attacks such as protocol fuzzing that result in DoS or intrusion. Future networks must be sensitive to all of these issues. He would like to see a comprehensive security architecture, for which work is currently in progress. The architecture needs encryption at various layers; however, unnecessary redundancy is bad, as it will hurt performance. Devices need to be made more secure, and the network must have the ability to quarantine malicious devices. John also believes DoS is a crucial issue that is not really dealt with. His lab was able to knock over all the equipment it has tested. They need help from the research community to fix this problem in protocols and applications.

Thomas La Porta presented an overview of the evolution of 2G to 3G. Traditionally, the telecommunications networks were closed. Who knows how SS7 (an out-of-band telephone signaling protocol) works? It is hard to do damage to something if you don't know how it functions. However, this situation is changing. The first step in opening up the cellular network has been to convert the core to IP. The purpose is to attach services that are not directly attached to the Internet. This will result in many new avenues into the network. As the network evolves, a common IP core will be connected to an ANSI-41 core (older cellular network), a UMTS core (3G cellular network), and IP access. The network accessed by a client does not necessarily "own" that client, and it must rely on the owning network for authentication. Hence, a network is only as strong as the weakest network it is connected to!

Thomas foresees both single infrastructure and cross-infrastructure attacks that will result from modifying message data, modifying services logic, and denial of service. We have already seen with the Greece incident that such attacks can compromise provider equipment. Again, a network is only as strong as its weakest connected network. Problems will get worse, and we need to be proactive about finding problems and fixing them.

Simon Mizikovsky presented an overview of the 3GPP and 3GPP2 architecture evolution. The trend has been to push functionality toward the edges. This will result in an intelligent, concentrated base station. He foresees new base station routers (BSRs) with IP interfaces showing up in a bunch of different configurations (e.g., hanging on a wall, in a basement, or on a telephone pole). The BSR will be the gateway to an IP core and will handle security connections with client devices and will act as foreign agents in mobile IP. By moving functionality toward the user premises, the user must now be viewed as adversarial. How do we protect you from your neighbor? As the network transitions into this form, we need to build components so that they can be trusted, and ensure that when one component falls, the rest of the network remains.

Hao Chen reiterated previous concerns by asking, "Are cell phones friends or foes?" As more complex services and phones become available, there is a greater chance of malfeasance by end users. Traditionally, phones have been viewed as dumb terminals and not a significant threat. However, this is no longer the case. Hao reviewed a number of attacks that allow a user to circumvent billing and fairness constraints. He concluded that the network should no longer consider the cell phone a friend.

After the panelists completed their presentations, Radu kicked off the questions by asking why none of the panelists mentioned anything about privacy. Ron responded, agreeing that it is a problem. Devices are becoming more like PCs, which means they contain more private information. How to simultaneously protect a phone from a malicious network and the network from a malicious phone is a hard problem.

Conversation then moved toward equipment security. Radu asked whether network devices placed in town are more than simple PCs. John confirmed that many are PCs. From the audience, Perry Metzger noted that in New York City, components are stored in poorly locked rooms. Simon responded that such instances will go away because the equipment of today will go away. John added that new equipment is rolling out fast; however, physical access problems will remain.

In the next line of questioning, Radu asked the panelists whether there is concern with natural disasters and DoS. Thomas replied that there has been significant work and research looking at graceful degradation for overload. However, one of the biggest concerns is that a lot of equipment is outdoors, and components such as towers can be blown down. Simon commented that as the network becomes flatter, and network functionality moves toward the edge, with the number of nodes increasing, survivability increases as well. A flat network is more resilient. John added that systems are getting smaller and more portable. This makes it easier to rapidly replace infrastructure.

With all this talk of impending collapse of connecting networks, an audience member asked where everyone had been in the past 20 years. Have the telcos not been preparing for this? Are they preparing to protect the end user

from threats? John responded that the answer is defense in depth. Many measures need to be put in place to help the situation. Some parts have been deployed, but the solution is not there yet, and a better security architecture is required. When asked about lessons learned, Ron responded that the situation is similar to laptops and desktops five years ago, when users began installing applications from random origins. However, the situation is better for the telcos, because they do not have to deal with a legacy OS. However, the hackers have had 20+ years to learn better techniques. Protecting in both directions at the same time will be difficult.

Conversation changed as another audience member wondered how easy it would be to spoof caller ID/NAI in new networks. Simon responded that such data is passed in the clear in IP networks; however, 4G devices do not have unique identifiers other than their MAC addresses. John added that devices will have x.509 certificates; however, a challenge here is binding device and user authentication.

Radu then asked what skills students need that are not being taught and that the telcos would like to have. Ron responded that students need to have the ability to look at use cases and figure out attacks. Students can look at and understand requirements, but they are not good at looking for problems that don't follow protocols. Also, there is a lack of software people who know what is going on below in the hardware. The software/hardware boundary is where many attacks are occurring. John echoed Ron's comment and added that there is a need for much better software delivery systems. We should not be seeing buffer overflows in these devices. It is very difficult to find people with both security and telecom experience. There is a need for people who can build security-resiliant-protocols.

From the audience, Luke St. Clair asked whether there are proposed solutions for securing endpoint devices and whether any proposals consider secure hardware. Simon responded that some work has gone toward provably secure protocols. As for secure hardware, it depends on how much you want to spend for the platform. Hao commented that there is always the "our phone" solution, where users cannot install software. John mentioned that as the industry moves toward more open APIs, user access will increase. Today's EVDO PC cards do not allow the user to gain access to lower levels, but this access will be available in 4G cards. Thomas added that the network designer can never assume end devices are secure. You must assume it will misbehave. You must protect yourself. Patrick commented that defense in depth is always a good model. Thomas, referencing an earlier audience comment, replied that all locks are doing is buying you time. If you have a strong lock on your apartment, a burglar will pass by you and go to your neighbor; however, eventually that lock will be broken, just as people will break end devices.

Perry Metzger inquired about SS7. He said it is as vulnerable as you can imagine and the only thing standing in the way is "will." People do not know how it works, for now, but this will not always be the case. Thomas replied that it is too expensive to retrofit SS7 networks. If you want to keep the upper parts of SS7, you need to replace the lower levels. Even MAPSEC isn't that secure. Patrick prompted the industry panelists to discuss how widely deployed MAPSEC is. Simon responded that the standards were defined and that it was deployed in two networks, only to be removed because it was determined to be a performance bottleneck.

Next, an audience member asked how he can get involved in testing cellular security without going to jail. Patrick responded that it is getting easier as things move away from SS7 and toward IMS. It is possible to set up an IMS/SIP network for yourself. John added that he has a active security group looking to recruit good people.

INVITED TALK

Mobile Malware

Mikko Hypponen, F-Secure Corp.
Summarized by T. Scott Saponas (ssaponas@cs.washington.edu)

Mikko Hypponen opened his talk by saying, "With new mobile platforms we get new viral vectors." He said that many analysts originally projected that the first viruses for mobile platforms would be on Windows Mobile and be spread by email, because it would be very easy to take existing Windows viruses spread by email and retarget them for the mobile platform.

However, the first mobile viruses were on the Symbian platform and used a Bluetooth vector to spread. Hypponen confirmed the common view that mobile viruses are indeed real and are really spreading. There are more than 370 mobile phone viruses so far and tens of thousands of infections worldwide. One operator with more than nine million customers claims almost 5% of their MMS traffic is infected. Another large operator says 8,000 infected devices have sent more than 450,000 messages, with one mobile device sending 3,500 messages. Hypponen demonstrated some attacks during his presentation.

Hypponen described the prerequisites for a mobile malware outbreak: have enough functionality for malware to work, have enough connectivity for malware to spread, and have enough targets for the platform to become an interesting target. The first mobile virus appeared in 2004 and since then 370 new viruses, worms, trojans, and spy tools have been discovered. Hypponen said that so far they have not seen mobile rootkits, worms that do not need user interaction for spreading, mobile botnets, or any large-scale profit-oriented malware (professionals).

Currently, most malware is for Symbian, and no malware exists for Windows Mobile. Most are trojans (297), with some viruses (58) and a little spyware (9). These trojans

break phones so that they do not work, break a subset of services, cause monetary loss by sending viruses, steal users' private information, or delete email and other important information. This contrasts with desktop viruses, which in the past two years have not been focused on breaking machines because there is no money in breaking machines.

We know that some mobile malware has come from Norway, Spain, Brazil, and many countries in Southeast Asia, Hypponen said. However, the source of many viruses is still unknown. Most infections by this malware also occur in Europe or Southeast Asia. He also explained that Symbian has likely been targeted so much because it is common, SDKs are available, and much existing viral code exists on the Internet. Most malware is just a modification of an existing piece of malware.

Recently the first examples of mobile spyware have been seen. Mobile spyware can record text messages, call information, voice recordings, and even physical location. In fact, there are even vendors on the Internet who sell this software or phones already modified with this software. Although only Symbian-signed applications can have access to the functionality needed for spyware, some spyware has passed the Symbian signing process. For example, one piece of spyware passed as a "system backup" tool.

Hypponen concluded by saying that in the future we can expect to see more for-profit mobile malware and mobile botnets. F-Secure sells a mobile antivirus and firewall product that helps protect against many of these threats.

LOW LEVEL

■ *OSLO: Improving the Security of Trusted Computing*

Bernhard Kauer, Technische Universität Dresden
Summarized by Sarah Diesburg (diesburg@cs.fsu.edu)

Bernhard Kauer began by stating that the goal is to have a secure system with only a minimal trusted computing base (TCB). Kauer then explains that he has found bugs in trusted computing systems based on a static root of trust, such as Microsoft's Bitlocker. He gave a general overview of a Trusted Platform Module (TPM), described the PCR register (which is a 160-bit-wide register holding a SHA-1 hash), and discussed what makes a trust chain unfakeable.

During his security analysis, he examined three trusted bootloaders (LILO Darthmouth, GRUB IBM Japan, and GRUB Bochum) and found that all three have bugs that break the trust chain. He then talked about current attacks, including TPM resets and BIOS attacks. For example, in a certain BIOS attack all one has to do is flip just a single bit in the BIOS image to get a TPM with fresh PCR values.

Kauer then discussed using the dynamic root of trust feature instead, because it (1) shortens the trust change, (2) can minimize the TCB of applications, and (3) is less vulnerable to TPM and BIOS attacks. He then introduced the Open Secure LOader (OSLO), which features the first publicly available secure loader that is based on AMD's skinit instruction. It includes its own TPM v1.2 driver and is available at http://tudos.org/~kauer/oslo/. He explained that OSLO works by initializing the TPM, stopping other processes, executing skinit, hashing every module into a PCR, and starting the first module. Kauer also stated that the code size and binary size of OSLO are much smaller than those of BIOS and GRUB.

Kauer's future work includes incorporating memory type detection, DMA protection, and a port of OSLO to the Intel LT platform. He would also like to search for other attack points of Trusted Computing Systems. In conclusion, Kauer stated that OSLO is one step to a minimal TCB.

One questioner asked what he meant by removing the BIOS from the TCB. Kauer explained and also stated that it is an open problem to ensure that ACPI tables are secure. Another questioner asked why OSLO is so much faster than Trusted GRUB if they both incorporate SHA-1. Kauer invited the questioner to look at the code.

■ *Secretly Monopolizing the CPU Without Superuser Privileges*

Dan Tsafrir, The Hebrew University of Jerusalem and IBM T.J. Watson Research Center; Yoav Etsion and Dror G. Feitelson, The Hebrew University of Jerusalem

Dan Tsafrir introduced an attack dubbed "cheat" that can be easily launched by a nonprivileged user in order to get any desirable percentage of CPU cycles in a secretive manner. He explained that this means users can arrange things such that their application would get, for example, 95% of the CPU cycles, regardless of any other programs that may be running, and despite any OS fairness considerations. Further, Tsafrir explained that the cheating program would appear as consuming 0% CPU in system-monitoring tools such as top and ps, making the exploit very hard to detect. Tsafrir noted that arbitrary programs can be turned into cheaters through binary instrumentation or other means.

Tsafrir identified two unrelated mechanisms that independently make systems vulnerable to cheating. The first is time-keeping. Specifically, the OS measures time in "tick" units, where each tick is a few milliseconds. The way it works is that the OS wakes up every tick and charges the currently running program for using the CPU during the last tick. This happens even if the program actually used a small fraction of it. This sampling approach is fairly accurate if applications "play by the rules" (the more the program runs, the bigger the chances are of being billed), but Tsafrir showed how a program can "cheat" this mechanism by systematically sleeping when the OS wakes up. As a consequence, the OS erroneously thinks the cheating program consumes no CPU cycles at all, which grants it a very high priority, thereby allowing it to monop-

olize the CPU. Tsafrir stated that all examined "ticking" OSes were found vulnerable to the attack to some degree, including Linux, FreeBSD, Solaris, and Windows XP.

In contrast to the tick-based time-keeping mechanism that is used by OSes since the 1960s, the second exploitable mechanism identified by Tsafrir is much more recent: scheduling for multimedia. Tsafrir noted that, in an attempt to better support the ever-increasing CPU-intensive multimedia component within the desktop workload, some modern systems (Linux, FreeBSD, and Windows XP) have shifted to prioritizing processes based on their sleep frequency rather than duration. Tsafrir concluded that this major departure from the traditional general-purpose scheduler design plays straight into the hands of cheaters, which can easily emulate CPU-usage patterns that multimedia applications exhibit. To the question of how, then, OSes will be able to provide adequate services for multimedia applications, Tsafrir responded that an OS can favor one program over another only if it is reasonably sure that the former is much more important to the user, and that for this purpose, his research group has done a lot of work to explicitly track interactions with users (through the appropriate device driver) and to leverage this information for improved multimedia scheduling.

■ ***Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems***

Thomas Moscibroda and Onur Mutlu, Microsoft Research

Onur Mutlu began by introducing a new class of Denial of Service (DoS) attacks prevalent on multicore chips in which the cores share the same DRAM memory system. He explained that as different threads or processes execute on different cores, those threads or processes can interfere with other memory access requests. He noted that most scheduling policies are not even thread aware and are very thread "unfair." Mutlu then introduced the concept of a Memory Performance Hog (MPH), which is a thread that exploits unfairness in the DRAM controller to deny memory service (for long periods) to threads co-scheduled on the same chip.

Mutlu then demonstrated the problem by walking through a memory scenario caused by running the popular benchmarking program *stream*. Stream has a very high memory row-buffer locality. Unfortunately, because many memory systems implement a First-Ready First-Come First-Serve (FR-FCFS) scheduling algorithm, stream's requests will be serviced before other, older memory requests, to take advantage of the current row buffer. Mutlu says that this unfair memory request servicing can severely degrade other threads' performance and that it is easy to write an MPH.

DRAM fairness was then discussed. A DRAM system is fair if it slows down each thread equally. Mutlu defines the goal of a fair scheduling policy as equalizing DRAM slowdown *i* for all threads *i*. Two fair memory scheduling algorithms were discussed. A hardware implementation must keep track of experienced latency and estimate ideal latency.

After the presentation, Mutlu commented that this problem is probably even more severe in hyperthreading. He also noted that the stream benchmarking example is not even the worst case. An attacker could create much worse cases.

INVITED TALK

■ ***Computer Security and Voting***

David Dill, Stanford University

Summarized by Adrienne Felt (felt@virginia.edu)

David Dill spoke about the history and current state of trustworthy electronic voting in the United States. Dill became involved in the politics of electronic voting in 2002–2003 when he wrote the "Resolution on Electronic Voting" petitioning for user-verifiable voting. He challenged the audience to question the current system of blindly trusting voting machines and to contribute to the campaign for a voting audit trail.

The resolution calls for a voting trail that can be authorized by the user and then is indelible. The goal is not to discourage people from voting, and there is no proof of wide-scale voting theft. However, Dill argued that the absence of known vulnerabilities is insufficient; instead, it is the responsibility of the government and voting machine manufacturers to provide evidence that the election was correctly carried out. He said that the core security problem is the threat of internal attacks. Regardless of how well employees are screened, Dill asserted that it is morally wrong to force voters to trust strangers with their votes. Even if the technical issues are perfect, how can voters understand and trust this? Even if the hardware and software are secure, how do you verify that the correct equipment is in the box? Voting must be auditable and audited.

In August 2003, few people agreed with his position. Since then, however, public opinion has begun to change. Big blunders in security design were uncovered. Manual auditing is catching on, with 13 states now practicing this. The House of Representatives will soon be considering the Voter Confidence and Increased Accessibility Act. Dill credited the computer security community for this and thanked his colleagues who have taken the time to learn about e-voting and talk to politicians and the press.

Despite the progress that has been made, Dill stressed that their work is not yet complete. It is still necessary to emphasize that patching external attacks is insufficient. They need to go district to district and ensure that the election officials understand the relevant issues. There is currently no good legal recourse for the worst-case scenario of a broken election. The general public doesn't understand that security is a continuous spectrum and a continuing problem. There remains a need for members of the computer security community to join the trustworthy voting campaign; even if the current problems are fixed, new attacks

and attackers will develop in the future. For those who are interested in participating, more information is available at VerifiedVoting.org.

His talk generated numerous questions. Two audience members asked whether paper trails were more reliable than electronic ones, and Dill responded that paper-based voting also needs to be subjected to more scrutiny but that voter registration and verification are separate issues. There was also a discussion about how secure but complex technologies can lead to errors, and Dill emphasized the importance of good human-factor design for both voters and election volunteers. When asked about the plausibility and potential effectiveness of increasing election centralization and uniformity with a constitutional amendment, Dill replied that the federal government can only regulate the national elections and not the hundreds of local elections. Additionally, the federal government is slow and counties and states are easier to influence. The bottom-up approach is part of what makes the trustworthy e-voting campaign so resource-intensive.

OBFUSCATION

■ *Binary Obfuscation Using Signals*

Igor V. Popov, Saumya K. Debray, and Gregory R. Andrews, The University of Arizona

Summarized by William Enck (enck@cse.psu.edu)

Saumya Debray began by reviewing background material. The program compilation process strips the high-level code semantics from an application. Reverse engineering and disassembly can reestablish these semantics from a binary executable. Sometimes, a program's author wants to keep high-level code from prying eyes. Binary obfuscation provides such a mechanism. However, motivations ranging from protecting intellectual property to concealing malware make binary obfuscation a double-edged sword.

There are two classes of reverse engineering: static and dynamic. As the names indicate, static analysis does not execute code. It has many advantages, including complete code coverage and simpler algorithms; however, it cannot account for self-modifying code. Dynamic analysis, in contrast, accounts for self-modifying code, but it only disassembles the executed code. Additionally, it requires significantly more complex algorithms in order to track state. Furthermore, running code can take defensive measures to hide execution from debuggers. The goal of this work is to make static reverse engineering so impractical that the adversary is forced to perform dynamic analysis, at which point runtime defenses can be implemented.

Static reverse engineering primarily relies on control flow analysis, which involves both identification of control flow instructions and inference of resulting jump locations. In short, it must find where code branches and where it ends up. Hence, the authors hypothesize that hiding control flow instructions will make static analysis impractical. Saumya proposes two techniques to hide legitimate control flow instructions. First, all such instructions are converted to "ordinary instructions" that raise a signal when executed (e.g., a segmentation fault or division by zero). A special signal handler traps the execution and first checks the code location. If the instruction corresponds to a jump, execution jumps to the correct location; otherwise, the signal is propagated to standard handlers. Second, bogus control transfer instructions are inserted at unreachable locations, further confusing the static analysis tool. Saumya explained that this binary obfuscation technique makes static analysis both provably NP-hard (when pointer aliasing is introduced) and practically expensive. Finally, he showed that the technique is a very effective deterrent against popular disassemblers and incurs only a 21% slowdown.

David Wagner pointed out that the kernel traps signals, thereby allowing the kernel to observe control flow. Saumya indicated that this is dynamic analysis, and the goal of the work was to force the adversary to use dynamic analysis; other defensive runtime mechanisms can be used to thwart such attacks. Another audience member raised concern over the location of the mapping table used by the special signal table to determine if a signal is really a jump. Saumya replied that there are multiple ways to obfuscate the mapping table, for example, smearing it across the code; however, the current implementation uses a simpler method of XORing addresses.

■ *Active Hardware Metering for Intellectual Property Protection and Security*

Yousra M. Alkabani and Farinaz Koushanfar, Rice University

Farinaz Koushanfar explained hardware piracy. Chip fabrication facilities cost billions to build and maintain. Most hardware design firms can't afford such facilities and therefore outsource the fabrication process. Unfortunately, to do so, design firms must divulge their raw intellectual property (IP) and have no way to ensure the honesty of the fabrication facility. Hardware IP piracy has been estimated to cost one billion dollars per day. The solution is active hardware metering, a process that ensures that pirated fabricated chips are unusable. A number of passive hardware metering systems have been proposed; this is the first work to consider the active variety.

Active hardware metering aims to protect a design firm from a fabrication facility wishing to make extra copies to sell for itself. Note that this is a well-funded adversary, as it owns a multi-billion-dollar facility. Farinaz proposes an approach where fabricated integrated circuits (ICs) are initially locked, and the design firm must unlock each IC before it can be used. This is done by adding additional useless states into finite state machine (FSM) logic. The resulting Boosted FSM (BFSM) is seeded by a unique value for each fabricated chip. This value is derived from fabrication variability as shown by Su et al. in ISSCC'07. With high probability, the IC starts in an inoperable state, and

only the design firm has the ability to correctly transition the BSFM into the correct operational state. To further hinder adversarial analysis and protect against brute force attacks, the BSFM may also contain black-hole states that render the IC useless. Finally, Farinaz explained that FSM logic contributes only a minimal amount to overall chip area, and converting FSMs to BFSMs has negligible impact on chip size. Further analysis shows the same for power consumption and timing delays.

Adrian Mettler inquired whether the logic containing fabrication-specific identification circuitry could be removed from the IC netlist, thereby allowing simulation to derive BFSM logic and allowing the adversary to unlock chips. Farinaz replied that hardware simulation is nontrivial. A single simulation can commonly take over six months for today's microprocessors. Furthermore, as circuit component size metrics count atoms, variation is very important. The nondeterministic portions cannot be taken out of the design. Robert Cunningham expressed concern over the scalability of incorporating active metering. Farinaz replied that much of the process is automated, and the parts that needed to be hand-designed could be automated by incorporating the technique into existing hardware design tools.

INVITED TALK

Advanced Rootkits

Greg Hoglund, HBGary

Summarized by Sarah Diesburg (diesburg@cs.fsu.edu)

Greg Hoglund's talk covered funded rootkits, types of attackers, a new attack trend of desktop exploitation, classifications of rootkits, ways in which rootkits may be installed, goals of rootkits, and, finally, how to build, package, and install a rootkit. Although Hoglund's talk was targeted at Windows operating systems, the general concepts can be applied to all other major operating systems.

A funded rootkit is subversive malware developed with a budget. Hoglund reminded us that rootkit authors put rootkits they are making through an extensive testing and development process by testing them against all known antivirus software.Types of rootkit authors include competing corporations, foreign and multinational corporations, foreign governments, intelligence services of friendly and allied countries, former intelligence officers, extremist groups, and organized crime and drug cartels. Many organizations do not report rootkit exploitation, so the extent of this problem is relatively under-reported.

Desktop exploitation was discussed as the culmination of recent rootkit trends. It involves such desktop applications as Yahoo! Toolbar Helper, Microsoft Certificate Authority Control, Crystal Reports Control, and Quicktime. A parallel was drawn between desktop exploitation attacks and Internet attacks by comparing exposed system API calls to exposed TCP ports.

Hoglund felt that the current method of classifying rootkits based on technological mechanisms is too inflexible. He described multiple rootkit types and gave examples of each. These types include tool trojans and log cleaners, permanent installations of parasitic code into existing programs, basic backdoor programs, operating system trojans, dynamic parasitic infections of existing processes, parasitic application extensions, modifications of operating system library functions, parasitic device drivers, free code, memory cloaking rootkits, rootkits that hide from DMA, rootkits for embedded systems such as cell phones, rootkits lower in hardware such as hypervisors, boot vectors, and overflow activation.

Finally, methods of building and packaging rootkits were discussed and snippets of code were shown. These pieces of codes and methods are available at www.rootkit.com.

A questioner asked how Hoglund believes we should design security solutions in the future. Hoglund suggested that the problem might be solved through trusted computing and DRM in hardware-level support. Another person asked whether there will always be a never-ending supply of undiscovered rootkit techniques, and Hoglund believes there will be until hardware stops it. He also commented that the price of rootkits is rising because demand is rising.

NETWORK SECURITY

On Attack Causality in Internet-Connected Cellular Networks

Patrick Traynor, Patrick McDaniel, and Thomas La Porta, The Pennsylvania State University

Summarized by William Enck (enck@cse.psu.edu)

Patrick Traynor began by warning of the paradox of specialization. Optimizing a subset of functionality provides benefits under normal conditions, but it causes disasters when the environment changes. A canonical example is the Tacoma bridge collapse. Although the bridge was designed to withstand strong forces, it was only a 40-mph breeze that led to its collapse. We are seeing a similar phenomenon with today's telecommunications networks. They were designed for rigid constraints and use patterns; however, many assumptions no longer hold because of the incorporation of the Internet. We have seen a number of low-bandwidth attacks against cellular networks, and Patrick claims that adding more bandwidth will not solve the problem. He believes the problems are the result of a clash in design philosophies, that is, the connection of smart and dumb networks, which differ in conceptual definitions such as traffic flow.

To further justify his claim, Patrick introduced two new low-bandwidth attacks against cellular data networks. GPRS (the data service for GSM) sets up channels whenever a client requires communication. This is an expensive

operation which potentially requires multiple paging sequences to locate a device. To alleviate setup strain, devices maintain the channel for at least five seconds. However, there are a limited number of channels. GPRS specifications allow for up to 32 concurrent flows per sector, but many equipment manufacturers use less, for performance reasons. Exhausting this virtual channel resource is straightforward. An adversary need only ping 32 devices in a sector every five seconds to ensure all channels are occupied. Patrick used mathematical analysis and simulation to show that an adversary only requires 160 kbps of bandwidth to block 97% of legitimate traffic in Manhattan, a value dwarfed by the theoretical maximum capacity of 73 Mbps. A similar attack is shown to be effective on the PRACH (packet random access channel).

So what is the problem here? For phone calls, the channel technique makes sense; the setup costs are amortized by multiple-minute phone calls. Data communication, however, does not fit this mold, and many protocols (e.g., instant messaging) require only very small messages with significant delays between network use. Throwing more bandwidth at the problem does not reduce setup latencies. Patrick showed a simple throughput equation as a function of packets, setup latency, and bandwidth. He explained that as bandwidth is taken to infinity, the throughput is entirely reliant on setup latency. Hence, although the rigid design works well for phone calls, the choice of a circuit-switched network architecture provides a fundamental limitation because the cellular network is attached to the Internet, where end points do not care about what happens inside (as in the end-to-end argument).

In the question and answer session, Niels Provos agreed with the fundamental problem of specialization and inquired how the telcos are going to fix the problem. Patrick replied that he cannot speak for them directly, but technologies such as WiMAX are promising; however, 4G is in its infancy, and it might be a while before these networks are deployed. Another audience member asked how much of the current design is dictated by battery power and if this will prevent architectural changes. Patrick agreed that power limitation of phones has a large influence, but this will change as providers expand to target laptops and more versatile devices.

■ ***Proximity Breeds Danger: Emerging Threats in Metro-area Wireless Networks***

W.Y. Chin, Institute for Infocomm Research (I^2R), Singapore; V.T. Lam, University of California, San Diego; S. Sidiroglou, Columbia University; K.G. Anagnostakis, Institute for Infocomm Research (I^2R), Singapore

Summarized by William Enck (enck@cse.psu.edu)

Periklis Akritidis began by reminding the audience of the pervasiveness of wireless networks. According to wardriving data, most access points are left unprotected. Of the remaining protected access points, most use WEP, which has been broken many times over. In metropolitan areas, access points are literally on top of one another, and wireless clients can easily see multiple SSIDs. This work proposes a new theoretical worm, called a wildfire worm, which spreads by physical proximity to wireless access points.

The wildfire worm gets its name from the way it propagates. Once infected, a wireless host listens for other SSIDs and, if one is found, it attempts to infect hosts on that network. In turn, those hosts infect other hosts on adjacent networks, and so on. There are two techniques to propagate a wildfire worm: pull and push. The pull technique requires the victim host to download malicious code (from a Web site), which may be achieved via ARP or DNS poisoning. More interesting is push propagation. To better understand the propagation potential, public wardriving data from major metropolitan areas was acquired and analyzed, suggesting that in denser areas, a well-crafted worm can infect up to 80% of wireless hosts within 20 minutes.

The existence of such an out-of-band worm allows for a twist on a number of well-known attacks. Among these are packet sniffing, ARP and DNS spoofing, and phishing. Additionally, Periklis proposes tracknets, in which an attacker "rents" wireless networked zombie hosts to track specific users as they move throughout the city. All attacks are much more practical with the use of a wildfire worm, because the adversary is within the network. Furthermore, it is harder to extinguish, because it does not rely on the Internet to communicate. Fortunately, all is not lost. Periklis proposes a number of countermeasures, which when used in conjunction will limit the effectiveness of many attacks while allowing the wireless networks themselves to remain open.

Angelos Stavrou asked whether a fast-moving car could aid propagation speed. Periklis replied that this would be similar to seeding the attack in more locations. Another audience member inquired about wildfire worms applied to access points that separate an internal wireless network from an intentionally open and public wireless network. Periklis said that if the attack can bypass the access point, it can still occur. A final audience member noted that there is something useful and attractive about having an open access point environment and inquired if there is any way to salvage it. Periklis replied that the reactive and filtering techniques described in their paper will allow networks to remain open.

■ ***On Web Browsing Privacy in Anonymized NetFlows***

S.E. Coull, Johns Hopkins University; M.P. Collins, Carnegie Mellon University; C.V. Wright and F. Monrose, Johns Hopkins University; M.K. Reiter, Carnegie Mellon University

Scott Coull began by explaining that many research areas desire real network logs. Network administrators realize the benefit of such research and would like to provide logs; however, their primary function is to protect the anonymity of their users. For example, if many employees visit monster.com, a clueful investor may steer clear of that

company's stock. A number of anonymization techniques have been proposed, but how effective are they? If made public, anonymized network logs are available for long periods of time and are subject to the scrutiny of complex inference algorithms. Previous work has revealed a number of flaws in anonymization processes; however, Scott believes there is more to learn. Ultimately, the goal is to better understand what properties are necessary for secure anonymization and to provide guidance to publishers desiring to contribute their data.

This work specifically considers anonymized NetFlow data, which consists of a time-ordered sequence of records. Each record provides information about packets in a TCP connection between a server and a client. The goal of the adversary is to find a specific Web site. The general approach is to download the front page of the target Web site and compare the many possible flows to the data set. Previous work has considered purely flow analysis; however, as Scott indicated, objects often shift between flows, resulting in an overlap between Web sites. Hence, the cumulative size of the Web page is incorporated. Analysis shows that each Web server for a given site serves a unique set of Web objects discernible in a 3D plot. Furthermore, physical servers that occupy the same space, owing to servicing the same sort of content, can be consolidated into logical servers. Now, each Web site can be identified by the presence of transactions with specific servers, and a Bayes Belief Network is used to match the target Web site to the data set. The deanonymization technique was evaluated in a number of scenarios, including two real-world scenarios with real data. The evaluation found that complex but stable sites are very detectable; simple sites have high false detection rates; and volatile sites result in low true detection rates.

An audience member inquired how easy it would be to find multiple sites matching the same model (i.e., is there deniability?), Scott replied that it is hard to tell from the small sample set of Web sites they have tested thus far. He believes that the more complex the site is, the harder it will be to find a Web site with a similar fingerprint. Another audience member posited that the more sensitive part of the problem is identification of the client IP address. How clustered is the data if you try to detect IP addresses? Scott replied that previous work shows that client IP addresses do not provide much information about users. However, other knowledge, such as knowing a specific person is always in the office at 8 a.m., will go a long way toward deanonymizing that person's traffic. Finally, Roger Dingledine asked about the reason for including the sensitive fields in the first place, and, for that matter, who is allegedly using the data? Scott replied that simply removing the fields makes the data useless. There are a number of repositories (e.g., DHS predict) that are invaluable in areas such as IDS. Fabian Monrose added that the upshot is that past threat models are too weak and this work helps better understand anonymization. The goal is not to stifle the release of anonymized data but to get more people involved to help build better frameworks.

INVITED TALK

Covering Computer Security in The New York Times

John Schwartz, The New York Times

Summarized by Sarah Diesburg (diesburg@cs.fsu.edu)

John Schwartz began by commenting on the oddity of a reporter functioning as a speaker instead of being on the other side of the podium. Schwartz went on to discuss what he sees as a general view of mainstream media: The mainstream media often get technical stories wrong, and get them wrong repeatedly, because their reporters are not technically adept, are looking for scare stories, and are trying to get the newspaper equivalent of ratings. He was there to talk about why this isn't true for those in the reporting industry that work at the top of the game, and that it truly is possible to write about these issues without hype.

Schwartz jumped right into his views of the reporting industry, his values when reporting a technical story, trends in the newspaper industry, and general advice on getting a story reported correctly. One of his mottos is "Dare to be dull." He believes it is important to cover what is most important and that it will get out to the public even if it is not the front-page story. He acknowledges that some journalists are aiming to write "sensational" pieces, and he advises us to avoid them. Instead, find those journalists who hold the old-fashioned concept of "serving the reader."

He acknowledges the substantial effect the Internet has had on the newspaper business. People expect to get free news online, and this brings new challenges to the industry.

As the floor was opened to questions, one questioner asked about those doing the fact-checking for technical articles, as it seems that many of them are not very accurate. Schwartz commented that some publications tend to push journalists to write too much and too fast. He is also stunned by what he sees as lack of attention to detail and lack of caring. His advice is to figure out who in the industry gets it right. Another questioner asked whom he should contact if he has important knowledge. Schwartz said that local reporters are very accessible, and an offer of help can go a long way. In response to a question about how it seems reporters always want to give the other side of the story equal weight, even if one side of the story is obviously more accurate, Schwartz said that he believed that everyone affected by a story needs to be in it. Nonetheless, he added, there is a tendency in the media to equate balance with equal weight. He says many reporters don't reflexively understand the distinction, but a good reporter needs to understand this.

WORK-IN-PROGRESS REPORTS

Summarized by Adrienne Felt (felt@virginia.edu)

- ***VM-Based Malware Detection System***

  Yuhei Kawakoya, NTT Information Sharing Platform Laboratories

In this presentation, Kawakoya introduced two methods of externally monitoring the security of a virtual machine. Current antivirus software is installed in the same operating system that it is trying to protect, which gives malware a way to attack the antivirus software. However, Kawakoya's approach separates the protection mechanism from the protected operating system. In the first of the two methods, called Outside System Call Hooking, the host OS recognizes the invocation of a system call and checks the status of the guest OS. The second technique, Execution Cache Investigation, uses pattern matching to compare the cache with static virus signatures. An implementation demonstrated the effectiveness of this approach with samples collected from the wild.

- ***Controlled Reincarnation Attack to Subvert Digital Rights Management***

  F. John Krautheim and Dhananjay S. Phatak, University of Maryland, Baltimore County

This presentation explained how to use a virtual machine (VM) to circumvent digital rights management restrictions. The simplest way to use a VM to avoid copyright constraints is to store the software or media in a VM with the correct state (e.g., system time). This can be prevented by requiring communication between the product and a company server, with the server remotely maintaining the state of the license. Krautheim then outlined how virtualization technology can be used to fake the server's half of the communication to trick the product into believing it has received permission to be used. Copies of the VM are identical at startup, so the VM-server communication will be exactly the same for each instance of the VM. In a "controlled reincarnation" attack, the server's authentication can be sync'd and replayed indefinitely to provide access to the restricted content without needing to break the communication's encryption. Krautheim and Phatak are working on a defense for this attack that works by preventing the execution of restricted content in a virtual machine. The protected media would come packaged with a utility that uses timing benchmarks to detect a virtualized environment.

- ***The Performance of Public Key–based Authentication Protocols***

  Kaiqi Xiong, North Carolina State University

Kaiqui Xiong compared the performance of two authentication schemes that combine Kerberos and public key cryptography. Public-Key Cross Realm Authentication in Kerberos (PKCROSS) reduces the need for maintaining cross-realm keys so that Kerberos is easier to implement on large multirealm networks. Public Key Utilizing Tickets for Application Servers (PKTAPP) was intended to improve the scalability of PKCROSS, but Xiong showed that PKTAPP does not actually scale better than PKCROSS. He also showed that PKCROSS outperforms PKTAPP in multiple remote realms.

- ***Automatic Vulnerability Management Based on Platform Integrity***

  Megumi Nakamura, Seiji Munetoh, and Michiharu Kudo, IBM, Tokyo Research Laboratory

This presentation discussed automatic vulnerability checking, the goal of which is to simultaneously reduce the administrator workload and improve security. However, automated security tools have weaknesses and can be attacked. Nakamura et al.'s work uses a trusted platform module to store integrity and vulnerability information about the security tools so that they can be compared. When a discrepancy between the expected and actual integrity values is found, the vulnerability information is used to diagnose the problem. This places some of the responsibility for security on the hardware, which is more trustworthy than software-only approaches.

- ***Attacking the Kad Network***

  P. Wang, J. Tyra, T. Malchow, Y. Kim, N. Hopper, D. Foo Kune, and E. Chan-Tin, University of Minnesota

The authors of this work created a successful attack on the Kad network, which supports the eDonkey peer-to-peer file-sharing network. They found structural vulnerabilities that allowed them to interfere with the entire network's keyword search functionality from a small number of nodes. Their experiments showed that their attacks worked on eMule and aMule, the most popular Kad clients, and that a single user could halt 65% of Kad searches. These structural weaknesses could be exploited to target specific keyword searches or to hijack the network for DDoS attacks.

- ***Virtual Machine Introspection for Cognitive Immunity (VICI)***

  Timothy Fraser, Komoku, Inc.

Timothy Fraser presented a rootkit detection and repair system that uses virtualization and artificial intelligence to monitor the state of an operating system. In the setup, a GNU/Linux kernel is observed while running in a Xen virtual machine. The VICI system is trained to detect and undo the behavior of kernel-modifying rootkits. The artificial intelligence is based on the Brooks Subsumption architecture for autonomous robots and is expected to improve its performance over time.

- ***Polymorphic Shellcode Detection Using Emulation***

  Michalis Polychronakis, Foundation for Research & Technology—Hellas (FORTH)

Michalis Polychronakis presented a network-based behavioral analysis system designed to detect malicious shell-

code. The detector runs on a network intrusion detection system CPU that intercepts and inspects traffic. Unlike traditional malware techniques, their system does not do pattern matching for known static signatures. Instead, it executes the instruction sequences included in the traffic and analyzes the behavior of the code. The focus on behavior allows for the detection of new attacks and self-modifying polymorphic code. This approach was shown to be effective on a real-world deployment of the system. An audience member asked about false positives, and Polychronakis responded that the only two false positives were due to a bug that was subsequently fixed.

■ ***CANDID: Preventing SQL Code Injection Attacks***

Prithvi Bisht, University of Illinois, Chicago

Prithvi Bisht presented a defense against SQL injection attacks that compares the post-input query to the programmer's intended query structure. He said that a SQL injection attack, by definition, changes the structure of the involved query. The defensive system attempts to dynamically discover the structure of the programmer's intended query by evaluating the query behavior over known benign inputs. The system has been implemented in a tool, CANDID, that transforms and thereby safeguards Java Web applications.

■ ***Protecting User Files by Reducing Application Access***

William Enck, Patrick McDaniel, and Trent Jaeger, SIIS Lab, Pennsylvania State University

This presentation introduced new file access controls that prevent files from being accessed by programs that are not on a permission whitelist. PinUP, an access control overlay for Linux, extends file system protections by explicitly defining program access permissions at the inode level. This approach is intended to make file access control more intuitive and secure by saying that data should only need to be accessed by applications that the user specifies. An audience member asked whether PinUP is vulnerable to a confused deputy attack, but Enck clarified that this is not currently a problem since PinUP has not yet been implemented for binary applications.

■ ***Securing Web Browsers Against Malicious Plug-Ins***

Mike Ter Louw, University of Illinois at Chicago

Mike Ter Louw described the dangers of Web browser plug-ins and how to defend against them. To demonstrate the potential harm that could be done by a plug-in, the authors of the work created a malicious extension called BrowserSpy that can be installed without special privileges. Once installed, it has access to all the functionality of the browser. They then created a code integrity checking technique to control the plug-in installation process and defend against malicious extensions. They are also exploring behavioral analysis to monitor the actions of extensions. More information and details can be found in the work "Extensible Web Browser Security," published at the Detection of Intrusions and Malware and Vulnerability Assessment in 2007.

■ ***Detecting ISP-Injected Ads with Web Tripwires***

Charles Reis, Steven D. Gribble, and Tadayoshi Kohno, University of Washington; Nicholas C. Weaver, ICSI

Some ISPs have begun altering Web page content to add advertisements and paid links to Web sites. The authors of this work created a Javascript Web tool that allows users to determine if their Web content is being modified by ISPs. The tool is aware of its intended content and can detect when it has been changed. Their site received approximately 50,000 hits from Digg and Slashdot, leading them to identify several kinds of ad injections. They have not yet revealed the names of the ISPs that are practicing this form of advertising. The tool is available at http://vancouver.cs.washington.edu. The topic proved extremely interesting to audience members and received several questions. One audience member asked whether it was possible for them to differentiate between site changes made by a client-side adware proxy and ISPs; Reis admitted that it is not always possible, but they tried. Another asked whether ISPs could use this same method of site code injection for a beneficial service such as local tailoring, but Reis clarified that although that could be good it is not what is happening.

■ ***Leveraging Non-Volatile Memory for Advanced Storage Security***

Kevin Butler, Pennsylvania State University

Kevin Butler introduced ways to use fast-access non-volatile RAM-enhanced hybrid disk architectures to facilitate secure processes such as authenticated encryption. A secure boundary is placed between the disk interface and the operating system, with the operating system as an untrusted party. This approach is significantly more sophisticated than what is possible with traditional storage systems.

■ ***Tor***

Roger Dingledine (arma@mit.edu)

Tor is an anonymous browsing site funded by various organizations including the DoD and EFF. The third largest group of users is in China, where Tor gives users the ability to circumvent Chinese Internet censorship. Roger Dingledine challenged the audience to help plan for the possibility that China will try to block access to Tor. One suggestion is to allow a non-Chinese user to act as a relay between the Chinese user and a Tor bridge, but how can this be set up? It would be necessary to hide thousands of IPs from the Chinese government yet give them out one at a time to users. Dingledine discussed the use of scarce resources, such as time and IP addresses, to determine which IPs are distributed. He said that the project is looking for members and needs ideas from the security community.

2007 USENIX/ACCURATE Electronic Voting Technology Workshop

Boston, MA
August 6, 2007

ANALYSIS I

Summarized by Kyle Derr (derrley@rice.edu)

- ***Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective***

  Rop Gonggrijp and Willem-Jan Hengeveld, Stichting "Wij vertrouwen stemcomputers niet"

Rop Gonggrijp gave a narrative summary of his experiences in dealing with the general issue of electronic voting in The Netherlands, and he presented the findings from his security analysis of the Nedap ES3B voting computer.

Gonggrijp began his talk by explaining that while the argument for voting on paper with pencil is rather compelling in the United States, it is even more so in Holland, owing to the simple (and short) nature of the information that the average voter intends to communicate. When electronic voting machines were installed and used in Amsterdam for the first time in 2006, it was cause for significant alarm from an expert in computer security living in Amsterdam (as he was at the time). Not only were these machines most likely not secure, but they were also mostly unnecessary, as there does not exist a Netherlands analogue to what happened in the United States in Florida in the 2000 presidential election. The low-tech solution was entirely adequate.

Gonggrijp's security analysis yields two feasible attacks, which he spent the majority of his talk explaining.

The first attack demonstrates that because of the lack of physical security on the device coupled with the vintage nature of the hardware, reverse engineering, modifying, and installing malicious software is a trivially simple task. In fact, his team was able to do just this in under a month's time. The security of the physical components can easily be compromised by an intruder; the model of physical lock used to keep intruders from accessing the machine's internal components only has a single, universal key, which can be purchased for less than $5 from any one of several retailers which can be found via a simple Google search for the key's product number. The software that drives the machine is programmed on a pair of EPROMs, which are inserted into a pair of sockets on the motherboard. This property allows an attacker to swap the good pair of EPROMs with a set that contains malicious code (such as the software his team developed, which silently steals votes from randomly chosen candidates and delivers them to a preconfigured party) in less than 60 seconds.

The second attack demonstrates that voter privacy and anonymity can easily be compromised without any physical access to the device. The LCD used to display instructive information to the voter only supports mostly-ASCII text. A certain number of non-ASCII special characters can be used on the display. In the case where these non-ASCII characters are displayed, because the display controller must wait for this additional character information to be given to it from the computer, the refresh frequency of the display drops. This change in frequency is audible in some cases as far as 20 meters away with the assistance of a short-wave radio receiver. Because a popular Dutch political party has such a character in its name, a vote for this party can be detected from outside the polling place.

Gonggrijp's responses from audience questions focused mostly on rectifying confusion regarding why such an audible frequency change was being emanated from the device. In addition, Gonggrijp also forcefully noted, again, that paper ballots are more than adequate in The Netherlands. Citizens must *go for the throat* of manufacturers such as Nedap, insofar as their systems do not provide a significant advantage over paper, all things considered, accessibility gains included. When questioned specifically about the DRE's accessibility wins, Gonggrijp responded that the gain in accessibility is negligible compared to the loss in trustworthiness of the system as a whole.

- ***Security Analysis of the Diebold AccuVote-TS Voting Machine***

  Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten, Princeton University

Ariel Feldman presented the results of his team's independent analysis of the Diebold AccuVote-TS. Diebold has the largest market share among US vendors: This family of Diebold machines (the AccuVote-TS and the AccuVote-TSx) currently records more than 10% of the vote nationwide. His team's contributions include a program that can silently, undetectably, and arbitrarily alter election results and a virus that can propagate this program.

Feldman first described the vote-stealing software his group designed and implemented. Because the AccuVote-TS family machines are simply general-purpose computers running a general-purpose operating system (in this case, Windows CE), Feldman and his team were able to design their software to run as a separate Windows process; patching the actual Diebold source code that runs on election day was unnecessary. The vote-stealing software simply directly edits the cast ballot data and modifies the log data such that it is consistent with the fraudulent cast ballot data. It accomplishes this by taking advantage of the fact that although this stored data is encrypted, it is encrypted with a well-known DES key (which is hard-coded in the source).

Feldman next described three different mechanisms that an attacker could exploit to install such malicious software: An attacker could boot from a malicious EPROM in an onboard socket, install a malicious boot loader by using

the software upgrade mechanism, or develop a voting-machine virus that propagates using the software upgrade vulnerability.

Whereas gaining access to the motherboard to install a malicious EPROM is a bit cumbersome, installing a memory card that contains a malicious boot loader disguised as a legitimate software upgrade is a fairly trivial task. Poll workers install and remove these memory cards as part of election day procedures; therefore all that stands between an attacker and the memory card is a locked door. Because all Diebold machines are known to be fitted with locks that can all be opened with a universal key (a key that can be obtained via several Internet retailers, including Diebold), this lock would not pose a problem to an attacker. Feldman's team was even able to clone one of these keys simply by using as a source a photo of such a key taken from the Diebold Web site. Once the malicious memory card is secured, the machine needs to be rebooted. Upon rebooting, the machine will install the software included on the memory card without asking for any sort of cryptographic guarantee that the software is from a credible source.

In addition, Feldman explained that an attacker could also exploit this software upgrade facility to create a voting-machine virus, making physical contact with every target voting machine unnecessary. The malicious boot loader installed from the attacker's memory card can clearly do much more than install the vote-stealing program: It could also disable future upgrades and copy itself to any memory card inserted into the voting machine in the future. The reason this is a credible threat can be reduced to procedure: It is common practice (indeed, Diebold-recommended practice) for a county to use several machines as accumulators in the tallying process at the end of the day. This process requires that memory cards that contain cast ballot data be inserted into the machine being used as an accumulator for the purpose of including this cast ballot data in the tally. If the accumulator machine were to be infected, it could easily infect hundreds of other machines in the course of one tally. By the same mechanism, a machine could infect the accumulator, which in turn would infect all other machines it touched throughout the course of the tally. It is reasonable to assume that a somewhat motivated attacker armed with a virus such as Feldman's could gain control of all voting machines in a county during the course of one election,in order to drastically affect the outcome of the next.

Interesting questions from the audience included one from Peter Neumann, who pressed Feldman to reveal an even more recent exploit, which allows the attacker to not have to reboot the infected machine, but simply use a buffer overflow in the running Diebold software. When asked if his virus could change the votes in real time (so as to alter printer output), Feldman responded that his team had not attempted it. When asked if a voter (as opposed to a poll worker) could easily launch the attack, he seemed skeptical, because of how obvious it would look if a voter tried to walk up to a machine and install a memory card. The session was wrapped up with Dan Wallach's wondering how Feldman responds to the common claim by election officials that election procedures defend effectively against these kinds of attacks. Feldman answered that procedures aren't always followed, and even if the ones in question were followed to the letter, they still wouldn't adequately defend against the kinds of attacks his team has developed.

■ ***An Analysis of the Hart Intercivic DAU eSlate***

Elliot Proebstel, Sean Riddle, Francis Hsu, and Justin Cummins, University of California, Davis; Freddie Oakley and Tom Stanionis, Yolo County Elections Office; Matt Bishop, University of California, Davis

Elliot Proebstel summarized four attacks and related mitigation strategies his team developed while conducting a Red Team, Black Box analysis of the eSlate DRE on behalf of Yolo county. The team found that a motivated attacker could use a covert channel to gain knowledge of a voter's choices, could cause the VVPAT printer to print duplicates easily mistaken for unique ballots, could modify vote tally data mid-day, and could use the lack of randomness in voter ID numbers to create a ballot-stuffing attack. Proebstel also offered a number of attack scenarios that could cause record inconsistencies.

The particular class of eSlate machine Proebstel's team analyzed was a Disabled Access Unit. This means it contains two features that help disabled voters. For the hearing impaired, Hart has added an audio jack near the rear of the machine, whose job is to emanate a reading of the screen content, as well as to audibly notify the voter of his or her selections. Proebstel explained that although this feature obviously makes the machine accessible to the hearing-impaired, it is also a potential source of privacy loss for the average voter. The audio can easily be captured by a hidden iPod-sized device. Because the audio must help disabled voters both through the process of entering their voter ID numbers and through the process of expressing their voting preferences, this hidden device would be able to capture enough information to link a voter to a cast ballot. Proebstel also explained that a more recent study indicates that a shortwave radio can pick up this audio, making the hidden device and physical access to the machine unnecessary. As a mitigation strategy, Proebstel offered that poll workers should be trained to detect such suspicious behavior.

Proebstel next explained a feasible ballot-stuffing attack. The machine that his team studied was connected to a thermal printer, whose purpose is to record, on paper, the voter's preference so that a manual recount is possible in case it is needed. Each printout contains a plain-text recording of the voter's preferences (intended to be verified by the voter shortly after the ballot is cast) positioned

above a barcode encoding of the same data (for ease of counting at a later date). Proebstel's team found that if communication was interrupted between the eSlate and the printer when the ballot is cast, this printer would print duplicate barcodes (without corresponding attached plain text). If the scanning of these barcodes is automated, this is a feasible ballot-stuffing attack. As a mitigation strategy, Proebstel suggested that humans sort through the printouts to ensure that each barcode is attached to corresponding plain text. An even simpler option, Proebstel suggested, is for humans to perform the entire recount.

The next attack Proebstel explained would require corruption at the poll-worker level. The eSlate machines store the effects of each cast ballot in three locations: in internal eSlate memory, in internal memory of the Judge Booth Controller, and on a removable memory card installed in the Judge Booth Controller (called a Mobile Ballot Box, or MBB). Per Hart's recommendation, Yolo county uses the MBB as the single, trusted source of election data, except in the case where a recount is necessary. Proebstel's team found that the MBB can be easily removed and modified on a simple laptop computer, then reinserted into the Judge Booth Controller, without the Judge Booth Controller noticing any inconsistency. As a mitigation strategy, Proebstel suggests strict chain-of-custody procedures be implemented and followed and that physical tamper evident seals be used on the voting machines themselves.

Proebstel's team also found that not only is the order in which voter ID numbers are chosen predictable, it is the same on all Judge Booth Controllers. When a voter walks into the polling place on election day, the voter authenticates him- or herself with election officials in some way and then is assigned a voter ID number (which is not recorded). This number is printed by the Judge Booth Controller on a small piece of paper, which is then handed to the voter. The voter then waits for a machine to become unoccupied. The voter will only enter his or her ID number into an eSlate once one becomes available. An attacker with knowledge of this predictable ID order could potentially cast ballots for other queued voters. The Disabled Access Unit input jack near the back of the eSlate only makes matters worse, as the attacker could automate much of this process. As a mitigation strategy, Proebstel suggests that only one eSlate be used per Judge Booth Controller, and only one voter ID be active at a time.

Questions from the audience included one that pressed Proebstel to compare his experience in a black box analysis of the eSlate to the one he conducted more recently, where he had access to source code. Proebstel stated that access to source made the process much easier, and he opined that now that source access is being established as precedent, future analyzers should require this. When asked if he had any advice for researchers who want to try to get election officials to work with them, rather than against them, he said he did not. In his case, Yolo county officials initiated the conversation.

DESIGN I

Summarized by Kyle Derr (derrley@rice.edu)

■ *Casting Votes in the Auditorium*

Daniel Sandler and Dan S. Wallach, Rice University

Daniel Sandler gave a narrative of his experience collecting post-election evidence on ES&S machines used in a Democratic primary election in Webb County, Texas, and how this experience motivated him to design Auditorium, a secure and distributed logging environment that makes post-election auditing easier and makes audit data much more tamper-evident and tamper-resistant. Sandler also described several potential attacks that would succeed on standard-issue logging environments but that Auditorium would prevent. The description of these attacks is omitted for brevity.

In 2006, Dan S. Wallach was hired as an expert witness by the close-second-place finisher in a Democratic primary in Webb County, Texas, after said second-place finisher received more votes on paper than he did on the ES&S DRE machine. Wallach's job was to perform a post-election audit of the records found on these ES&S machines. If enough evidence could be gathered from audit logs to prove the tallies retrieved from the ES&S machines were inaccurate, this second-place finisher would have a decent case in court. Sandler was the primary investigator in this endeavor.

Although Sandler could not find enough evidence in the logs to do his advisor's client any good, he did find many anomalies in the audit data found on the machines, such as logs starting mid-day (indicating a loss in audit data for half of the election), events taking place several days prior to the election (including 26 machines that had the same two votes cast, indicating possible inclusion of test votes in final tallies), and some machines that had no audit data at all. For the audit data that Sandler did find to be benign, he had no guarantees of the correctness or authenticity of this log data, as ES&S audit data is stored on simple, rewritable flash memory in plain text. His conclusion from this investigation was that audit data needs to be replicated in as many places as possible (to mitigate against accidental loss) and to be cryptographically authentic (to mitigate against pre-election or post-election modification). In short, it should be harder to make mistakes on election day that could prevent a provable audit after election day, and it should be easier to perform this audit after election day, even if some accidental loss has happened.

Sandler then described Auditorium, a secure, distributed logging network—his solution to this problem. In Auditorium, logs are hash chained, entangled, and broadcast. By assuming the existence of a one-way hash function, hash-chaining log entries allows an auditor to have a much higher degree of certainty about their relative order, given a certain amount of unpredictable data in each log entry. Entangling the log entries essentially makes this hash-

chaining process global across all machines in a polling place. In an entangled log, it is not the case that each machine constructs its own timeline of events: Events in one machine's timeline are necessarily dependent on events in the timelines of other machines. Because entries in the log are signed, forgery of a log event would require collusion of every machine in the network. Finally, broadcasting log events allows for replication. Because no single machine can be trusted to store everything, his design forces every machine to store everything. The result of using Auditorium on election day is a global log, stored on every machine in the polling place, whose entries can be cryptographically proven authentic and can be cryptographically reconstructed into a provable timeline.

Interesting questions from the audience indicated a bit of skepticism regarding voting machines being on a closed network. To address real-time ballot stuffing by a machine, Sandler explained that a cast ballot would not be counted unless a corresponding authorization to cast from a supervisor machine is present. This means that the supervisor machine and a voting machine would have to be in collusion. Furthermore, an act of stuffing would be noticed and logged by good machines. This is simply not possible without a network. Sandler gave a similar response when he was questioned about possible network partitions causing audit data loss: Because each machine requires an authorization to cast the ballot, this partition would be quickly noticed and rectified, causing at most the loss of one vote cycle being broadcast. Correct design would mitigate against even this, causing the broadcast of this pending vote as soon as the machine is able to reconnect to the network. Peter Neumann pressed for a better explanation of denial of service mitigation. Sandler explained that such an attack would have to disrupt every machine in order to prevent audit data from at least being replicated once.

■ ***Extending Prerendered-Interface Voting Software to Support Accessibility and Other Ballot Features***

Ka-Ping Yee, University of California, Berkeley

Ka-Ping Yee gave a summary of the extensions he made to Pvote (his prerendered user interface voting engine) to assist the visually impaired. These extensions allow the voter to *hear* as well as *see* the ballot being presented on screen. He also rehashed his justification for a prerendered user interface and small runtime code base.

Yee began by explaining that although efforts have been made to force voting machine vendors to release their source code for review, pressing for this solves only half the problem. Most voting machines in use today rely on a complex and difficult-to-audit code base. Pressing for disclosure of this code simply does not solve the problem if an audit is incredibly time-consuming and cumbersome. Yee suggested that the trusted code base needs to be explicitly small, so that an audit is simple given full source code disclosure. Furthermore, Yee emphasized that the best way to minimize the amount of trusted code is to offload determination of user interface behavior to the tool that generates the ballot. This effectively makes the user interface *prerendered*. For a given election, then, all that should be trusted is the ballot definition, with its associated user interface, and the small amount of code running on a voting machine that allows the voter to interact with and cast said ballot. As a demonstration of the simplicity of this voting machine logic Yee said that his Pvote system does just this, and it is written in 460 lines of Python.

Yee then demonstrated his system to the audience. His software did, indeed, emanate a audible recording of the screen's contents. It also emanated information regarding which option was currently selected on screen.

Comments that Yee made during his talk regarding trusting general-purpose software (such as the Python interpreter) were the cause of most of the debate during the question period of this talk. Warren D. Smith expressed specific concern with trusting software insofar as it is general-purpose: The generality of the software, Smith claims, actually proves nothing regarding its trustworthiness. Yee acknowledged that it is an open question whether or not to trust general-purpose software, but his opinion is that because the software existed before this particular application of it, and because the software isn't made specifically to run voting machines, it should be more trusted than specific voting software. Ron Rivest suggested that Java might be a better option than Python, because Java runtime environments are quite a bit more studied.

■ ***Verification-Centric Realization of Electronic Vote Counting***

Joseph R. Kiniry, Dermot Cochran, and Patrick E. Tierney, University College Dublin

Dermot Cochran gave a summary of his work, which involved his explanation of how formal specification can be used to ensure correctness of critical path voting machine modules (such as the vote counter). His talk focused on how verification-centric software engineering practice produces software that is more trustworthy. This paper's contribution is a protocol for good verification-centric software engineering practice, as well as a case study.

Cochran went over the process his team used for extending KOA, a research platform for electronic voting technologies, to support the Irish proportional representation Single Transferable Vote system, whose algorithms are legally specified. In the analysis and design phases, EBON (Extended Business Object Notation) is used to model the problem, as well as to strictly model contractual relationships between different modules of the design. These contractual bindings include preconditions, postconditions, and more general behavioral assertions. During the specification phase, this generic EBON specification is formalized using JML (the Java modeling language). JML specifications for each method and class can be formalized in JavaDoc comments, and verification that the Java code

meets the specification written in JML can be done by the JML tool chain. Currently, in the implementation and testing phase, not only can passing unit tests ensure confidence in the correctness of the code, but the code can also be tested against the formal JML specification defined in the previous phase.

Cochran showed several code snippets from his case study, highlighting Java code next to its formal JML specification.

During the question period, David Wagner pressed that some of Cochran's examples seem to hint that frequently the JML expresses exactly the same semantic concept as the Java code itself. Wagner wondered whether the JML was simply expressing the same thing and was therefore superfluous. Although admitting that this is sometimes the case, Cochran said that often it is not the case (since sometimes complex preconditions and postconditions can be more succinctly specified in JML). What's important is that the contractual specification happens before the implementation happens and that this specification can be formalized in such a way that the actual code can be checked against it.

AUDITING AND TRANSPARENCY

Contractual Barriers to Transparency in Electronic Voting

Joseph Lorenzo Hall, University of California, Berkeley
Summarized by Kyle Derr (derrley@rice.edu)

Joseph Lorenzo Hall explained his work on analyzing the contractual agreements state and local election jurisdictions make with electronic voting machine vendors, and he suggested how many of these agreements can blatantly challenge election transparency. He also made several recommendations for how these contractual agreements should be changed in the future.

First, Hall explained that this sort of research is challenging, because analysis must be done on a convenience set: Sometimes the contracts themselves are considered proprietary information, making them hard to acquire for study. Five major vendors were present in his data set of 55 contracts, whose signing dates were distributed between the years 2000 and 2006, and 82% of the contracts were with the biggest three US vendors: Sequoia, Diebold, and ES&S.

Hall organized the subset of the findings he chose to present into the following categories, where each category represents a type of transparency-barring clause. Hall found clauses that protect trade secrecy, prohibit certain types of use, discourage public record, separate escrow agreements, and limit disclosure of benchmarking and require mandatory software upgrades. Examples of trade secrecy clauses include disallowing both source disclosure and reverse engineering. In some cases *analysis* of the system is categorically prohibited. In one case, even the unit's pricing information is considered proprietary. Use prohibitions include restrictions on what hardware can be used to run the software and where (geographically) the software can be run. As for public record protection, sometimes Hall found the contract itself to be considered confidential. He also found clauses that could limit the liability of the company in the case where damages were sought because of confidential information being released as a result of legislative or judicial requirements of certain pieces of information being in public record. One of the most egregious was a mandatory upgrade provision, which forced the client to install all software upgrades no more than 10 days after their release. This, of course, means that the software upgrade would not be certified in any fashion.

Hall made several general recommendations for future clients looking to protect and encourage transparency in elections. First, contracts should always be disclosed. There is no known reason why revealing these contracts could hurt the vendor. A more pressing reason is each voter's right to know what is in them. He also suggested that limited access to source code, ballot definitions, audit logs, and vote data should be allowed, testing should not be forbidden, and damages should certainly not be limited in the case where they are incurred because of public records disclosure.

During the question session, Peter Neumann asked why only *limited* access to source code should be allowed. Hall responded that until the quality of the software improves, disclosing it to anyone who wants it will encourage attacks. Limited disclosure to trusted, third-party sources, however, cannot be stopped, as this is a useful avenue for performing audits. When asked whether any contracts speak about security obligations of the localities, Hall responded that they didn't, and that the closest thing to this is barring security analysis. When asked how long these contracts typically last, Hall responded that most clauses are in effect for the entirety of a machine's use, and in some cases, clauses do not expire (for instance, those related to proprietary information).

On Estimating the Size and Confidence of a Statistical Audit

Javed A. Aslam, Northeastern University; Raluca A. Popa and Ronald L. Rivest, Massachusetts Institute of Technology
Summarized by Kyle Derr (derrley@rice.edu)

Raluca Popa gave a summary of her simple formula for determining the number of precincts running DRE voting machines that need to be manually audited, given a desired level of confidence that no precincts have been compromised. Because it is likely that this formula will need to be computed on a hand calculator (as trusting another piece of software in elections is definitely not wanted), this formula needs to be simple and operate on few parameters. The form of audit she suggests is a simple comparison of VVPAT printouts to electronic records. The formula depends on n, the number of total precincts, b, an upper bound on how many of them could be corrupted,

and *c,* the required confidence level. Popa showed that her formula was almost exactly accurate, with error only in the positive direction, and by no more than ceil((*n* – (*b* – 1)/2).

Questions from the audience included whether the strategy assumed that there is a maximum number of two candidates per race and whether differently sized precincts were considered. Popa clarified that her formula works with any number of candidates per race and precincts of varying size: Both of these factors go into the calculation of *b*. When asked why not simply distribute a table rather than the formula, Popa claimed the table could be corrupted but that the formula is simple enough to be distributed and computed in the field. When asked if this scheme could be applied to vote-by-mail, Popa claimed that some sort of granularity would need to be applied to group the sets of cast ballots, even if it was somewhat synthetic (as would be the case in vote by mail).

■ ***Machine-Assisted Election Auditing***

Joseph A. Calandrino, J. Alex Halderman, and Edward W. Felten, Princeton University

Summarized by Elliot Proebstel (proebstel@ucdavis.edu)

Joseph Calandrino presented a short talk on Princeton's recent work in machine-assisted election auditing. Motivated by observations on electronic voting system flaws and the costs associated with traditional paper audits, the authors worked to develop software-independent auditing mechanisms that would work within a fixed budget. The authors, building on past work by C.A. Neff in 2003 and K.C. Johnson in 2004, suggest a methodology whereby ballots are machine-tallied, printed, and stored during an election. After the election, the ballots are scanned and sequentially numbered by a specialized recount machine; poll workers manually check that the recount machine correctly numbered the ballots and verify the ballot contents of a number of sampled ballots.

Calandrino also referenced the statistical research done by the team, which effectively reduces the number of ballots that must be included in a recount in order to achieve 99% confidence in the results. By implementing ballot-based audits using the authors' recommendations, Calandrino reported, the total number of ballots that would need to be audited in the Webb vs. Allen race (of Virginia's November 2006 elections) would be reduced to 2,337. This is in contrast to the 1.14 million (out of a total 2.3 million ballots cast) that would need to be audited using precinct-based auditing to achieve the same confidence. Extensions to this work include auditing only the ballots of winning candidates. In the future, the authors plan to consider cost estimates and also practical concerns, such as how to deal with errors.

Various audience members pointed out that the proposed scheme is illegal in Virginia and also that in the U.K. and New Zealand, ballot serial numbers are required by law. Someone asked about a hybrid approach that includes both precinct-based and ballot-based auditing. Calandrino responded that it depends on your definition of "hybrid." When only sampling within a precinct, the savings are smaller than expected.

■ ***An Examination of the Auditability of Voter Verified Paper Audit Trail (VVPAT) Ballots***

Stephen N. Goggin and Michael D. Byrne, Rice University

Summarized by Elliot Proebstel (proebstel@ucdavis.edu)

Stephen Goggin gave a short talk on the findings of a recent study on the auditability of Voter Verified Paper Audit Trail (VVPAT) ballots. Noting that HAVA suggests VVPAT usage, and also that 37 states legally require VVPATs, the authors looked into the difficulty of auditing the records generated by the thermal-receipt-style printers that have been installed (often retrofitted) onto Direct Recording Electronic (DRE) voting systems. The use of VVPATs has two goals: to force the voter to verify a paper copy of the ballot, and to produce a physical record for auditing. The authors assumed that the first goal was met (while mentioning that this assumption is generous and may not hold), and then set to study how error-prone and costly the auditing of such records would be.

The authors generated fake ballots, closely following the VVSG standards, trying to ensure that their ballots looked realistic. There were 120 ballots per spool, and each ballot was 2 feet long. In compliance with the VVSG standards, all ballots had "rejected" or "accepted" notation. The spools were set onto a recount fixture, and the recruited testers were given scissors and a tally sheet. These auditors (undergraduate students with good vision and fluent in English) were asked to cut apart the ballots and tally the results of a single race. After they completed this race, they were asked to go through the ballots again to tally the results for a second race.

On average, the auditors completed the first tally in 25 minutes and the second tally in 12 minutes. When generalized for full-scale elections, this represents a time requirement that is untenable for large jurisdictions. Furthermore, the error rates were particularly notable. The authors found variation from a 17% undercount to a 19% overcount. In particular, when the race being counted was lopsided, auditors tended to overcount rejected ballots at a much higher rate, suggesting that the performance of individuals counting ballots is biased by ballot contents and expectations. Auditors reported a low level of confidence in their performance, and the authors found that confidence levels did not correlate with the actual accuracy of individual auditors.

An audience member interjected to ask why 19-year-olds were recruited for this task and how their performance can be correlated to the performance of real election auditors, whose demographics were not represented in this study. Goggin replied that the authors had chosen the students as a best-case scenario: Younger participants with good eyesight were likely to be more efficient and accurate than older auditors. He concluded the talk by reporting that

these results indicate that recounts will require considerable labor and high cost, be subject to human error, and likely be influenced by auditor bias. Future work will include a comparison of auditing VVPAT records with auditing paper ballots, as well as an examination of the VVPAT usability for voters.

Q: You may need to check your assumptions about the process. Specifically, are you sure that real election auditors will separate the ballots from the spool? Also, you should look into the design of the accept and reject messages at the tail of ballots; I'm not sure your font and size choices were accurate representations. Finally, you should research the counting procedures that are actually used in jurisdictions and try to mimic those, especially with regard to your choice of having a single counter auditing the ballots. I'm not sure that's representative.

A: First, the separating of ballots from the spool made things easy for the participants. If they aren't separated in real elections, this would actually generate worse results. With regard to the font and size choices, we tried to mimic a real VVPAT by using the VVSG "large font" specifications. And we know that there are some counties that use a single counter to audit ballots.

Q: I watched the eSlate counting in San Mateo. There were three people counting. Ballots were sorted and then counted, and this was done twice. Maybe you could do follow-up work to test this?

A: Yes, that would be a good idea.

Q: Could this be addressed with machine-assisted auditing?

A: There are issues of trust. It would need to involve barcodes or op-scan systems, both of which have trust issues.

Q: Did you display undervotes or just print who was voted for?

A: We printed: [X] No Vote.

ANALYSIS II

Summarized by Elliot Proebstel (proebstel@ucdavis.edu)

- ***On the Difficulty of Validating Voting Machine Software with Software***

  Ryan Gardner, Sujata Garera, and Aviel D. Rubin, Johns Hopkins University

Ryan Gardener presented the work being done at Johns Hopkins University to investigate options available to allow poll workers to verify the authenticity of software running on electronic voting machines. The adversarial model used for this work assumes the attacker has full control of the software but is unable to make any hardware or firmware changes—even those that are seemingly benign. Gardner explained that neither direct hashing nor such hardware-based solutions as hashes signed by a TPM would suffice, because these options both ultimately rely on trusting sources that cannot necessarily be trusted to report honestly. Thus, the authors searched for a primitive that could be trusted.

Current state-of-the-art software attestation, Gardner explained, is a product called Pioneer, which is from Carnegie Mellon University. Pioneer resides in the memory and allows the verifier to provide a challenge to the system and then verify both the checksum and the time required to produce it. Because Pioneer is designed for optimal implementation, the additional instructions required to subvert the checksum process show up as overhead during runtime. Gardner's team increased the number of iterations run by Pioneer in order to magnify the attack overhead so that it could be human-measurable.

However, the authors found that even to raise the attack overhead to 3 seconds, Pioneer had to be run for 31 minutes. Gardner reported that requiring poll workers to wait for Pioneer to run for over half an hour and then detect a 3-second delay is not an acceptable solution. Furthermore, the team suspected that attacks on Pioneer-type solutions would only become more effective over time, as increased parallelization and faster CPUs incorporated into voting systems more effectively conceal attack overhead.

There were several questions about how Pioneer functioned, including about checksumming all of memory (which Pioneer doesn't do). Someone else asked whether there is a possible way to exploit the time it takes for the human to verify the long checksum? Could the checksum change itself after completion while the human is attempting to verify it? Gardner suggested reading the paper. Another person asked about the reproducibility of the timings and whether CPU temperature affect this. Gardner answered that they tested the timings at different times of day, but they did not go too far into this. Finally, someone asked whether the poll worker is supposed to use a stopwatch. Gardner said that they recommend that the poll worker use an alarm, which will go off at the expected time.

- ***An Authentication and Ballot Layout Attack Against an Optical Scan Voting Terminal***

  Aggelos Kiayias, Laurent Michel, Alexander Russell, Narasimha Sashidar, Andrew See, and Alexander A. Shvartsman, University of Connecticut

Andrew See reported on the vulnerability analysis conducted at the University of Connecticut on the Diebold AV-OS and, more recently, the AV-TSx. Previous vulnerabilities discovered in the AV-OS required access to a reader/writer for the memory card, but the authors of this work found that they could execute attacks on the AV-OS using only direct access to the device and a serial connection to a laptop.

After booting the AV-OS into debugging mode, the team was able to imitate the GEMS server with a laptop, because the AV-OS does not perform authentication on the serial connection. Using this access, the team was able to recover a dump of all memory card contents. From these contents, the team extracted the supervisor PIN and used it to enter

supervisor mode on the AV-OS. The team was then able to disable the AV-OS printer, edit communication parameters, and erase or replace memory card contents. Leveraging public knowledge about the ballot configuration, the team could use this access to remap candidate names to arbitrary locations on the ballot, allowing them to, for example, swap votes between two candidates or invalidate all votes for a given candidate. The attack code could also use cues available to it, such as the time of day and total number of ballots cast on the unit, to make an educated guess about whether it was being tested; if it suspected it was, it could report accurate results rather than attack results.

On the AV-TSx, the team developed similar attacks on the ballot layout. The text to be displayed for a candidate's name is stored in an RTF file. By swapping two RTF files on the memory card, the team could swap votes between two candidates. The database results would not match VVPAT records, but this might not be detected.

The team recommends that voting machines must ensure that ballots, VVPATs, and electronic results are consistent. This should be designed into the system, and ballot layouts should be auditable.

One audience member mentioned that Sequoia systems had been found to have similar flaws, so Sequoia has implemented the printing of (x,y) coordinates on VVPAT records. Another audience member suggested that the VVPAT could print the image from the screen in order to raise compatibility with prerendered ballots.

■ ***GEMS Tabulation Database Design Issues in Relation to Voting Systems Certification Standards***

Thomas P. Ryan and Candice Hoke, Cleveland State University

Candice Hoke presented work that was proposed in September 2006, before the team had privileged access to any GEMS (Diebold election management software) databases. The authors have subsequently had access via their work at the Center for Election Integrity in Cleveland and as public monitors for Cuyahoga County, but Hoke stressed that this work was not born of that privileged access. Furthermore, despite having participated in the California 2007 Top to Bottom Review, her contract required that she not speak of any results from that study, because the document review reports have not yet been released.

Hoke presented findings to prove that GEMS is seriously flawed in architecture and technology, resulting in data errors and erroneous results. The team outlined industry standard design requirements for database software, known as 1NF and 2NF (First Normal Form and Second Normal Form, respectively) and explained how these requirements help ensure the integrity and accuracy of database contents. Then, the team provided copious examples to demonstrate that GEMS violates both 1NF and 2NF, resulting in data errors, anomalies, and no notification that errors are occurring. Anecdotal evidence from Cuyahoga county supports these theoretical claims: Election officials reported that issuing the same query in different ways resulted in different responses, and the public audit found evidence of database corruption. GEMS uses Microsoft JET, which even Microsoft has publicly warned users is inappropriate for systems where absolute data integrity is essential.

Next, Hoke demonstrated that the federal regulatory system encourages lower standards for database design. By requiring more documentation from vendors who identify higher quality and higher horizons for database design, the federal regulatory system effectively streamlines the process for vendors with lower design standards. This reverses the incentive structure, favors low-quality design, and fails to level the field with uniform standards.

Hoke concluded with a strong call for technical experts and regulatory lawyers to forge strong partnerships in order to favor regulatory structures that will generate higher quality electoral performance and other key public functions. This partnership would focus on multiple fronts—academic, legislative, administrative, and judicial—as well as involving the media in an attempt to overcome the "fog" (glazing over of technical issues). She suggested that computer security experts are uniquely qualified for this interdisciplinary work.

The first questioner asked about avoiding the problem of regulations that suggest particular technical restrictions rather than general functionality. Hoke answered that the regulatory system structure should be discussed by both legal and technical communities. It must be played out. Hoke is in favor of a "federal floor but not a ceiling"; we should have baselines, but not limits. A second questioner commented that this work focuses on properties of relational databases instead of object-oriented databases. How can we extend this? Hoke answered that we should have discussions to find the best possible solution, weighing technical and realistic aspects. However, we cannot have federal standards that operate as a ceiling.

DESIGN II

Summarized by Elliot Proebstel (proebstel@ucdavis.edu)

■ ***Ballot Casting Assurance via Voter-Initiated Poll Station Auditing***

Josh Benaloh, Microsoft Research

Josh Benaloh presented Microsoft Research work on allowing voters to verify that their votes are being cast as they intend. Benaloh observed that attendees at VoComp (University Voting Systems Competition) in July 2007 were confused about the verifiability of all systems except the DRE. It wasn't verifiable, Benaloh noted, but they trusted it because they understood or recognized it. This raised the challenge: Can an open-audit voting system be built with full end-to-end verifiability without trusting the software and look like a DRE? Benaloh claimed that it can be done, but the details are difficult to implement.

Benaloh reported that we have solid protocols for taking a set of encrypted ballots and verifiably processing them to produce an accurate tally. The transformation of encrypted ballots to a tally is a black-box process that requires no trust in software, hardware, or people. Thus, we can solve the "counted as cast" problem. The "cast as intended" problem is not so easy. Voters need to be able to ensure that machines are allowing them to cast ballots as intended. Clever ideas recently developed forcibly engage voters in the verification process, but these are mostly still too cumbersome for users. What happens if voters are allowed but not required to check ballot validity? The principal requirement is that a voting device cannot know the identity of the user. The user of the voting device need not even be a qualified voter; it could be an election official or a suspicious voter.

Benaloh explained a first effort where voting devices are isolated, with stand-alone units charged with capturing a voter's intentions and turning those intentions into encrypted ballots. The voter can either cast this encrypted ballot or have it decrypted to verify that it has been properly formed. As long as the selection of which ballots will be challenged is unpredictable by voting devices, it takes very few challenges to obtain extremely high confidence. With lots of voters, even a small percentage of voters who challenge their ballots results in very high confidence. Some problems remain with this scheme, but Benaloh reported that it looks promising. The scheme is this: A voter walks into a poll station and (if legally required) provides ID to a poll worker. The voter receives a token indicating the correct ballot type. The voter inserts the token into the voting device and makes selections. The voter receives an encrypted ballot. At this point, the voter can (1) provide the ballot to be cast or (2) have the device open the ballot via challenge. This should be unobtrusive. After the selections are made, a voter can be asked, "Do you wish to cast this vote?" If the voter chooses "yes," the device digitally signs the encrypted ballot to indicate its eligibility for casting, and the voter is instructed to take the vote to a poll worker; the poll worker scans the encrypted vote and gives the voter the original as a receipt. If the voter selects "no," the device provides a verifiable decryption which the voter may take home.

The encryption is deterministic. Encryption of ballot b is performed by selecting a random value r and forming the encryption $V = E(b,r)$. The voting device reveals a vote V by revealing b and r; a voter can take this home and verify it on his or her own computer. Most voters probably won't, but at least they can. Ballot protection is ensured by printing the encrypted ballot before the voter indicates whether or not it is to be challenged, but the specifics of the encrypted ballot cannot be known to the voter before the choice is made, in order to avoid voter coercion. Chain-voting is still possible within this scheme and needs to be addressed. Remote voting, however, has very substantial coercion problems, but many people want it anyway. Benaloh reported that the proposed scheme could even be supported in Internet voting. The bottom line, according to Benaloh, is this: Adding a single question to the end of the voter process can add verifiability.

Someone asked whether the system could still cheat by decrypting "incorrectly." Benaloh replied that you cannot prove that the machine is recording incorrectly. You need to be able to check in real time, online. Then someone asked how the voter can tell that the number r is really random. Benaloh answered that the voter can't know this, and that I can't prove privacy, but neither can you. In response to whether there can be any protection against coercion, Benaloh answered, "No, you can't prove absolute privacy, just take good steps."

■ ***Bare-Handed Electronic Voting with Pre-processing***

Ben Riva and Amnon Ta-Shma, Tel-Aviv University

Amnon Ta-Shma presented an end-to-end scheme for election verification that is intended to allow voters to vote "bare-handed," that is, without bringing their computers to the voting booth. This should ideally be as simple as a DRE but with cryptographic guarantees. Ta-Shma reviewed previous work from Chaum and Neff and indicated that neither of those schemes provides the voter with privacy against the booth or the encryptor. This is a primary goal that the team from Tel-Aviv University is seeking to meet; the voter should be able to prepare his or her own ballot, without having to trust anybody else, while still being able to vote bare-handed. This leaves a quandary: If voters prepare their votes at the voting booth, they must bring a computer, but if they prepare their votes at home, they can be subject to coercion.

The authors' work seeks to avoid these problems and pitfalls. There are three primary advantages of pre-processing: (1) The voter can use open-source public code; (2) The voter can use any computer hardware; (3) The scheme is coercion-resistant, so a voter can get his or her ballot encrypted by a friend, a government machine, a coercer, or a political party. The voter still gets privacy as long as the party preparing the ballot does not maliciously cooperate with the voting booth. This allows voter to choose what level of privacy is desired.

The protocol works as follows: A voter comes into the booth with a ballot for each candidate and chooses which one to use at the booth. The booth uses a cut-and-choose test to ensure that each voter comes with a ballot for each candidate and that the voter can match ballots to candidates. Each ballot has two sides, a front and a back. On the front side, the ballot is in plain text; on the back, it is encrypted. Both front sides are published. A poll worker randomly chooses one ballot, and its back side is also published. The booth re-encrypts the front side of the remaining ballot twice and prints it, covered with a scratch surface. The voter chooses a candidate from one column and uses the other column for testing the booth. To test the

voter, auditors check that the published back side matches the published front side. Every candidate appears exactly once, and the encryptions on the front side match the candidates on the back side.

The booth needs to re-encrypt in order to prevent a coercer from having full information. If the booth prints the re-encryptions without a scratch surface, vote-buying is possible. For example, a coercer could say, "Vote using a re-encryption that starts with 110 and get $100," forcing a random vote. We want the vote to be independent of the encrypted strings. This scheme: (1) provides unconditional unforgeability even against all-powerful adversaries (common also to other crypto schemes); (2) is receipt-free: outsiders only see the encrypted vote and the revealed column; (3) provides coercion-resistance, because there is a probability of 1/2 to coerce a voter without being caught, which deters large-scale coercion (as there should be a significant risk attached to coercion); and (4) allows the voter to vote barehanded. Moreover, it is a modular scheme, and it can be based on several existing schemes. This protocol transfers the ballot preparation from the booth to the voter (at a preprocessing stage) and the tallying is unchanged. Future directions for this work include simplifying the scheme and relaxing the assumptions—mainly the assumption that the public board is readable from anywhere.

The first questioner observed that at VoComp, most people thought cryptography was a study of where dead people go (crypts). The population won't understand this if it involves cryptography. Ta-Shma responded that they ask the voter to come in with two ballots. The voter doesn't have to understand crypto; only the auditors do. Someone pointed out that there are already variants that deal with booth-trust issues. Ta-Shma agreed, but said that their contribution is the use of multiple ballots. Someone asked what happens if the poll worker colludes with a coercer. Ta-Shma conceded that the scheme fails in that case. Finally, someone expressed this concern about the crypto process: The public is the auditor. Ta-Sham answered that the crypto behind the scheme is very simple and easy to understand. All you have to do is calculate some function, and the software to do it can be downloaded from the Internet.

■ ***Three Voting Protocols: ThreeBallot, VAV, and Twin***

Ronald L. Rivest, Massachusetts Institute of Technology; Warren D. Smith, Center for Range Voting

Ron Rivest presented some "outside of the box" ideas on end-to-end voting systems, which he claimed were a most promising general direction for election verification. The most common end-to-end systems are based on cryptography, but he was able to present three options for doing it without crypto: ThreeBallot, VAV, and Twin.

In ThreeBallot, each voter casts three plain text ballots. All three go on a public bulletin board (PBB). The voter takes home a copy of an arbitrarily chosen one as a receipt. It doesn't indicate how he or she voted, but serves as an integrity check on the PBB. Every ballot has a serial number that is not easy to remember but is easy to type. Each row of a ballot has at least 1 mark, not 0 and not all 3. Each candidate gets n extra votes (where n = number of voters) but the election outcome is the same. This works for everything except rank-order choices or write-ins. Votes are cast in a physical ballot box. The order of ballots is random and is not tied together, but a machine checks before casting to ensure that ballots are valid: It only checks, it doesn't tally. The voter arbitrarily gets to choose one as a receipt, and no record is kept of which was the receipt. Receipts should be unforgeable. The voter confirms the posted ballot on the PBB after the polls close. Each ballot has a unique ID, so it can be located. Voters should not see (and/or be able to memorize) IDs for ballots that were not copied (to prevent vote-selling.) Plain-text ballots are subject to short ballot requirements, to prevent reconstruction attacks. Since an attacker doesn't know which ballots posted on the PBB have copied receipts, any significant tampering is likely to be detectable. The use of three ballots makes this coercion-free. Voters can't sell their votes by using their receipts. Using only the PBB and voter receipts, neither an adversary nor a voter can determine which three ballots were in an original triple. However, the usability is not so good, and the system is confusing to many. It would be possible to mix "OneBallot" (ordinary ballots) with ThreeBallot, but no receipts could be issued. End-to-end security provides voter confidence; the voter can check that his or her ballot is included in the tally and can check that collection and tallying are done correctly, all without crypto.

Rivest next presented VAV (Vote/Anti-Vote/Vote), in which the voter casts three ballots and takes a copy of one home as a receipt, but one ballot must cancel another. The anti-vote ballot is marked as "ANTI," so the voter casts one ballot the way he or she wants, another ballot the way he or she doesn't want, and an anti-vote to cancel the unwanted one. The tallier finds and removes pairs of ballots that cancel one another and only counts the remainders. This handles any voting system.

In Rivest's final scheme, known as Twin, the voter gets to take home a copy of somebody else's ballot. The voter can verify it from the PBB. All original ballots are put into a bin as they are cast, and every voter (after the first ten) is given a copy of an arbitrary ballot from the bin. Voters cannot prove their own ballot and don't know whose ballot they have. An attacker cannot collect all copies of any chosen receipt, because receipts are given with random selection, using replacement. A constant fraction of all receipts are taken home with high probability. Rivest concluded that it is possible to implement end-to-end security without crypto, and end-to-end schemes provide improved assurance of correctness of an election outcome.

The first question was about how to get an unforgeable receipt without crypto. Rivest answered, "Maybe watermarked paper? A digital signature?" The next person pointed out that, with VAV, people are more likely to take home the one that's the actual ballot. Rivest explained that

this is the same in other schemes, but the voter has deniability here, which helps prevent coercion. The next questioner had a stumper: What prevents voters from casting two ballots for the candidates of their choice and then an Anti-Vote ballot against the candidates they oppose? Rivest confirmed that there isn't a solution to that problem yet. Someone else wondered, if ballots are posted in plain text, what prevents stray marks from allowing vote-selling? Rivest said that the posted records should be digital versions of the plain text. Finally, someone pointed out that posting ballots on a bulletin board is new. What new opportunities does this present for wholesale fraud? Rivest answered that new kinds of verifiability will help in detecting and preventing wholesale fraud. This is a whole new layer of defense.

First USENIX Workshop on Offensive Technologies (WOOT '07)

Boston, MA
August 6, 2007

Summarized by Dominic Spill (dominicgs@gmail.com) and Robert N.M. Watson (robert.watson@cl.cam.ac.uk)

The First USENIX Workshop on Offensive Technologies was opened by Tal Garfinkel. He thanked the program committee and USENIX and gave an overview of what to expect from the workshop.

INVITED TALK

■ ***Fast-Flux DNS and Overlay Networks Using Botnets***

David Dagon, Georgia Institute of Technology

David got the workshop off to a start with a discussion of his current work on botnets, focusing on botnets that rapidly change their DNS responses to avoid detection.

In botnets, which date back to the 1990s, pieces of malicious code (bots) are often spread using Web sites, email, and vulnerabilities. The bots originally communicated with their controller using IRC channels. The countermeasure to this was based on the DNS requests made to find the IRC channel. As these countermeasures were used the sophistication of the botnets increased, and they began using peer-to-peer applications for communication and replication. These botnets were stopped because they had fixed points in their network, and these could be tracked and stopped.

The current generation of botnets use domain names for which the DNS response changes rapidly, with different bots within the network taking the role of server for the others. This is known as Fast-Flux DNS. The botnets avoid having fixed servers and therefore attempt to avoid being shut down; they use themselves to respond to DNS requests and propagate themselves.

These botnets are then used for a number of different applications. Two of the most well known are sending unsolicited email and distributed denial of service attacks. The botnet provides a platform for these applications, which is sold as a service by the controller of the botnet.

David has investigated the locations of the bots in the networks, using IP addresses. He found that most of the bots were in centers of population, where the use of broadband Internet is greatest. So David mapped the growth of the botnets using the IPs that were given in response to repeated DNS requests. These graphs showed that initially only a very small number of IPs were returned, but shortly afterward there was an explosion in the number of IPs returned, as more systems were infected by the bots.

The reason these botnets persist is that they avoid detection by constantly changing, either by repacking the binary or by downloading an updated version from another bot in the network. They are often observed, by antivirus researchers, within virtual machines, but there is a large amount of research into detecting virtual machines, meaning that as research attempts to stop one exploit it helps botnets continue to go undetected.

David has also analyzed the effectiveness of using existing blacklists and user traffic analysis as predictors of infection, and he observed that although there are troubling social issues associated with usage analysis, based on usage patterns some users are more likely to be exposed to, and hence infected by, malware than others. This prompted a healthy discussion of the interactions between privacy and monitoring in malware prevention.

Robert Watson asked about the feasibility of bots communicating using the Tor network or botnets providing stronger anonymity services to protect their maintainers. David said that there are botnets that do this, but the performance of the Tor network is not high enough for the traffic required by most of the networks. Additionally, the botnet managers may use Tor to control the botnets, but the only anonymity they care about is their own, not that of the systems that they have exploited, so they would not use Tor to hide the systems with bots.

FROM THE METAL TO THE INFRASTRUCTURE

Niels Provos chaired this session.

■ ***Flayer: Exposing Application Internals***

Will Drewry and Tavis Ormandy, Google, Inc.

Will presented work on an advanced fuzzing tool, based on Valgrind, with the ability to taint input and skip over checks in the code. He also showed some of the bugs that it had uncovered in libtiff, openssl, and openssh.

The Flayer tool is a combination of a fuzzer, an auditing tool, and a patch analyzer. It can be used in automated testing scripts or as a stand-alone application. Flayer taints input to an application to allow it to be tracked through the execution of the code. It can also bypass the execution of branches to avoid version checks.

The goal of the tool is to find errors in the code of an application without having to get into the depths of how it works. It can be used with /dev/urandom and also files filled with random data, and it can be beneficial to run the tool with a file of random data and then alter a small amount, as little as one bit, before running it again. So far it has been used to find bugs in libtiff, openssl, and openssh. The tool and source code can be found at http://code.google.com/p/flayer.

Niels Provos asked about the degree of automation, noting that it seemed the process of using Flayer was very manual. Will answered that this is a user-assisted analysis tool, relying on the insights of the user into the code. However, it is significantly less manual than tools that require, for example, coming up with a complete specification of correct behavior to generate fuzzing input.

When asked about performance, Will said that he had measured it to be roughly 20 times slower than the execution of the code, which is faster than other debugging applications; it also has high memory usage, which is one of the factors they are currently trying to reduce.

■ ***The ND2DB Attack: Database Content Extraction Using Timing Attacks on the Indexing Algorithms***

Ariel Futoransky, Damián Saura, and Ariel Waissbein, Core Security Technologies

Ariel presented a new attack technique for extracting data from databases using timing attacks. This attack relies on the variable cost of inserting fields into sorted tables based on the I/O cost of b-tree node splitting, assuming that all keys are unique. This service may be available to anonymous users even if select queries are not, and it offers significant efficiency improvements over a simple but computationally infeasible $O(n)$ search. David analyzed the performance characteristics of the MySQL database using the InnoDB storage format on Windows XP, reporting experimental results in which 64-bit keys in the table were extracted in tens of thousands of insert queries.

Tal Garfinkel asked whether this attack could be performed through Web-based front ends to databases, rather than via direct database queries; Ariel answered that they had not experimented with this yet and it would introduce some amount of noise, but if it didn't introduce disk I/Os this effect may be small compared to the cost of node splitting. A general discussion of the effects of noise on this and other timing attacks ensued.

■ ***Exploiting Concurrency Vulnerabilities in System Call Wrappers***

Robert N.M. Watson, Computer Laboratory, University of Cambridge

Robert was at WOOT to unveil his finding on the vulnerabilities introduced into a system by the use of system call wrappers. He explained how the wrappers function and how they are exploited and gave recommendations as to how the problems can be addressed by the operating system authors. He also showed examples of exploits for single and multiple CPU systems.

System call wrappers are used to increase the portability of applications without needing to recompile them on every target system or for systems that may not have kernel source code available. Many applications use them, notably antivirus tools. Robert gave a comparison to resource managers but said that system call wrappers are not atomic with respect to the system call, and this is where the vulnerability is introduced.

Robert demonstrated the existence of two new types of race condition, introduced by system call wrappers, which can be exploited in addition to the well-known "Time of check to time of use" race. "Time of audit to time of use" and "time of replacement to time of use" are the names given to these new race conditions. He had also written example code to exploit these and other conditions for both uniprocessor and multiprocessor systems, showing that Systrace and GWSTK, two commonly used wrapper toolkits, were both vulnerable.

A number of solutions were proposed, such as additional memory synchronization; however, this was shown to introduce more vulnerabilities. Robert recommends moving toward message passing, a move already made by Linux Security Modules (LSM) and the TrustedBSD MAC framework. With this work Robert has shown that system call wrappers are a threat to the security of any system that uses them, and he calls on developers to change their practices.

■ ***Billing Attacks on SIP-Based VoIP Systems***

Ruishan Zhang, Xinyuan Wang, Xiaohui Yang, and Xuxian Jiang, George Mason University

Xinyuan Wang discussed research that he has been doing in the area of SIP-based VoIP applications, especially the billing systems used. This is important because SIP is now the standard used by most VoIP systems, including those sold by Vonage, AT&T, and Verizon. The number of VoIP users is expected to reach 44 million by 2010, all of whom will want their billing to be accurate.

The call setup procedure is similar to that used for non-VoIP telephone calls. To make a call the caller sends an INVITE message, which is replied to by an OK message from the server, and the server then contacts the recipient of the call. However, once the call is answered the two parties communicate directly. The caller or the recipient must then tell the server that the call has ended so that the server can calculate billing, and this is done with the BYE message. If the recipient is already using the phone a BUSY message is sent to the caller.

Four parts of the call setup and tear-down process were identified as vulnerable to man-in-the-middle attacks: replaying the INVITE message, fake BUSY responses, delayed BYE messages, and dropping BYE messages. Replaying the INVITE message will allow an attacker to create a second

call with the cost being charged to the original caller, assuming the attacker can alter the destination IP of the call. The fake BUSY message attack involves two man-in-the-middle attacks, allowing the two attackers to communicate at the expense of the caller. The likelihood of this attack is low as the two attackers must be online and know that they wish to communicate, so they could simply connect directly.

Xinyuan rounded off the presentation with some suggestions on preventing these attacks. The INVITE replay attack could be prevented with a nonce; this is done by some providers, but others allow replay after a week. To prevent the fake BUSY attack the messages need integrity protection, so that they cannot be altered by an attacker. The group is going to continue research in this area, including using these attacks to consume resources on the server of the service provider.

SNIFFING AND SCANNING

- ***BlueSniff: Eve Meets Alice and Bluetooth***

  Dominic Spill and Andrea Bittau, University College London

Dominic presented his work to produce an affordable Bluetooth sniffing device using easily available hardware. This combines existing and new techniques for breaking Bluetooth encryption and works with the GNU Radio software signal processing framework. The resulting device is affordable (approximately US$700 vs. commercial products at $10,000).

Dominic began his talk by summarizing the Bluetooth protocol, in which master and slave devices build associations based on unique MAC addresses and internal clocks. Bluetooth whitens (scrambles) packets by XORing pseudo-random values with the data stream based on the lower six bits of the internal clock, as well as using frequency hopping spread spectrum (FHSS) at 1600 hops/second chosen using the MAC address and master device's clock, in order to minimize interference.

BlueSniff must determine the MAC address and clock value in order to determine the hopping pattern and whitening sequence. Only a portion of the MAC appears in the packet, although frequently the remainder may be guessed, or relatively easily brute-forced by using the CRC to test candidate values in a single packet.

The primary limitation of this work is that more than one physical radio must be used to track all available Bluetooth channels; the current hardware can monitor only one-eighth of the frequency space.

Tal Garfinkel asked what support plans exist for the tool in the future. Dominic answered that it depended on future deployment, but if time was available, allowing eight devices to be used at once is an important next step. Niels Provos asked about the research impact of this work. Dominic answered that the lack of a promiscuous mode for Bluetooth meant that almost all published work on the Bluetooth protocol was purely theoretical and that with easily accessible sniffing the doors would be opened for a great deal more research. Dominic indicated that the current range limit was several meters. In answer to a question about Bluetooth 2.1, Dominic stated that the modulation technique changes, and current GNU modulators are not yet able to handle the new technique.

- ***Toward Undetected Operating System Fingerprinting***

  Lloyd G. Greenwald and Tavaris J. Thomas, LGS Bell Labs Innovations

Lloyd presented an in-depth analysis of operating system fingerprinting techniques utilizing the differences in TCP/IP implementations. Although this is a commonly known technique, Lloyd presented optimizations drawn from the entropy of information provided by the tests. These are useful for identifying the correct tests to run to pin down the exact version of an operating system with the minimum number of packets sent and received, thus reducing the chances of the fingerprinted system detecting the traffic.

The tool used to perform the fingerprinting was based on nmap (http://www.insecure.org), specifically the second-generation operating system fingerprinting system. The database of operating system characteristics was used as provided with the tool and was not modified for the purposes of calculating information entropy.

The result of the analysis was that it was possible to use fewer than the 16 standard packets to identify some operating systems, especially with the knowledge that the majority of systems are Windows-based, and therefore the test that reveals the most about the Windows TCP/IP stack should be used first.

The work makes it much easier to order the tests for operating system fingerprinting to avoid the more detectable tests and reduce the number of packets sent to and received from the system.

- ***Catch Me, If You Can: Evading Network Signatures with Web-based Polymorphic Worms***

  Matt Van Gundy, University of California, Davis; Davide Balzarotti and Giovanni Vigna, University of California, Santa Barbara

Matt presented his work on Web-based polymorphic worms with a focus on those written in scripting languages such as PHP. He identified the key parts of the PHP language that allow the text of a script to change but the overall execution to remain the same, and he showed how this can be used to fool applications that attempt to find software signatures.

He began with an introduction to the two most common applications for calculating software signatures, Polygraph and Hamsa, and an explanation of the techniques they employ. Matt also explained some features of the PHP script-

ing language, such as the ability to store function names in variables and insert random strings, that are ignored by the interpreter. These allowed him to alter a script while keeping the execution constant. PHP also allows for inline use of compression, which was useful for changing the signature of the script, but it reduced the size and therefore the amount that it could be changed.

He then showed how frequently his morphed script was detected by both Hamsa and Polygraph, given that both had been shown the original script and had created a signature for it. For Hamsa, the morphed worm went undetected almost 100% of the time. Polygraph was able to detect the worm more easily, and it had much lower false negative rates, but it was unable to handle the worm at its full size and could only create a signature for a compressed version of the worm.

Matt was asked if this technique could be applied to other Web-based exploits. He said that it could be used to cloak attacks such as remote file injection. When asked how the use of compression affected the results, he said he was confident that the morphed script could beat the Polygraph detection if it allowed the larger worm to be tested.

HIDDEN ATTACKS

- ***An Encrypted Exploit Payload Protocol and Target-Side Scripting Engine***

  Dino A. Dai Zovi, Two Sigma Investments

Dino presented his work on exploit payloads for mobile clients, introducing a three-stage payload that creates a connection back to a server, downloads an execution environment, and then exploits the host system.

A widely accepted network security model is that there exists a boundary between internal and external hosts, but this is often an oversimplification. Mobile clients may be connected to many different networks, each of which could have different levels of security and allow the systems to be exploited. Dino has produced a payload for these systems that allows them to exploit a network once the client returns.

The payload initially establishes a secure connection to a server in order to download the rest of the payload. This first stage needs to be small: in this implementation it was approximately 1200 bytes. The second stage uses this secure connection to download an execution environment for the Lua scripting language to run the main attack.

The final stage is the attack within the network. It is written in Lua to allow portability, and Lua environments often simply wrap system calls, giving the script a large amount of execution scope. The Lua scripting environment allows the payload access to many common protocols such as HTTP or FTP, allowing the malicious code to spread itself further through the internal "protected" network.

There were some suggestions as to how the payload could be further streamlined by only retrieving a Lua bytecode interpreter rather than the entire execution environment, and there was a discussion on using this proof-of-concept payload with real-world attacks.

- ***Exploiting Redundancy in Natural Language to Penetrate Bayesian Spam Filters***

  Christoph Karlberger, Günther Bayler, Christopher Kruegel, and Engin Kirda, Secure Systems Lab, Technical University Vienna

Christoph presented one of the first papers designed to assist spam producers. He was attempting to pass Bayesian spam filters by reducing the spam score of a message using word lists to replace key words. The resulting messages were run through common spam filtering software to assess the effectiveness of the word replacement.

The most common method for detection of unsolicited email currently is through the use of Bayesian filters. These give each piece of email a score based on factors such as content and sender identification. The filters rate the spam value of each word, and this rating is used to determine whether a message should be filtered.

The principal idea of the work was to use the rating of words to choose alternatives to high-scoring spam words. Using the word lists from WordNet (wordnet.princeton.edu), the high-scoring spam words were replaced; this took some analysis of the grammar to allow for words with multiple meanings.

The resulting email messages were run through spamassassin and dspam filters, giving significantly lower scores for messages that had passed through the word replacement software. The results showed that although this technique helps the messages to achieve a lower score, the inclusion of URLs in most spam messages will still allow the filters to reject the message.

FIVE MINUTE MADNESS

The workshop concluded with a session entitled "Five Minute Madness," an opportunity for brief descriptions of work in progress and suggestions for potential research and ideas based on the works presented. The talks were mostly based on potential attacks using botnets.

First up was a suggestion that blind SQL injection attacks could be made less detectable by using the many nodes of a botnet to reveal table information. The principle of this was to use the distributed power of a botnet to increase the size of the server logs and make tracing such attacks much more difficult; it would also make automated detection more difficult, as no single host would be extracting an entire database entry.

This was followed by an anti-spam-filter technique derived from ASCII art. The idea was to convert the intended mes-

sage to an image and then represent this using ASCII art. The ASCII art could be based on real words that are not considered to be associated with spam and therefore bypass most common spam filters.

Next came two suggestions for finding better uses for botnets, particularly attacks that we have not seen. These ranged from a distributed attempt to find private keys of large organizations, to capturing audio and video on home or office systems for blackmail or fraud purposes. A brief debate followed about physical security between a user and his or her own system, with regard to protecting the user from the system.

The session was rounded off by a suggestion that DNS traffic should be monitored, because the first lookup request for a domain, for example a botnet distribution server or a phishing site, will come from the person who set it up testing their work. This was disputed by some, suggesting that most botnet controllers would route the traffic through the botnet or Tor network to hide their tracks.

MetriCon 2.0: Second Workshop on Security Metrics

Boston, MA
August 7, 2007
Summarized by Dan Geer

MetriCon 2.0 was held on August 7, 2007, as a single all-day, limited-attendance workshop, in conjunction with the USENIX Association's Security Symposium in Boston, Massachusetts. MetriCon 2.0 was the second meeting with this name and topic, the first having been held a year before in Vancouver. The self-selected organizing committee was co-chaired by Betsy Nichols (PlexLogic) and Gunnar Peterson (Artec Group). Also on that committee were Fred Cohen (Fred Cohen & Associates), Jeremy Epstein (Software AG), Dan Geer (Geer Risk Services), Andrew Jaquith (Yankee Group), and Russell Cameron Thomas (Meritology). Dan Geer is the principal author of these notes and assumes full responsibility for any inadvertent reporting errors. The agenda and presentation slides can be seen at http://www.securitymetrics.org/content/Wiki.jsp?page=Metricon2.0.

Seventy-three people attended (compared to forty-four at MetriCon 1.0), predominantly representing industry (62) rather than academia (5) or government (6) (comparable numbers for MetriCon 1.0 were 30, 10, and 4). The meeting lasted from 08:30 until something after 21:00, with meals taken in-room, so as to maximize output—as may be reflected below.

This second such event could perhaps have benefited from more meeting time, but it is likely there will be another and, in comparing this one to the last, the amount of progress is best gauged by the sharp change from "I plan to . . ." toward "I tried this and it turned out that . . ."—which you are invited to consider a metric on MetriCon.

KEYNOTE "DEBATE"—DO METRICS MATTER?

This was not so much a debate as a point-counterpoint from two keen observers.

METRICS DO MATTER

Andrew Jaquith (Yankee Group), describing himself as Dudley Doright, simply went straight to a list of "ten" reasons why metrics matter:

1. Metrics quantify the otherwise unquantifiable.
2. Metrics can show trends and trends matter more than measurements do.
3. Metrics can show if we are doing a good job.
4. Metrics can show if we are doing a bad job.
5. Metrics can show if you have no idea where you are.
6. Metrics build bridges to managers.
7. Metrics allow cross-sectional comparisons.
8. Metrics establish where "You are here" really is.
9. Metrics set targets.
10. Metrics benchmark yourself against the opposition.
11. Metrics create curiosity.

METRICS DO NOT MATTER

Not to be outdone, Mike Rothman (SecurityIncite) started by reminding us all that it is (way) too easy to count things for no purpose other than to count them. He wanted us all to "Stop thinking like a security person, or all this metrics stuff will be a waste; you cannot measure security, so stop trying." This means that you measure, if you measure at all, not just to measure for the purpose of satisfying the counting instinct, but to make a difference. Rothman's own list of what matters includes:

1. Maintenance of availability
2. Preservation of wealth
3. Limitation on corporate liability
4. Compliance
5. Shepherding the corporate brand

Rothman went on to say, "Who cares what Jaquith's (separately published but widely quoted) 'five characteristics of a good metric' are when we already know that Rothman's own list is what really matters?"

With that, Betsy Nichols (PlexLogic) exercised her role as moderator by calling on the audience to ask questions.

DISCUSSION

First up was a suggestion that there are, in fact, metrics that speak to what Rothman was talking about, such as Apdex. Rothman answered with a question of sorts: If you don't have time to burn, then shouldn't you actually be careful what it is you are measuring? Once made, using the results of measurement takes time, but measurement for no purpose is way too easy, making useless work for

yourself and others. Plus, once you start measuring something and incorporate it into the culture of a firm, you will find it harder to stop measuring whatever it is than to have started measuring it in the first place.

Another questioner asked whether to start large or small and whether to risk too much ambition or too little. Rothman took that one as well and reminded us that unless you, the measurer, are seen as a colleague you will be seen as a crank, something he characterized as "making a deposit in the incredibility bank."

Another questioner asked, "Is it not true that metrics only matter when something can be said to be under at least a modicum of control?" Put differently, what good are metrics in a hurricane? Rothman put a new spin on the aphorism that the sum of beauty plus brains is a constant by suggesting that if all a metric does is make you look good, then it has already contributed all the value it ever will.

TRACK 1—GUNNAR PETERSON, TRACK CHAIR

- ***Security Meta Metrics—Measuring Agility, Learning, and Unintended Consequence***

  Russell Cameron Thomas, Meritology

Thomas began by reminding us of the great difficulty of our field: the mutation rate, which, of course, translates into a challenge to continuously learn. That challenge leads to his thesis that meta-metrics, the measurement of whether we are rightly measuring the right thing, is essential, as the learning demand will not recede. More fully, it is learning, agility, and unintended consequences upon which he wants to focus. Thomas distinguished single-loop learning, a control structure with a defined outcome, from double-loop learning, which adjusts the single loop's outcome. This has direct connection to the balanced scorecard idea as found in management schools.

In distinguishing puzzles, problems that have a solution, from mysteries, problems that may have no solution, Thomas suggested that meta-metrics studies focus on the latter through coverage metrics, decision effectiveness metrics, and investment return metrics. Agility meta-metrics (e.g., "Are we learning fast enough?") is richly studied in other fields, but it can be summarized here as meta-metrics for speed (such as the time between "sense" and "respond"), cost, error, and maximum response capability. Rounding out the suite is meta-metrics for discovering and mitigating unintended consequences, including familiar items such as blame shifting and excessive risk aversion, detecting the existence of these unintended consequences, measuring their significance and cost, and scoring their perversity. Thomas's bottom line is that unless your enterprise is small, simple, and static, you need at least one metric for each of learning, agility, and unintended consequences.

A questioner raised the possibility of studying latency in the agility domain with Fourier analysis. As to "Who is doing this learning?" Thomas suggested that it be the enterprise risk team, not individual employees. As to the problem of indirect costs, Thomas referred the questioner to the "total cost" section of Thomas's Web site. The idea of "malicious compliance" came up (e.g., Accounting saying, "Security is important but costs must decline"). Thomas suggested that the most common finding for the root cause of a disaster is that of a "failure of imagination."

- ***Security Metrics in Practice: Development of a Security Metric System to Rate Enterprise Software***

  Fredrick DeQuan Lee and Brian Chess, Fortify

Lee described the "Java Open Review" during which the Fortify team examined 130+ open-source projects for both quality and security defects. Given that many of these projects overlap to some degree in function, this examination naturally led to the question of which project is better.

That question is, even given this work, unsolved, as the downstream risk is dependent on deployment context as well as the existence of defects. In their estimation, risk assessments need an enumeration of either threats, vulnerabilities, and controls or event probability and asset value and, given that static analysis only uncovers vulnerabilities, it cannot yield a risk metric.

Static analysis can, however, measure defects in source code and benchmark software components, use objective and repeatable measures to improve software over time, and feed into any existing risk management system. The Fortify SCA product used in this work can provide most of the base information for a CVSS score, as well as code volume, cyclomatic complexity (per function), and defect densities along several axes.

Fortify's customers, as do perhaps all metrics end consumers, want condensed thumbs-up/thumbs-down views, and Fortify chose to copy the mutual fund star system (much as did the OWASP group). Those stars are:

* No remote/SETUID vulnerabilities
** No obvious reliability issues
*** Follows best practices
**** Documented secure development process
***** Passed independent security review

Lee points out that this rating system is not without flaws: it is harsh, there is some subjectivity, and the introduction of a tiering forces some compromises because of inexact ordering. Nevertheless, such a scheme can be directly used as screening criteria (e.g., "Show me 2-star, mid-size, shopping cart software"), a comparator (e.g., "How does this set of 1-star components compare?"), or, as described earlier, as an input among many to an existing risk management model, if any. Going forward, it will be important to validate this method against the closed-source world and to compare this method's hard numbers to (the accumulation over time of) security auditors' reports.

The direct question of "Are there any open source projects with a nonzero number of stars?" revealed a few (e.g., Tomcat). The similarly expected question, "How do you handle false positives?" was that people remain essential to this. One observer noted that as new attacks appear old ratings lose meaning, which Lee said had no solution other than to say that as of such-and-such a date the rating was X and to retain in a public fashion the rule set that was in use as of that date. Some questions on consistency and rigor were raised, but the truthful answer is that they were early, though Lee did point out that reranking old work with successive new rule sets would shed some light on the consistency questions (over time).

■ *A Software Security Risk Classification System*

Eric Dalci and Robert Hines, Cigital

Dalci described the purpose of the Risk Classification System (RCS) as estimating an application's potential risk with respect to other systems in the portfolio and determining what SLDC actions to require for given risk levels. This would yield, as RCS outcomes, the ability to prioritize (impose an ordinal scale) and an indication of where midcourse corrections in ongoing development should go. As with all efforts to summarize risk, there are separate foci on business risk and technical risk.

In producing the RCS, Dalci and Cigital dropped cyclomatic complexity (because it was not clear how to correct for language differences), process-related metrics (since organizations rarely are internally consistent in how they apply security processes), and generally any factors that contribute expensive or squirrelly answers. Roughly speaking, their strategy involves weighted aggregation of various measurable characteristics and then use of the weighted sum as a score for portfolio segregation. Dalci listed the systems that tended to have a high score as:

- Independent security review systems
- Web-facing systems
- Large code-size applications
- Complex applications
- New applications

and those with a low score as:

- Low user count and/or internal applications
- Low corollary (downstream) impacts
- Small code-size applications

His slides displayed the weights used and the correlation achieved with aggregate scores.

In response to a question, Dalci clarified that no dependent downstream applications would be scored as low, while more than four such downstream applications would be judged as high. Another audience member suggested that adapting data gathering to the measurement system sounded consistent with Thomas's double-loop learning construct. Dalci confirmed that the aim of this effort was that the method be fast and light. He also described the correlation figures as essentially a measure of cascade failure.

Another questioner suggested that the business and technical risk views would be good to summarize as a 2x2 table. In response to a question as to where revenue factors in here, Dalci said that that is a subject for future work. Another respondent suggested that using Dalci's method to get a probability of failure makes this similar in style to a credit risk score.

TRACK 2—JEREMY EPSTEIN, TRACK CHAIR

■ *Web Application Security Metrics*

Jeremiah Grossman, WhiteHat Security

Grossman stated his bias with respect to security metrics, namely that bad things are generally unmeasurable. As of today, there are 128 million Web sites and these sites are accessible to 1 billion people. We will all acknowledge that a percentage of these can be hacked and that when hacked there are consequences. Grossman's study looked at the composite outcome of 20 months of weekly remote black-box assessment of hundreds of the largest and most popular Web sites (in all sectors), all of which are custom Web applications without well-known issues. The threat classification from the Web Application Security Consortium (WASC) was used as the baseline. His results are that 7 of 10 Web sites have "serious" vulnerabilities, and he assessed the likelihood that a Web site has a vulnerability of a given severity.

Grossman went on to say that, putting aside infrastructural matters such as PHP, cross-site forgery remains very difficult to scan for, and new ways to evade XSS filters keep showing up. HTTP response splitting is, he believes, the coming thing and must be watched carefully. He provided a number of looks at what his data shows, such as cross-tabulating the kinds of flaws found with their severity, ranking the filename extensions most involved, and showing that the kinds of flaws present do vary by industry vertically.

Perhaps more hopefully, the custom Web applications that are more secure come from development environments where the security configs are actually turned on, have a software development life cycle that does include security in a formal way, and prioritize the remediation of vulnerabilities in a rational fashion. Looking ahead, Grossman particularly wants to continue comparisons across verticals and technology and examine the rate at which problems reappear.

A questioner asked whether one can include Web site complexity or size in the vulnerability rankings; Grossman does not believe that complexity is related to security: Security comes from the code being beat on. Another thought Grossman's SQL injection numbers were low, and Grossman confirmed that they could be hiding issues in that space. Grossman did not yet have prevalence by platform data but it is coming, and he will also be introducing trending. A hard problem is in environments where part-

ner Web sites function as an apparent whole; much work needs to be done on how to characterize the risk in such settings.

- ***Operational Security Risk Metrics: Definitions, Calculations, and Visualizations***

  Brian Laing, Mike Lloyd, and Alain Mayer, Redseal Systems

Mayer's work includes many graphics aimed at making objective operational security metrics and visualizing them in ways that make for real communication. One part of his visuals shows tracing a network path through a set of servers from the outside (Internet) to a DMZ to an internal host that becomes compromised, thus leading to a general compromise. With that as a lead-in, Mayer stated the goals and nongoals for a metrics program. The main idea is that hierarchies are natural, that cascade failure is their downside feature, and thus that drill-down for root cause analysis is a high-value capability. Mayer suggests treemaps as a well-matched tool for this. To illustrate this point, the reader will have to consult his materials, as they are visually rich. Mayer's main point, and one on which he was questioned as well, is that treemaps are effective in conjunction with more traditional topologic visualization, but that some people take to treemaps immediately and some do not.

Mayer called his metrics "opinion-based math" and wondered about the absence of user-side pushback—do the users get it, or do they not? Nevertheless, mapping of cascade failure to the hierarchies in which they occur with a drill-down-friendly visual summarization does seem to be an advance.

A questioner asked whether using absolute-risk or delta-risk is better. Mayer said that delta-risk might be more informative. Another questioner asked whether this might be aggregated over industrial verticals, which Mayer acknowledged but thought to be too early. Mayer responded to "Where does the source data come from anyhow?" by suggesting that firewall configuration files and scan data suffice.

- ***Metrics for Network Security Using Attack Graphs: A Position Paper***

  Anoop Singha, NIST; Lingyu Wang and Sushil Jajodia, Center for Secure Information Systems, George Mason University

Singhal described his group's motivation by contrasting the typical qualitative questions about a database's security (e.g., "Is that server secure from intruders?") with the quantitative questions that are actually needed (e.g., "How secure is that server?"). He sees the challenge as one of composing a variety of measures into one metric.

He focused on attack graphs, annotated with both point probabilities (of exploit of a given flaw) and cascade probabilities (of reaching through this flaw to the next host, beginning with an attacker at node 0). The point is that such graphs can make clear the value returned in hardening (blocking the exploit of) any given node in such a graph. Singhal suggests that such mechanisms of analysis are common-sensical and can be generalized, which he proposes to do as further research. Questioners asked about the level of effort required to set the probabilities in such graphs, whether vulnerabilities were statistically independent, whether this was scalable, and how it meshed with business needs.

TRACK 3—ADAM SHOSTACK, TRACK CHAIR

- ***Software Security Weakness Scoring***

  Chris Wysopal, Veracode

The purpose of Wysopal's work is to develop a standardized set of software security analysis techniques addressing inter-rater and test-retest reliability, and with actionable outcomes. Wysopal's method builds on what is available at the outset, the Common Weakness Enumeration (CWE) and the Common Vulnerability Scoring System (CVSS), noting that all current techniques have serious levels of false positives and false negatives.

Wysopal's method is layered and should be looked at in the original, with the logical outcome of being able to rank weaknesses in the sense of "How likely is it that bad things will come from this weakness?" The ranking is thus a contributor to security decision-making, and the metric proposed is thus well worth further effort.

Wysopal suggests that the CVSS Environmental Score can be used unchanged, although, of course, this implies foreknowledge of the deployment environment into which software will go. He further suggests some plausible goals:

- Standardized false positive rate testing
- Possible use of data and control flow between taint source and weakness
- Addition of false negative rates, moving from "badness" score to "goodness"
- Empirical field testing

Questioners asked which version of CVSS Wysopal was using (version 2) and whether the appearance of new attacks would change the risk scores he computed. Wysopal thought that the appearance of new attack methods was likely a research-grade problem at this time.

- ***Developing Secure Applications with Metrics in Mind***

  Thomas Heyman, Christophe Huygens, and Wouter Joosen, K.U. Leuven

Building on their work presented at Metricon 1.0, Heyman et al. set out to answer, "How secure is my application?" In their prior work, a "pattern" is the observable connection between the core of one's computing environment and the ecosystem in which it lives, leading to ratio scores such as the number of firewall invocations versus the number of service invocations, or the number of guards versus the number of access points for each component. With this new work, they are trying to use patterns to piggy-back security metrics into applications.

In this case, domain-specific security requirements are assigned domain-independent security objectives, and design involves composing systems from primitives, such as Accountability through Authentication plus either Auditing or Non-Repudiation, and, in turn, Auditing through both an operational interceptor and a secure logging facility. Just as the building blocks are composed into the final system, the measurements that come with each building block are rolled up into a final metric. As in the aphorism "A chain is only as strong as its weakest link," this roll-up process will propagate minimum values upward, such as if Auditing decomposes into both an operational interceptor and a secure logging facility; whichever of those two is the least reliable will determine the reliability of the Auditing function.

Heyman expects a proof of concept where sensitivity analysis can be done on the dependency graph and, perhaps, to automate the integration of metrics into the code base of the building blocks. Questions went right to the hard parts, such as "Where might the numbers come from?" Heyman said they are assigned heuristically and can be thought of as relative capabilities. Confidence scores seem eventually possible, as would sensitivity analysis. Although multidimensional methods are not in place now, they may be necessary if risk is taken into account.

- ***Correlating Automated Static Analysis Alert Density to Reported Vulnerabilities in Sendmail***

  Michael Gegick and Laurie Williams, North Carolina State University

The security metrics arena has many parallels to the field of reliability, such as the similarities between fault-prone components and vulnerability-prone components and between failure-prone components and attack-prone components, making borrowing from the latter field useful. The research objective of Gegick and Williams's work is to predict vulnerability and attack-prone components from static analyzer alerts.

This objective leads them to a general linear model with a Poisson distribution for the number of vulnerabilities per component based on the alert density for that component. Although Gegick and Williams scanned (with Fortify's SCA) ten releases of Sendmail totaling 1,000 files, they still had few data points when it came to vulnerabilities per se. With that caveat, they did show a relationship between SCA alert density and the number of vulnerabilities per file but found no relationship between SCA alert density and the number of exploits per file.

A questioner led Gegick to describe how the SAMATE project at NIST is a similar effort to this work and to note how version changes in Sendmail make double counting likely. Gegick is working on other targets besides Sendmail, as the main issue at this stage is getting more data. He hopes that in due course he will be able to publish correlations between vulnerability density and the alert density from Fortify.

PRACTITIONER PANEL—BECKY BACE, TRACK CHAIR AND MODERATOR

Brad Freeman of GE GIS Security Services, Shambla Naidoo of Wellpoint, and Ed Georgia of Booz Allen Hamilton's Information Security Practice described how they use metrics to make better decisions. The panelists opened with a few remarks.

Freeman began with the desirables for a metrics program within a firm the size of GE: simple, flexible, and hierarchical. Their program is roll-up oriented with a homegrown built around the products of ClearPoint Metrics. The basic issues in building any metrics program are:

- What are we measuring?
- Beware of poorly defined metrics and poor measurement systems
- Why are we measuring it?
- The "So what?" factor and tying metrics to business benefits
- How are we measuring?
- Manual vs. automated, actionable reports

In response to a question, Freeman said that comparison across departments is valuable and helps justify a metrics program.

Naidoo also began with a set of basic questions:

- With whom are we communicating?
- What is the message?
- Why is it important to hear?
- What do the numbers mean?

In so many words, she stressed that the top of an organization is populated by people who are overwhelmed by distractions, and thus brevity will be a key factor in getting through. She made several like points:

- Messages must be aligned with corporate priorities.
- Metrics will not get you an audience with the Board of Directors.
- Clarity for risk profiles is essential.
- You must show ROI and/or risk reduction if you are to be heard.

A questioner asked if, in so many words, this was selling, and Naidoo said that at the top everything is about selling. When asked whether that selling is just a matter of FUD, Naidoo reminded us that when fear declines so does funding. A third questioner asked whether the numbers are pushed on the management committee or pulled from Naidoo's team. She suggests that you ask top management, "What are your problems?" and speak only to their answer; that is all they will listen to anyway.

Giorgio stated that measurement perturbs a system and, as such, you must put metrics in the right hands. In government, the reason you measure is to subsequently acquire dollars. In business, the reason you measure is to drive di-

rection. Visualization matters because visualization, such as in dashboards, is what drives tactical decision-making. He challenged the audience to ask themselves, "Whom do we serve?" and, in that light, reminded the audience that metrics do not a compliance program make.

A questioner took this to heart and asked, "So what are the 'go-to' numbers?" Giorgio said that the Board of Directors wants to know, "Am I safe?" with a strong emphasis on the "I." With that in mind, Giorgio pointed out that if all you are doing is counting something, then that is not Board-worthy. He also pointed out that, in government, certification and accreditation only cost money—there is no positive return on the investment in them.

A questioner asked about the government point, whether there was a way to boil down the mix of program dollars, other resources, head count, and so forth. Giorgio said no, and that that is why he (we) are not welcome, and that metrics will only be useful as a backstop in an argument.

In Bace's view, it is time to rethink how we practice. As an industry, we are now into a period of specialization, and only in like specialization can our metrics be meaningful. An unanswered question was raised about how this guides the particularly vexing problem of counterparty risk, where the trading of data with counterparties endangers both sides of the transaction.

- ***Off-Program Comments***

  Adam Shostack

Shostack argued that breaches are great for metrics programs because they create sources of information with very low levels of bias. He referred all to two sites, http://attrition.org/dataloss and http://etiolated.org/.

- ***Debate: Stump the Chumps***

  Russell Thomas, Meritology; Mike Rothman, SecurityIncite; Pete Lindstrom, Spire Security; Andrew Jaquith, Yankee Group

Rather less organized than other interactions, the "chumps" took questions from the audience entirely. The present author regrets that he could not make enough sense of what followed to make a useful addition to this digest.

New Security Paradigms Workshop

White Mountain Hotel and Resort, NH, USA
September 18–21, 2007

Summarized by Matt Bishop (bishop@cs.ucdavis.edu)

The 2007 New Security Paradigms Workshop (see http://www.nspw.org) began with a reception and dinner on Sept. 18 and ended at noon on September 21. The workshop was highly interactive, with participation limited to about 30 people. It encourages authors "to present ideas that might be considered risky in some other forum," and all participants were charged with providing feedback in a constructive manner. The resulting intensive brainstorming proved to be an excellent medium for furthering the development of these ideas.

- ***Security and Usability: The Gap in Real-World Online Banking***

  Mohammad Mannan (presenter) and Paul van Oorschot

This paper examined what banks expected their online customers to do, and how that matched what customers knew they had to do and whether they could do it. The notice that banks give users (typically on the bank's Web site) is small, often overlooked, and contains fine print. As a result, many users are unaware of these expectations. For example, when the researchers asked a group of computer science students, researchers, and professionals how many of the requirements they met, most did not meet them all—and the researchers thought this group would be most likely to know, and meet, those expectations.

Banks expect online customers to have firewalls and antivirus software, and to keep up to date with security patches. But many users are not aware of security problems. The banks also gave misleading information. For example, one bank instructed users to ignore a message about an SSL certificate that failed to verify for its intended purpose. Banks often contracted with third-party firms for marketing purposes, and the resulting URLs looked suspiciously like phishing URLs. Finally, the banking Web sites failed to authenticate themselves to online customers, which contributed to the problem.

The researchers concluded that expecting users to follow the "shared responsibilities" or protecting their banking information was unreasonable given the lack of clarity and the nature of those expectations.

- ***A Privacy and Security Assurance Offer System***

  Jeffrey Hunker (presenter)

Currently, when a provider fails to protect a consumer's private information given to it for a limited purpose, the consumer has to take extensive action to protect him- or herself, while the provider usually faces only the consequences of reputation loss. To better link the responsibility and accountability for security of privacy-related information, this talk suggested an alternative approach, in which the consumer can opt in to one of several privacy guarantees (contracts) for a fee. The provider would have insurance policies supporting these guarantees. If the provider violates the guarantee, the consumer would have appropriate redress (e.g., be financially compensated or receive some other form of restitution). This scheme is a risk management scheme with insurance providing much of the incentive.

Pricing insurance premiums is not an exact science. Some markets do not support pricing risk (e.g., insurance for rock concerts), but insurance companies provide insurance for them. Two approaches enable violations to be detected.

The first is to write the privacy guarantees (contracts) in such a way that violations become clear. The trial bar also has an incentive to detect these problems, because its members can sue for them.

This in many ways resembles an architecture that provides software services rather than software. Finally, if the different privacy guarantees could be structured as a lattice, much of the work done on multilevel security policies may be applicable.

- ***Authenticated Names***

  Stanley Chow (presenter), Christophe Gustav, and Dimitri Vinokurov

This paper tackles the problem of authenticating identity to prevent phishing. For example, if you get a telephone call and have caller ID, the name of the caller is displayed. How can you be sure that it is accurate?

The authors propose a scheme based on the way trademarks are handled.

The RealName scheme defines "local jurisdictions" as geographical or professional groupings of brand names. Each jurisdiction runs a RealName registry that registers brand names. Each registry has its own name space and is authoritative over that name space. When a company registers, the registry gives it a certificate.

A user who wants to verify that a site's claim to belong to a particular brand is true requests the certificate from the site. The user then validates the certificate as coming from a trusted RealName registry. The user will normally trust a small set of registries.

Suppose a user is looking for a particularly unusual item, and a search engine says it is available at the XYZ company. The user finds a Web site for the XYZ company. If the company is in a trusted registry, the user can authenticate the identity to be sure it is the right XYZ company. If not, the user must establish trust in the company and then import its certificate, or establish trust in the company's registry and import its certificate, and then proceed as before.

Various extensions to handle delegation were presented.

- ***Security Automation Considered Harmful?***

  Keith Edwards (presenter), Erika Shehan, and Jennifer Stoll

Conventional wisdom holds that users comprise the weakest link in the security chain, so the system should do as much security management as possible to eliminate this link. The authors' thesis disputes this approach, holding that inappropriate automation is a direct cause of many of the problems associated with "usable security."

Misunderstood social and environmental contexts, mismatched values, and missteps in user experience all limit the effectiveness of automation. Automation implies "one size fits all," but differences in contexts mean different security needs. Because the end user usually uses a preconfigured policy, the user's need for anonymity, for example, can conflict with the preconfigured settings ensuring accountability. Finally, if automated security mechanisms are only "mostly right," the mechanism may call upon the user to disambiguate exceptions (which most home users are not knowledgeable enough to do) or may ignore errors.

The authors recommended exposing the security infrastructure, rather than hiding it for all but exceptional cases, tying security decisions to user actions, and using approaches drawn from social networking. They agreed with a questioner that making the workflow model of the system match the user's mental model of the system would improve the ability to automate appropriately.

They concluded that automation has inherent limitations even if the technology behind it is faultless.

- ***Self-Healing: Science, Engineering, and Fiction***

  Michael Locasto (presenter)

This position paper argued that a self-healing system is a pipe dream, because computers cannot anticipate failure conditions that the programmer does not know about. The standard code for binary search demonstrates this; despite having been proved correct, it had an integer overflow flaw overlooked for 30 years! Further, systems are products of inherently flawed human processes, with constantly shifting demands, and can be physically unreliable.

This thesis distinguishes between restorative healing, which responds to symptoms rather than causes (e.g., the skin healing in response to a cut), and improvement, which repairs the underlying cause of the problem. This is essentially the difference between detecting new instances of known classes of failures (responding to symptoms) and detecting new, previously unknown classes of failures (responding to underlying causes). Which do we expect from self-healing systems?

The discussion identified some limits to self-healing. First, how does a system with inherently incorrect code "self-heal"? On a deeper level, how does the system establish that there is a problem that needs to be healed? This is the same problem as anomaly-based intrusion detection faces: establishing what "normal" means.

Another important question is whether developing self-healing systems is appropriate, given the cost of development and the impact of self-healing mechanisms on efficiency.

PANEL ON THE FUTURE OF BIOLOGICALLY INSPIRED SECURITY: IS THERE ANYTHING LEFT TO LEARN?

Chair: Paul van Oorschot

Panelists: Michael Locasto, Jan Feyereisl, and Anil Somayaji

To foster debate, initially the three panelists took simplified positions. Michael started by saying that we learned much from biological systems; for example, strategies for anomaly-based intrusion detection and response have been

learned from the immune system, and artificial diversity has been inspired by nature. But creating workable computer systems that really are analogous to biological systems has been pretty unsuccessful, because of biology's complexity. We've already learned the big concepts from biology, and so it is time to move on from biologically inspired security.

Jan pointed out that we understand relatively little of how biology works, particularly as security researchers. Medical doctors spend many years studying biology and they still don't understand very many things. There are many biological systems, such as those involved in reproduction, that have not been adequately studied for their security properties. Thus, we have just scratched the surface of the possibilities for biologically inspired security.

Anil then argued that applying biological metaphors to computer security was a mistake for three reasons: It led to poor research because people familiar with biology but not computer security tend to produce poor-quality security research; biological systems address the wrong problems from a security standpoint because they focus on availability for survival rather than integrity or confidentiality; and biological systems are too complex—if we imitate them, we'll produce computer systems that are too hard to understand.

The subsequent discussion was lively. One attendee noted that rejecting analogies to biological systems because they don't solve the security problem ignores the fact that nothing in security works. A reply pointed out that maybe biology got it right to focus on availability over confidentiality or integrity, as availability is what is most important in practice. Others from the audience argued that we do not yet have the right model for translating biology, because we need to take the "ecological context" of living systems into account (the security equivalent being a threat model). Some attendees thought, though, there was some promise in designing fear into computer systems.

A key point raised was the difference between the evolved systems of nature and the designed systems that we have in computer science. Designed systems are always different from evolved systems because they are created by different processes (purposeful design vs. random search). One attendee argued that it is important to pay attention to epistemology here, as there is a big difference between copying a system and being inspired by one. Further, biological systems and computer systems are fundamentally different, and the key question is whether looking into the commonalities is apt to be more fruitful than doing other things. Anil disagreed with this statement, arguing that there is no fundamental difference between designed and evolved systems.

The panel concluded with Michael saying that given that we will soon be engineering biology as we now engineer computers, bioengineering raises critical security issues itself!

■ ***Robustly Secure Computer Systems: A New Security Paradigm of System Discontinuity***

Jon Solworth (presenter)

The theme of this talk was that we need to stop doing what does not work. The problem is that today's systems were designed before the lack of security became a major problem. Over time, new features were introduced and others removed, without thought to the security consequences. Then came the attackers.

The speaker mentioned the usual pitfalls of nonsecure programming, emphasizing that experience shows that only a few programmers have the right mindset to write secure code. The solution is to write new operating systems and programming languages in which these pitfalls are engineered out of the system. He then described the "application trap."

The application trap is a circular trap: No one will use new systems with no applications, but neither will anyone write applications for a new system that has no users. This led to the observation that one could introduce a new operating system that is incompatible with every existing application (the "system discontinuity") but uses virtual machines to support existing application-rich systems with poor security, and new operating systems with few applications but good security. In fact, such development is underway.

Considerable discussion ensued about the nature of flaws and vulnerabilities, and whether remediating them at the operating-system level would fix them at higher levels of abstraction, e.g., in browsers. The conclusion was that a new operating system could improve things, but we need to determine how to build easier-to-use operating systems and programming languages.

■ ***Information Protection via Environmental Data Tethers***

Matt Beaumont-Gay (presenter), Kevin Eustice, and Peter Reiher

A data tether is a mechanism that makes data accessible only when in a secure environment. When a mobile computer containing data is moved out of that environment, either the data is removed or the data is encrypted and the key stored on a secure server and deleted from memory. If the data was in memory, the process must be suspended and memory encrypted, or the process must be terminated. In this way, if the mobile device were stolen, the thief could not access the data. But when the mobile device could access the secure server, the data would then become available again.

Someone made the point that even if an attacker could introduce malware onto the mobile device, the malware could not access the data, as it is either encrypted or nonexistent. There was considerable discussion about the secure environment, which was defined as one in which the secure server could be contacted. Other assumptions were

that the user is nonmalicious but not reliable, the computer is connected to a network when in the secure environment, and it was undesirable or impractical to store the data on the secure server.

The system would have to track information flow, because the policy associated with the data would determine whether the data tether needed to protect the data. Thus, the policy is associated with the data and not with its container (file, etc.). How to do this in a commodity system is one of the research challenges, as is determining the contexts of a secure environment and of the disconnected operation of the laptop. The last point caused a brief discussion of what would happen if the laptop could not reconnect to the server because the latter was unavailable. If the data on the laptop were mission-critical, this could turn the security mechanism into an effective denial of service tool.

■ ***The User Is the Enemy***

Vidyaraman Sankaranarayanan (presenter), Madhusudhanan Chandresakaran, and Shambu Upadhyaya

This position paper argued that user actions should be treated as malicious because users do not follow security best practices, through ignorance or maliciousness or because they are oriented toward immediate performance gains. It then proposed an incentive/penalty system to encourage users to abide by the security policies. Specifically, users who followed the security rules would be rewarded by (for example) allowing them to use programs such as instant messaging services and providing them with more bandwidth; those who failed to do so would be penalized by reducing their quality of service, denying use of some programs, and so forth.

The advantage to this scheme is that it directly addresses what the user does to protect, or weaken, the system. The effect of what actions the user takes is proximate and concrete. It also eliminates the problem of nagging alert boxes that users tend to close without reading or understanding.

This proposal was controversial, producing a heated discussion. Three points emerged. The first concerns the user who wants to comply but cannot; the example cited was the inability to use EndNote without triggering security alerts from Vista. The response was that a better, more stable system and software need to be designed; failure to do so would make the user feel that security is a good idea, but "not in my backyard." The second point was that the language of calling the user an "enemy," "ignorant," and so forth was probably counterproductive, driving users and security personnel farther apart; this led to the opinion that the paper really argued that user actions, and not users, are the enemy. The third point repeated an objection to the data tethers: that the consequences of penalizing the user could cause a catastrophic failure if the user were denied necessary resources because of the penalties. This led to the question of when the computer knows best and agreement that this approach assumes no false positives in detecting the user violating the security policy.

■ ***Computing Under Occupation***

Klaus Kursawe (presenter) and Stefan Katzenbeisser

The computer security battle is going badly, said the authors, and providing large-scale protection against platform compromise is becoming less and less plausible. Using a service-oriented business model, an advanced infrastructure, and high-quality attack programs, and recruiting highly skilled personnel, organized crime is outpacing the defenders, who fight compatibility issues, lack of user awareness, and slow adoption of security mechanisms that are generally not effective enough.

Consequently, we may have to accept the fact that most platforms are under control of some "cybermob" and learn to work under that assumption.

Thus, as defenders we need to make the attackers (ab)use of our systems resource-intensive and uneconomical, while protecting critical assets from attackers who fully control the defenders' PCs. The assumptions making this possible are that users are honest although unwilling or unable to expend resources, attacks do not target a particular individual, and the "attacker" is actually an organization with limited human resources seeking financial gain.

The discussion focused on the new paradigm of computing on systems known to have been compromised. Someone pointed out that this was to a large degree a social problem. Other suggestions revolved around mitigation techniques that would limit the gains of the attackers, but many of these also functioned as denial of service attacks against the legitimate users. The talk concluded by suggesting that the security war may be lost already, and we need to find ways to continue to use our systems by better understanding the internal structure of the attackers, our own assets, and how to use both to make the attacker's life hell.

■ ***VideoTicket: Detecting Identity Fraud Attempts via Audiovisual Certificates and Signatures***

Deholo Nali (presenter), Paul van Oorschot, and Andy Adler

This paper presented a method that helps detect identity fraud attempts by embedding audiovisual information in certificates and using audiovisual recordings in lieu of conventional user digital signatures. An av-cert is a signed audiovisual recording in which a user identifies him- or herself. An av-signature is an audiovisual recording in which the user gives consent to a particular transaction. A bank issues the user an av-cert. To purchase something, the user gives the av-cert and av-signature to the retailer, who passes both to a verifier. The verifier validates both the signature and the certificate and sends the authorization and authentication status to the retailer, who (assuming both are good) provides the services or goods to the user.

This scheme verifies identity by using biometrics and verifies the consent for the transaction. It works with both onsite and remote transactions, and it uses widely deployed tools such as Web cameras. Drawbacks include issues of privacy and questions about whether biometric identification is accurate enough to make this technique cost-effective. The automated method is inexpensive (with amortized cost of 5 cents per transaction over a three-year period), but negatives require manual intervention to determine if the negative is false, and this drives the price of a transaction up considerably (to nearly $5 per negative transaction). People pointed out thay although facial recognition mechanisms were quite accurate under laboratory conditions, when deployed in the field their accuracy was considerably more problematic.

Assuming the videoticket approach proves feasible, it shifts the risk of the transaction from the bank and retailer to the user; to compromise the transaction an attacker must coerce the user into performing the transaction. Also, given the state of the art, it is possible that an attacker could generate a human image good enough to fool current automated audio and visual biometric tools within the next five years. To address this issue, audiovisual signatures could include transaction-specific information (unpredictable by attackers). Finally, one participant pointed out that his evil twin brother Skippy might be able to impersonate him and carry out the transaction.

Statement of Ownership, Management, and Circulation, 10/1/07

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. 2007) published nearest to filing date of 10/1/07*
A. Total number of copies	7083	6825
B. Paid circulation		
Outside-county mail subscriptions	3785	3709
In-county subscriptions	0	0
Other non-USPS parcel distribution	1747	1734
Other classes	0	0
C. Total paid distribution	5532	5443
D. Free distribution by mail		
Outside-county	0	0
In-county	0	0
Other classes mailed through the USPS	57	54
E. Free distribution outside the mail	1146	678
F. Total free distribution	1203	733
G. Total distribution	6735	6175
H. Copies not distributed	348	650
I. Total	7083	6825
Percent Paid and/or Requested Circulation	83%	89%

I certify that the statements made by me above are correct and complete.
Jane-Ellen Long, Managing Editor

Participate in Upcoming USENIX Conferences

2008 USENIX Annual Technical Conference

JUNE 22–27, 2008, BOSTON, MA, USA
http://www.usenix.org/usenix08
Paper submissions due: January 7, 2008

Usability, Psychology, and Security 2008

Co-located with NSDI '08
APRIL 14, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/upsec08
Submissions due: January 18, 2008

17th USENIX Security Symposium (USENIX Security '08)

JULY 28–AUGUST 1, 2008, SAN JOSE, CA, USA
http://www.usenix.org/sec08
Paper submissions due: January 30, 2008

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

Co-located with NSDI '08
APRIL 15, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/leet08
Paper submissions due: February 11, 2008

8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)

DECEMBER 8–10, 2008, SAN DIEGO, CA, USA
http://www.usenix.org/osdi08
Paper submissions due: May 8, 2008

USENIX

writing for ;login:

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of ;login:, is to submit a proposal first.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, mini-paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, bear in mind that a page in *;login:* is about 600 words. It is unusual for us to publish a one-page article or one over eight pages in length, but it can happen, and it will, if your article deserves it. We suggest, however, that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at http://www.usenix.org/publications/login/sched.html.

COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in ;login: and on the Web. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

In the past, there has been only one focus issue per year, the December Security edition. In the future, each issue may have one or more suggested focuses, tied either to events that will happen soon after *;login:* has been delivered or events that are summarized in that edition.

Brilliant & Appreciated?

You'll be both at NetApp.

netapp.com/careers

Number six on the 2007 *FORTUNE* "100 Best Companies to Work For" List

NetApp is seeking enthusiastic people to join our team. Our employees are passionate about what they do and the synergy they create in the workplace. We empower our employees to creatively identify solutions to overcome challenges. We have aggressive goals, but great rewards too: flexible schedules, generous compensation, and pride in our innovative products.

Come see why we were named a premier employer by *Fortune* magazine. And also by *ComputerWorld*, *Capital Magazine*, and the *Silicon Valley Business Journal*, among others.

We are hiring for all positions. Visit www.netapp.com/careers to submit your resume for the career invitational.

NetApp®

Simplifying Data Management

;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