

conference reports

THANKS TO OUR SUMMARIZERS

17th USENIX Security Symposium (Security '08) 75

Kevin Borders
Andrew Brown
Joseph A. Calandrino
William Enck
Steven Gianvecchio
Dave King
Bryan Parno
Ben Ransford
Sandra Rueda
Joshua Schiffman
Gaurav Shah
Micah Sherr
Zhenyu Wu

2nd USENIX Workshop on Offensive Technologies (WOOT '08)..... 99

Joshua Mason
Sam Small

2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08).....102

Eric Cronin
Rik Farrow
Eric W. Smith

3rd USENIX Workshop on Hot Topics in Security (HotSec '08) 109

Kevin Borders
Alexei Czeskis
Dan Ports

Metricon 3.0..... 114

Daniel Conway

17th USENIX Security Symposium

San Jose, CA
July 28–August 1, 2008

OPENING REMARKS, AWARDS, AND KEYNOTE ADDRESS

Summarized by Bryan Parno (parno@cmu.edu)

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

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

Debra Bowen, California Secretary of State

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

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

Digital voting machines, introduced in part to prevent voting errors, are subject to their own collection of flaws and vulnerabilities. For example, an early electronic election in Orange County presented voters with the wrong ballot, preventing them from voting in the correct elec-

tion. The touch screen may be misconfigured or confused by a voter with unwashed hands. Voter education is also a challenge, as is poll worker training. Secretary Bowen noted that California employs almost 100,000 poll workers, and, nationwide, the average poll worker age is 77.

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

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

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

recounts for close elections where the final tally differs by less than 0.5%.

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

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

WEB SECURITY

Summarized by Ben Ransford

■ *All Your iFRAMES Point to Us*

Niels Provos and Panayiotis Mavrommatis, Google Inc.; Moheeb Abu Rajab and Fabian Monrose, Johns Hopkins University

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

Provos described how clients are infected. Worms cannot easily traverse NATs or firewalls, but almost everyone uses a Web browser. Malefactors have therefore begun using vulnerabilities in Web browsers to install malware that

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

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

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

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

■ **Securing Frame Communication in Browsers**

Adam Barth, Collin Jackson, and John C. Mitchell, Stanford University

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

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

sites, the authors propose it as the policy browsers should follow. They wrote patches for Firefox and Safari, and they notified Microsoft; all major vendors (except Opera, with whom the authors are talking) now implement the Descendant policy.

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

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

- **Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking**

Michael Martin and Monica S. Lam, Stanford University

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

authentication or changing or deleting data. For this study, the authors considered scenarios in which Web application developers are honest but careless.

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

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

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

INVITED TALK

- **Political DDoS: Estonia and Beyond**

Jose Nazario, Senior Security Engineer, Arbor Networks

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

Nazario opened his talk by giving background on the history of DDoS attacks and describing recent trends. There are several types of DoS attacks, including bandwidth exhaustion (e.g., UDP and ICMP floods) and server resource exhaustion (e.g., HTTP GET request floods and SYN floods). There are also different ways of performing DDoS attacks, from simple human coordination (in which

everyone repeatedly refreshes the site simultaneously: “the F5 attack”) to more sophisticated software-based tools with a variety of features. The data collected by Arbor Networks shows several interesting trends. The peak bandwidth of DDoS attacks has increased from approximately 200 Mbps in late 1998 to as much as 25 Gbps in 2007. In addition, attack traffic makes up 2%–3% of all backbone traffic and TCP SYN attacks are still the most common form of DDoS attack. In terms of the global trend, the most attack command victims and the most C&C locations are in the United States and China.

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

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

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

A number of lessons were learned from the Estonian DDoS attacks. In particular, with help from various CERT teams throughout Europe, collaboration in filtering traffic and outreach for the purposes of research and investigation

were very successful. This leads to possible definitions of the roles of various organizations in cyber attacks: ISPs for defense, CERT teams for coordination, law enforcement for domestic issues, the State Department for international issues, and the military for offense.

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

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

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

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

CRYPTOGRAPHIC KEYS

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

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

Awarded Best Student Paper!

Protecting the contents of unattended or even lost laptops has become a serious concern as companies begin to roll out stronger mandates for information security. Since a locked computer screen can be thwarted by accessing the hard drive directly from another machine, people have

turned to full disk encryption, which is supported by most modern operating systems. Although the contents of the disk are indeed encrypted, accessing the data causes the OS to load the cryptographic key into memory. Normally this is not an issue, but, as William Clarkson demonstrated, all an attacker requires is recovery of the key.

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

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

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

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

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

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

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

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

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

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

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

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

INVITED TALK

■ *Building the Successful Security Software Company*

Ted Schlein, Kleiner Perkins Caufield & Byers

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

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

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

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

Schlein described his introduction to security in 1988, when he worked to create the first commercial-grade anti-virus software, Norton AntiVirus for the Macintosh. In deciding to aim their product at the Macintosh computer, he said, they took into account that Macintosh users liked their computers much more than PC users liked theirs,

meaning that it was more likely that Macintosh users would pay money to protect them. There were two big ideas that Norton AntiVirus used: first, when a disk was inserted, the virus scanner would scan the disk and check it against known virus signatures; second, the scanner would check resident memory to determine whether a virus was resident.

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

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

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

NETWORK DEFENSES

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

■ **CloudAV: N-Version Antivirus in the Network Cloud**

Jon Oberheide, Evan Cooke, and Farnam Jahanian, University of Michigan

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

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

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

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

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

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

■ **Highly Predictive Blacklisting**

Jian Zhang and Phillip Porras, SRI International; Johannes Ullrich, SANS Institute

Awarded Best Paper!

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

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

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

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

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

■ **Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks**

Jerry Chou and Bill Lin, University of California, San Diego; Subhabrata Sen and Oliver Spatscheck, AT&T Labs—Research

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

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

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

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

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

INVITED TALK

■ *From the Casebooks of . . .*

Mark Seiden, Senior Consultant

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

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

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

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

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

Seiden relayed a number of shorter anecdotes as well. For example, he described an individual who had been successfully targeted by phishing scammers multiple times. On one occasion, the individual received an email purportedly from the Nigerian police department. The email indicated that earlier scammers had been caught but \$100 was necessary to reclaim his original money. The individual ultimately sent not just \$100 but \$200. Seiden also encountered a system with a root password "r00t" that came with a vendor recommendation not to change it in case technical support was necessary. Seiden described how almost every contact in the digital world still leaves a trace or leaks data. Opening a safe can leave a heat signature that can be detected and used to reconstruct the combination minutes later. Acoustic cryptanalysis can reveal intricate details of machine operations such as cache misses. Finally, in a promi-

ment murder investigation, several news sources received images from the perpetrators via email addresses that could only have been found via searches on the news outlets' Web pages. Ultimately, correlating the IP addresses across sources would have potentially assisted in identifying those perpetrators, but the news sources rejected this idea based on privacy concerns.

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

POSTER SESSION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

BOTNET DETECTION

Summarized by Andrew Brown (ackbie@yahoo.com)

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

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

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

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

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

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

■ **Measurement and Classification of Humans and Bots in Internet Chat**

Steven Gianvecchio, Mengjun Xie, Zhengyu Wu, and Haining Wang, The College of William and Mary

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

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

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

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

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

■ **To Catch a Predator: A Natural Language Approach for Eliciting Malicious Payloads**

Sam Small, Joshua Mason, and Fabian Monrose, Johns Hopkins University; Niels Provos, Google Inc.; Adam Stubblefield, Johns Hopkins University

Sam Small started by outlining the paper's hypothesis that malware systematically uses search engines to find Web servers with vulnerable Web applications in order to infect

them. The aim of the project is to attract these bots to a Web site where they can be studied.

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

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

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

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

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

INVITED TALK

■ **Security Analysis of Network Protocols**

John Mitchell, Stanford University

Summarized by Bryan Parno (parno@cmu.edu)

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

In fact, the protocol analysis community has developed several approaches, including cryptographic reductions and symbolic methods. Cryptographic reductions attempt to relate the security of a protocol to the security of basic

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

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

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

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

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

In the subsequent question-and-answer session, an audience member asked whether PCL had been added to the Isabelle

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

HARDWARE AND SECURITY

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

■ Reverse-Engineering a Cryptographic RFID Tag

Karsten Nohl and David Evans, University of Virginia; Starbug and Henryk Plötz, Chaos Computer Club, Berlin

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

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

By using manual traces (later automated) of 1500 connections, the authors were able to successfully reverse-engineer the RFID tag. The presenter then discussed several available countermeasures that were shown to be ineffective against

automated reverse-engineering. These included reordering connections in nonintuitive ways and adding nonfunctioning dummy cells and super-dense chips. They even were able to identify several severe flaws in the Mifare Classic's encryption algorithm which lets anyone recover the key quickly with an offline attack, using about 500 GB of possible keys, or even with a SAT solver.

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

- **Practical Symmetric Key Cryptography on Modern Graphics Hardware**

Owen Harrison and John Waldron, Trinity College Dublin

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

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

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

A member of the audience wondered what the future of GPUs would be if people began regularly using them for cryptography. In response, the speaker indicated that the

graphics cards would contain multiple GPUs that could be used in a more general-purpose fashion.

- **An Improved Clock-skew Measurement Technique for Revealing Hidden Services**

Sebastian Zander, Swinburne University of Technology, Australia; Steven J. Murdoch, Computer Laboratory, University of Cambridge

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

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

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

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

INVITED TALK

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

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

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

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

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

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

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

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

SYSTEMS SECURITY

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

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

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

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

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

- **Hypervisor Support for Identifying Covertly Executing Binaries**

Lionel Litty, H. Andrés Lagar-Cavilla, and David Lie, University of Toronto

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

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

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

only identifies code within it. Patagonix and Segvisor are similar works but differ in their scope.

- **Selective Versioning in a Secure Disk System**

Swaminathan Sundararaman, Gopalan Sivathanu, and Erez Zadok, Stony Brook University

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

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

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

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

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

INVITED TALK

■ Hackernomics

Hugh Thompson, Chief Security Strategist, People Security

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

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

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

Thompson provided a definition of "Hackernomics" and its five laws and six corollaries. The first law is "Most attackers aren't evil or insane; they just want something." The

two corollaries of this law are: (1) it is very hard to protect against evil attackers, but we can provide protections that makes most attackers look for weaker targets; (2) the appearance of security is effective, as long as it is not easy to test the security level.

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

The third law is "In the absence of metrics, we tend to over-focus on risks that are either familiar or recent." For example, because it was recently reported that each year many laptops with important business data are lost in airports, many companies have started to spend a lot of money to implement company-wide laptop hard-drive encryption, while ignoring other, more likely security risks. Moreover, with increased frequency and damage of attacks, businesses put more focus and budgets on IT security; however, because the firewall is the most familiar security product, many companies ended up buying more and more firewalls and even daisy-chaining them.

The fourth law is "In the absence of security education or experience, people naturally make poor security decisions with technology." Thompson told a funny and sad story that illustrates this law. Back in the days of 5.25-inch floppy disks, one of his friends made backup disks of a mission-critical system and gave them to the secretary to label and keep safe. However, when the system broke down and needed restoration, not one of the backup disks was readable. Only later did his friend learn that the secretary cranked the floppy disks through a typewriter to label them. The corollaries of this law are that software should be made easy to use securely and difficult to use otherwise and that with proper education, specifications, and good metrics, developers are smart enough to do things right.

The fifth law is "Most costly breaches come from simple failures, not from attacker ingenuity." Examples include lost laptops with unencrypted sensitive information and badly chosen passwords. The corollary of this law makes an exception that, with sufficient incentive, bad guys can be very creative. A good example is the use of pornography to entice human users to solve CAPTCHA problems.

During the discussion that followed the presentation, one audience member questioned whether the use of pornography for CAPTCHA solving is really the result of attacker

ingenuity, because the discussion of its possibility was “floating around” well before its existence was confirmed. Thompson agreed in this case. Another audience member contributed a “co-law” for the third law: “In the absence of metrics, security practitioners tend to make decisions based on what’s possible, not what’s probable.”

PRIVACY

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

■ *Privacy-Preserving Location Tracking of Lost or Stolen Devices: Cryptographic Techniques and Replacing Trusted Third Parties with DHTs*

Thomas Ristenpart, University of California, San Diego; Gabriel Maganis, Arvind Krishnamurthy, and Tadayoshi Kohno, University of Washington

Thomas Ristenpart began by informing the audience of recent laptop theft statistics, including an FBI report indicating that 97% of stolen computers are never recovered. In response, users and enterprises are increasingly considering Internet tracking systems to aid in the recovery of stolen devices. However, such systems, if implemented naively, pose significant privacy concerns for end users. For example, a system that periodically emails or otherwise connects to a remote server runs the risk of exposing user habits. Consider the following threats: Unencrypted transmissions can be eavesdropped, recent locations can be retrieved from a local cache (e.g., “Sent Mail”), and the remote server unintentionally contains an enormous record of user movements that is subject to subpoena. A location-tracking system need not have these drawbacks.

Thomas presented three design goals for a privacy-preserving device tracking system: (1) prevent outsiders from learning private data (“piggybacking”); (2) ensure forward privacy; (3) prevent the storage provider from tracking a user. These goals are achieved in the Adeona system (named for the Roman goddess of safe returns). Adeona provides anonymous, unlinkable, and forward-private updates that are efficiently retrievable from a data store. The scheme begins with a Forward-Secure Pseudo-Random Number Generator (FSPRNG) to create an initial seed. The seed is used to create a unique index and encrypt location data, both of which are sent to remote storage. The next update, which occurs at pseudo-random intervals decided by a Poisson variable, derives a new seed from the previous one and repeats the process, after which the previous seed is destroyed. Hence the user need only remember the initial seed to derive all future seeds and indexes. At a later time, the user can then retrieve encrypted location updates by index from the server.

The Adeona system is available as open source from <http://adeona.cs.washington.edu>, in versions for Linux, Mac OS X, and Windows. The location updates include internal and external IP addresses, nearby routers, access points, and

photos (Mac only). The data itself is stored in the OpenDHT distributed hash table, which is freely available to all clients.

Steven Bellovin inquired about the open source license and the additional “I Agree” button on the tool’s download Web page. Thomas responded that Adeona is licensed as GPL, with additional indemnification for the researchers. They urge that the software should be used to aid police in laptop recovery as opposed to endangering one’s self by confronting the thief directly. Another audience member inquired whether there are mechanisms to prevent abuse of the storage mechanism for arbitrary data. Thomas noted that the paper discusses techniques such as ring or group signatures to verify memberships without compromising a user’s anonymity. A third audience member asked how network availability impacts storage updates. Thomas replied that there is a privacy trade-off wherein the tool caches location updates and sends a bulk update when connectivity is restored; specific details can be found in the paper.

■ *Panalyst: Privacy-Aware Remote Error Analysis on Commodity Software*

Rui Wang and XiaoFeng Wang, Indiana University at Bloomington; Zhuowei Li, Center for Software Excellence, Microsoft

Rui Wang began the presentation by discussing privacy concerns when submitting bug traces to application developers. From the end user’s point of view, the memory dump may include private information such as passwords and credit card numbers; however, bug traces are immensely valuable for developers when fixing problems. Therefore Rui and his co-authors aimed to create a method of providing bug reports that minimizes private information while remaining accurate and efficient enough to help the developer. They achieve these goals with the Panalyst bug reporting system.

Panalyst works by iteratively including necessary portions of the memory dump produced on program crash. The bug-reporting framework initially identifies memory locations that potentially contain private information (e.g., inputs from forms in a Web browser or POST queries sent to a Web server). These areas of memory are initially blanked and sent to the Panalyst server, which uses taint analysis and symbolic execution to determine the cause of the crash. When more information is required, the server software generates a list of questions representing desired portions of the memory dump. These questions are sent to the Panalyst client software, where privacy policies (including thresholds of content entropy) are consulted to determine whether answering the questions would compromise the user’s privacy. This process of symbolic execution followed by questions and answers continues until the server can determine the cause of the program failure.

Panalyst was implemented and evaluated on a number of different applications, including Newspost, OpenVMPS, Null-HTTPd, Sumus, Light HTTPd, and ATP-HTTPd, versions of which contained bugs and were subject to stack-based overflow, format string errors, and heap-based over-

flow. All of the evaluated applications except OpenVMPS resulted in less than 10% information leakage; OpenVMPS experiments observed a 28.8% rate of information leakage owing to the type of bug (format string vulnerability).

A member of the audience pointed out that many portions of memory will not contain private information and asked how private portions are distinguished. Wang responded that input fields are partitioned to allow easier discovery and that they are working on better algorithms. Another audience member inquired how Panalyst handles data including embedded integrity codes or encrypted values. Wang replied that the client portion of Panalyst could decrypt the values before performing the analysis. A third audience member asked how Panalyst would respond to a software bug that writes private information to portions of memory not designated as private. Wang said that such a situation would be difficult to handle in Panalyst and that information accidentally written to a public field will be leaked.

- **Multi-flow Attacks Against Network Flow Watermarking Schemes**

Negar Kiyavash, Amir Houmansadr, and Nikita Borisov, University of Illinois at Urbana-Champaign

Negar Kiyavash began by describing different uses for watermarking technology, including detecting stepping stones and flows within anonymous networks. She forewarned that watermarking is used by both the “good guys” and the “bad guys,” and by “attack” she means defeating the watermarking mechanism itself (e.g., removing it) or simply detecting its existence (depending on the scenario). When applying watermarking techniques to network flows, traffic first passes through some fixed point, called the watermarker (e.g., a router), which encodes a signal; common watermarking techniques modify the spacing between packets within a fixed time interval. The traffic then passes through some sort of distortion (e.g., an anonymization network), before reaching the watermarking detector, which subsequently extracts the signal.

Negar and her coauthors focused on detecting and removing watermarks from interval-based watermarking schemes encoding messages across multiple flows. Specifically, they considered Interval Centroid-Based Watermarking (ICBW), Interval-Based Watermarking (IBW), and a spread-spectrum watermarking scheme (DSSS). Their attack observes packet arrival times, modeling the interarrival times as a Poisson process. This information drives a two-state Markov chain, which indicates the existence of packets in a flow. By properly tuning the model parameters, they were able to detect watermarking “templates” (i.e., gaps within the flow) by aggregating as little as 10 flows. An attacker could then use this information to detect and remove the watermark, defeating the scheme.

Negar concluded the talk by discussing possible countermeasures that strengthen watermarking. Their investiga-

tions indicated that changing the position of the intervals produces the best results. There were no questions from the audience.

INVITED TALK

- **A Couple Billion Lines of Code Later: Static Checking in the Real World**

Dawson Engler, Stanford University; Ben Chelf, Andy Chou, and Seth Hallem, Coverity

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

Dawson Engler is an associate professor at Stanford University. He, along with some of his former students, founded Coverity, a software company that sells static analysis tools to the industry. Coverity has over 400 customers and over 100 employees. The static analysis tool that Coverity uses is a commercialized version of a bug-finding tool that Engler and his lab had developed in academia. Before commercialization, their static analysis tool had been successfully executed on the BSD and Linux kernels; this led to them thinking there was little work to be done beyond boxing and selling the product. His talk focused on the large number of unexpected issues Coverity ran into while dealing with customers and selling a static analysis tool in the computer science industry.

Coverity checks code to make sure that certain ad hoc correctness rules are satisfied; for example, every time a lock is acquired, it is always released. Because systems have many constraints of this kind and many lines of code, they contain numerous bugs that are easy for a static analysis tool to find. If, when run, an analysis tool does not find thousands of errors in a typical codebase, then, said Engler, there is something wrong with the analysis.

For every prospective customer, Coverity does an on-site visit for a day. Its analysis tool is set up to work on the company’s source code in the morning; in the afternoon, it holds a meeting with the group to review some of the bugs the analysis has found. This on-site visit is meant to impress the client with how easily Coverity can be inserted into the development process (the first half of the day) and how it finds valuable bugs (the second half of the day). This requires an analysis that easily adapts to different codebases, because if something is wrong, there is no time to fix it. Most of the difficulties that the Coverity team has had with adapting to codebases are in compiling code that has been compiled with nonstandard compilers, uses domain-specific syntax, or uses nonstandard build hacks in order to compile. If any compiler allows it, then a static analysis checking tool must also allow it. However, the lack of uniformity among C and C++ compilers introduces a large number of syntax variants that any C/C++ static analysis tool must also handle. Engler stressed that these problems are not interesting from a research perspective, but without addressing

them, no sales will be made, because the customers will not be able to evaluate the quality of your analysis.

Social factors also proved surprising. Customers may dismiss legitimate bug reports as false positives if they are unable to understand the error. Other customers may not view bugs as worth eliminating, believing that if a bug occurs at runtime, the worst that will happen to them will be a call from a customer. Other bugs could be recovered from simply by rebooting the system, and if the QA department is unable to reproduce a bug, nobody can be blamed for it. Rather than arguing with the customer about bug reports, Engler suggested bringing large groups of people to the bug presentation meeting; the more people in the room, the greater the likelihood that someone will understand what you are talking about.

Another surprise was that customers have viewed improvement in Coverity's analysis as bad. Upgrades might increase the number of total errors, interfering with the customer's attempt to use Coverity's warnings as a metric of code quality. If a customer has fixed a thousand bugs and the new release, through better analysis, reveals a thousand more bugs, the customer is likely to feel that the upgrade has undone all of their work. False positives are also important: The first few error reports presented to the customer as a demonstration of the tool must not be false positives, and anything over a 20%–30% false-positive rate leads to the tool being untrusted, meaning that complicated but real error reports may be dismissed as false positives as well.

Engler concluded by reflecting that over the past four years customers have become far more receptive to the idea of static analysis, being aware of the positives it brings to a code project. As a result, it is much easier to sell static analysis tools to companies. As long as your analysis tool is able to find the source code and parse and compile it, and is set up correctly, it will find serious bugs.

One of the questions involved fixing: Although Coverity's static analysis can identify bugs, would there be a place in the market for an automatic resolver? Engler said that this would be possible, but care would have to be taken in how resolutions were inserted. Other questions focused on the customers themselves. Engler indicated that developers with a CMMI rating were likely to have a better overall quality of code, since developers following a coding standard are going to produce better code. Engler also said that the difficulties associated with parsing domain-specific extensions to C and other issues related to other exotic tools were unlikely to go away.

VOTING AND TRUSTED SYSTEMS

Summarized by Micah Sherr (msherr@cis.upenn.edu)

■ **Verifying Compliance of Trusted Programs**

Sandra Rueda, Dave King, and Trent Jaeger, The Pennsylvania State University

Sandra Rueda began her presentation by noting that trusted programs are expected to perform safe operations even though they have sufficient rights to exhibit unsafe behavior. The introduction of security-typed languages and reference monitors alleviates the need to blindly trust trusted applications. However, both techniques are only useful for verifying that trusted programs adhere to program policies. Sandra Rueda and her colleagues propose a mechanism for automating the composition of program and system security policies as well as the verification that a trusted program is compliant with the produced composite policy.

To automate the composition of program and system policies, Sandra Rueda and her co-authors envision augmenting Linux installation package programs with policy specifications and policy modules, the latter of which describe the rights that the system must grant for the application to operate correctly. Using their insight that “program integrity dominates system integrity (PIDS)” to simplify relating program and system policies, information flow techniques can be utilized to compose policies. Compliance testing can similarly be achieved by rephrasing the problem of verification in terms of reachability in the information flow graphs.

Peter Neumann pointed out that there may be some confusion between the terms “information flow” and “control flow.” He explained that information flow relates to multi-level security properties, whereas control flow describes the technique of never depending on any component that has a lower level of trust.

■ **Helios: Web-based Open-Audit Voting**

Ben Adida, Harvard University

Ben Adida began his talk by describing the promise of open-audit voting, in which any voter can both validate that his or her ballot has been cast by finding the encryption of his or her vote on a publicly accessible Web site and also provably confirm the winner of the election by following a “fantastic” cryptographic proof. Adida's Web-based open audit tool, Helios, brings us a step closer to this open-audit ideal.

Unlike voting systems used in public elections in the United States, Helios is designed for low-coercion elections and uses the Web browser as the voter interface. Although Helios does not introduce any novel cryptographic voting protocols, the contributions of the system are its ease of use and its ability to perform all cryptographic operations within the voter's Web browser. Adida described his “bag of tricks” for achieving the latter feature.

To illustrate the usability of his system, Adida performed a live demonstration by voting in an election (along with a handful of other conference attendees), obtaining a cryptographic receipt of his vote, confirming that his vote had been cast using his receipt, and tallying the results and outputting a cryptographic proof of the winner. Helios is currently available as a Web service at <http://www.heliosvoting.org/>.

Steve Bellovin wondered why voters who are not experts in cryptography should believe Helios's mathematical guarantees. Adida responded that it is not necessary for all voters to understand the security properties of an election. As with most complex systems, the public relies on experts for informed assessments. Adida posited that voters would be satisfied with a system that has been described as secure by experts in the field.

- ***VoteBox: A Tamper-evident, Verifiable Electronic Voting System***

Daniel Sandler, Kyle Derr, and Dan S. Wallach, Rice University

Despite the heavy criticism of deployed DRE (touchscreen) voting systems by the academic security community, Daniel Sandler noted that DRE systems have potential benefits (e.g., accessibility, instant feedback, flexibility, and a high level of user satisfaction). He and his colleagues propose a DRE system, VoteBox, that attempts to transition toward “software independence,” a property that prevents undetected system problems from creating undetectable changes in election results. To move toward this goal, VoteBox relies on a small trusted computing base, maintains “believable” audit logs backed by cryptographic guarantees, and offers both cast-as-intended and counted-as-cast voter verifiability.

Sandler described VoteBox as a composite of existing secure voting system techniques. VoteBox utilizes prerendered user interfaces to minimize the code running on the voting machine. Auditorium, a failure-resistant and tamper-evident network layer, is used to broadcast election events to all VoteBox terminals in the polling place. Using hash chaining and signed broadcast messages, Auditorium provides provable ordering of events and completeness of audit logs. Additionally, VoteBox uses a variation of Josh Benaloh's ballot challenge mechanism to provide evidence that the system is correctly recording ballots. After a voter has marked his or her selections, VoteBox publicly commits to the voter's choices by sending an encrypted commitment message to a remote challenge center. When a voter issues a challenge, VoteBox reveals the key used to encrypt the public commitment, proving that VoteBox had previously committed to the challenged (and consequently spoiled) ballot.

Daniel Sandler concluded by noting that the source code to VoteBox will be available soon at <http://votebox.cs.rice.edu/>.

Ari Feldman inquired about the potential ballot secrecy risk of permitting voting machines to broadcast messages outside the polling location. Daniel Sandler noted that Vote-

Box relays such messages through a separate listener device located in the polling place, and such a device could be configured to detect information leaks (e.g., hidden timing channels). Ka-Ping Yee pointed out that it would be useful to determine the minimal set of Java components necessary to build VoteBox in order to better enumerate the trusted components of the system.

- ***The Ghost in the Browser and Other Frightening Stories About Web Malware***

Niels Provos, Google, Inc.

Summarized by Ben Ransford (ransford@cs.umass.edu)

For his invited talk, Niels Provos promised a less academic—but more detailed, and therefore more frightening—version of the talk he gave during the Web Security technical session. The basic motivating point was the same, namely, that an underground economy thrives by exploiting PCs via Web-based malware and that the security community needs to understand the problem and develop solutions.

A fundamental problem that lacks a solution, according to Provos, is that the underground economy is not well understood, although studies such as Stefan Savage's (ACM CCS '07) are beginning to illuminate these dark corners. People meet in IRC channels to swap lists of credit card numbers, coordinate labor, and buy and sell harvested personal details. These resources are cheap and plentiful: A content provider might receive a few dollars for every 10,000 unique visitors it exposes to Web-based malware. Provos emphasized that the security community needs to find a way to make the business of botnets more expensive to conduct.

Google's unique position as a repository and conduit for much of the Web's information gives Provos and his collaborators unique opportunities for analysis. Provos described Google's method of discovering drive-by downloads, which exploit browser vulnerabilities to execute code on unsuspecting users' PCs and are often planted in legitimate Web sites by miscreants exploiting vulnerabilities in Web applications. (This part of the talk reiterated details from Provos's paper presentation.) Provos claimed that Google has been monitoring the use of JavaScript in exploits for about two years. Techniques they have seen include all manner of homebrew encryption functions, the most pernicious of which use their own decryption functions as the decryption keys; not-so-clever obfuscation; horribly nasty obfuscation; and combinations of these. Compounding the problem, exploit writers adapt very quickly to newly discovered vulnerabilities. For example, within a week of the disclosure of an animated cursor bug in Windows last year, the majority of the exploit code Google observed targeted that vulnerability.

Provos emphasized that, because exploits appear in Web sites of every type, no part of the Web can be considered safe. Attackers target general-purpose Web applications, meant for purposes such as forum hosting and database

administration, that are used across the Web. Furthermore, the modern Web bristles with third-party widgets that expose users to content from many different providers at once, and attackers are constantly gaining sophistication. The talk was rich with examples of ways in which clever malware authors use infected machines. Provos described an HTTP-based spamming botnet in which clients fetched batches of thousands of email addresses from a central Web server, then reported back after attempting to spam those addresses; this botnet appeared to know twenty-five million valid email addresses. Furthermore, Google observed that this botnet's kernel driver failed to install on some of its Windows test machines, so it sent a diagnostic memory dump to the central server. Other malware exfiltrates users' address books, which are almost certainly full of valid email addresses. Many malware programs record and upload all of the information users enter into Web forms. Provos told of an Apache module, inserted via an Apache vulnerability, that randomly injects polymorphic JavaScript code into outgoing HTTP responses. Other exploits behave selectively—for example, becoming malicious only when served to English-speaking users.

Users and server administrators can minimize their susceptibility by staying abreast of software updates and by using antivirus software. This is well known, but Provos pointed out nuances throughout his talk. First, it is easy to set up a Web site, but many Webmasters have no idea how to install security patches. Google tries to contact Webmasters when it finds exploit code on their sites, and it is working with volunteers from other organizations to help Webmasters, but it cannot provide one-on-one help to everyone. Second, running a fully patched database server does not protect against the kind of sloppy programming that invites exploits such as SQL injection. Third, people who use pirated software lag far behind patch cycles because doing otherwise might expose them to vendors' countermeasures; this problem is especially bad in countries where software piracy is common. Finally, the quality of antivirus engines varies widely; in Google's malware-hunting experience, detection rates with different engines have fluctuated from around 30% to around 80%. Provos advocated education of both users and Webmasters, but he acknowledged that this is an area that needs a great deal of further work.

An audience member asked about liability: Are the Webmasters of compromised sites ever held liable? Provos pointed out that many large sites use software—much of it free—that they did not write. He suggested that in the future Webmasters might use more aggressive automatic updating, intrusion detection systems, and periodic checking for their own sites in lists of compromised sites. He mentioned the “stopbadware” forum on Google Groups, where Webmasters and volunteers discuss specific problems and solutions. Rik Farrow pointed out Provos's focus on Internet Explorer, then mentioned alternative browsing strategies such as using separate virtual machines for

separate purposes or booting from a live CD to do sensitive browsing; both of these approaches suffer from usability problems. Provos agreed. Another audience member mentioned Greenborder, a company Google has acquired, but a topic on which Provos could not comment. The audience member then asked whether the security community might sometime endorse a single safe browser; Provos acknowledged the possibility and mentioned that some people were working on safe browsers; he mentioned the OP browser presented at this year's Oakland conference as an example. [Editor's Note: You can read about this browser in the August 2008 issue of *login*.]

SOFTWARE SECURITY

Summarized by Andrew Brown (ackbie@yahoo.com)

■ *An Empirical Security Study of the Native Code in the JDK* *Gang Tan and Jason Croft, Boston College*

Gang Tan spoke for his team about their investigation into the security of the native code found in the standard JDK. Tan notes that there has been a significant amount of work in verifying the Java (non-native) model in the JDK, but little on the native sections, of which there are approximately 800,000 lines of code in JDK 1.6.

The group used Splint, ITS4, and Flawfinder static analysis tools as well as custom tools that looked for common C errors such as buffer overflows and specific errors spelled out in the JNI manual. The advantage to using several tools is that the researchers were able to get very good coverage. The downside is that there were many false positives, which required manual verification. Because of the size of the code and because verifying the bugs was so slow, the group decided to focus on the code under the `java.*` package hierarchy.

The researchers were able to find many bugs in several categories. The first category was mishandled JNI exceptions. Unlike throwing an exception in normal Java code, doing a `Throw()` does not interrupt the flow of JNI code as would normally be expected. As a result, there were 11 instances where the developer failed to manually stop the flow of execution after throwing an exception. The second category of bugs occurred where JNI code stored pointers back into Java, where they were kept as Java ints. In some cases, it was possible for the Java code to change this value arbitrarily, which would certainly cause a crash when it was later dereferenced as a pointer in C.

Tan concluded by calling for tools with more sophisticated inter-languages analysis support and for sections of the JDK that are currently written in unmanaged code to be written in managed code if possible. An audience member asked whether the group had reported these bugs. Tan responded that they had. Somebody else asked whether they had considered using commercial tools that did not have as many false positives. Tan said they would look into it. Finally, Tan

clarified that they had written exploit code to prove that some of the bugs are exploitable.

- **AutoISES: Automatically Inferring Security Specifications and Detecting Violations**

Lin Tan, University of Illinois, Urbana-Champaign; Xiaolan Zhang, IBM T.J. Watson Research Center; Xiao Ma, University of Illinois, Urbana-Champaign, and Pattern Insight Inc.; Weiwei Xiong, University of Illinois, Urbana-Champaign; Yuanyuan Zhou, University of Illinois, Urbana-Champaign, and Pattern Insight Inc.

Lin Tan spoke for her team on the use of check functions to protect security sensitive operations (SSOs). The central premise is that, before these critical sections of code can run, special security checks must be performed to ensure the safety of the operation. Typically this check is code, such as Linux's `security_file_permission()` function, which should be called before file read/write operations to make sure that the user has the correct permissions. Tan stated that, although this type of function should always be called, inevitably there will be times when it is not. The authors created a tool that tries to check whether every instance where a security check is required calls the security check function.

One difficulty is converting high-level conditions such as “protect all file operations” into low-level rules that a source code checker can find. The group used the assumption that if an SSO needed to be protected by a check function, then most of the time it would be. In other words, forgetting the check is relatively rare. Another problem is that there are many ways to perform the same kind of operation, which leads to the challenge of locating which sections of code should be counted as SSOs. The group analyzed the SSOs where the check function was used properly and made them into a template for creating the general rule. The group found that an SSO represented as a “group of data structure accesses” allowed the checker to locate sections of code that should be protected by a check function but was not currently protected. The results of running the checker on the Linux and Xen codebases found 84 rules with 8 violations, with only 2 false positives.

Tan concluded that it was feasible to extract rules from existing code and to use those rules to find violations. One audience member suggested that the function might do different operations depending on the arguments. Another asked how the false positives appeared. Tan replied that the violation the checker found technically was a violation, but it was not of a variety that code authors meant to protect against.

- **Real-World Buffer Overflow Protection for Userspace & Kernel-space**

Michael Dalton, Hari Kannan, and Christos Kozyrakis, Stanford University

Michael Dalton presented a hardware-based method for providing runtime buffer overflow protection. Dalton started

the talk by reviewing existing buffer overflow protections implemented in hardware (e.g., nonexecutable pages), compilers (e.g., StackGuard), and kernels (e.g., W^X). He noted that these protections made buffer overflows more difficult but that they were insufficient.

The challenge is creating a system that works with unmodified binaries, catches most types of overflows, works in both user and kernel space, and does not slow down the execution of the binary. The group's answer is a dynamic information flow tracking (DIFT) architecture that adds extra taint bits to hardware registers. The system marks any memory that comes from input as tainted. Tainted memory is propagated from source operands to destination operands throughout the execution of the binary.

A key question is how memory can become untainted. The group considered forms of bounds checking but found that it broke legitimate code. Instead, the researchers opted for a method that recognized that buffer overflowing code relies on injecting new pointer values into memory. Their approach dictates that the hardware track user data and legitimate pointers. The enforcement rules are that all pointer values must be derived from known-good pointers and that tainted code should not be executed. Tainted data can, however, be used as an offset from a real pointer. Dalton detailed the procedures they used for finding pointers in the binaries and the libraries they used. They successfully booted Linux on the system with only a small amount of modification and found buffer overflows in user-space programs.

An audience member asked whether there was a way to prevent user data from being corrupted. Dalton answered that there is not enough information at the hardware level to decide what constitutes corruption. Another audience member asked whether pointers were sent over the network. The answer was that OpenSSH does actually send pointers over a local socket and so an exception had to be made for that.

INVITED TALK

- **Managing Insecurity: Practitioner Reflections on Social Costs of Security**

Darren Lacey, Chief Information Security Officer, Johns Hopkins University/Johns Hopkins Medicine

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

Darren Lacey began his talk by explaining the title, “Managing Insecurity.” With new security concerns constantly arising, and no clear solutions, IT departments frequently must do their best to manage their systems in the face of insecurity. Darren explained that he was giving the talk from the perspective of nonprofit organizations, which exhibit characteristics such as diverse management structures, a wide range of information collection, limited resources, and interesting and important missions. His goal was to

explain to security researchers how bosses see risk and to incite interest among researchers to develop novel solutions for the problems encountered by organizations like his. The talk was divided into three sections: a discussion of risk in hospitals, information security challenges at the ground level, and areas in need of academic pursuit.

As Chief Information Security Officer at Johns Hopkins Medicine, Darren sees risk differently from the typical IT department. A hospital's greatest risk involves the lives of its patients. Seven percent of patients in academic medicine fall victim to mistakes during medical procedures. A study in the Johns Hopkins ICU showed that each patient requires 178 discrete actions per day. Given a 1% mistake rate, there are an expected two mistakes per patient per day. Technology can be applied to reduce mistakes, but many complications and new risks result. For example, "wiring" hospitals and using Electronic Medical Records (EMRs) can reduce operational costs and eliminate illegible instructions and prescriptions, but it creates privacy concerns and legislature-mandating compliance. It also eliminates many redundant checks between the doctor and the pharmacist that frequently correct errors. Furthermore, it can result in additional risk for the hospital when computers contradict doctors who prescribe safe levels of medicine in combinations that otherwise conflict. The latter is resolved by creating an elaborate list of "exceptions." All of these new risks increase the demand for IT infrastructure and personnel.

Managing the IT infrastructure of a hospital is an unforgiving and never-ending task. Significant resources are devoted to legal and regulatory compliance, with e-discovery requiring nontrivial effort in an organization with dozens of different email systems. Darren stated, "Nearly every security decision undermines security somewhere else." Previously, the first person to arrive would stay logged in all day so that equipment could be quickly used. To make logging-in easier, a single sign-on system was implemented. Now, the person with the most access stays logged in. In many cases, efforts toward compliance are realized as simply slowing down noncompliance. For example, Darren is frequently asked why he sets password change policies when stronger, longer-lasting passwords are more secure. Unfortunately, in medical environments people continually hand out their passwords, and forcing change limits the risk of each disclosed password.

The core security trade-off that must be asked is, "How much are you willing to screw things up to improve security?" For example, whitelist firewall policies are a recommended best practice. Unfortunately, even the vendors do not always know everything that is supposed to run; this ends up killing people. In one case, a vendor responded, "Don't shut that down; that might be the emergency port." Security tools can never be initially deployed in "enforcing mode," and they must be watched carefully to ensure that the system will not break when the switch is flipped. Security has a cost. It adds complexity, slows things down, and

reduces creativity. It takes resources from other worthwhile IT projects. If you are very effective, the users hate you, which means you get less funding. If you are ineffective, the users hate you, which again means you get less funding. In summary, "security is a virtue, but it isn't the only virtue."

Darren finished the talk by discussing areas for academic research endeavors. On the forefront is usability. Most security tools are used and maintained by nonsecurity people. Security researchers also need to work with economists. Darren recently attended the Workshop on Economics in Information Security (WEIS), and he noted a lack of focus on a number of important security problems. Finally, he noted hopes of further research in measurements. He believes that useful measurements will come from sampling, using technologies such as honeynets, but the goals of such work are still unclear.

Audience questions, significantly, focused on the security implications of outsourcing, especially overseas outsourcing. Overall, Darren responded that the best method is to work with one vendor for all outsourcing needs. However, the amount of outsourcing they do has been decreasing. Much software is developed in-house, and transcription, which accounts for a large percentage of outsourcing, is decreasing in general and will eventually take care of itself. There were also questions raised about electronic devices taking over nursing tasks, as well as security concerns since these devices are connected to the corporate network. Darren replied that this is one of his favorite "scare stories." FDA regulation is mostly beneficial, but because security is an iterative process, you cannot get devices patched easily. One day this shortcoming is going to be acknowledged, and that will fundamentally change the way devices are developed.

WORK-IN-PROGRESS REPORTS

Summarized by Kevin Borders (kevin.borders@gmail.com)

■ **Detecting Injected TCP Reset Packets**

Nick Weaver

Reset injectors need to send multiple reset packets to be reliable, and they can be detected for this reason. P2P blocking has been witnessed with Sandvine, an Israeli ISP, and others that were identifiable based on signatures, such as anomalies in sequence and ACK number increments between reset packets. Weaver's techniques can be used to detect DNS spoofing, ARP poisoning, and other attacks. As a separate "WIP-Let," Weaver discussed potential intrusion-detection policies for stopping DNS packet injection for the recently announced vulnerability which involve alerting in response to two conflicting DNS responses.

■ **ROFL: Routing as the Firewall Layer**

Steve Bellovin

The idea is to take the port number, append it to the IP address, and route to the entire /48 address. In this way, packets are dropped early (in the routing layer) instead of at the

endpoint firewall. There are some disadvantages, however, including routing table size and a hacker's ability to map services without a scan.

- ***A Web Without the Same-Origin Policy***

Francis Hsu

The same-origin policy gets in the way of some Web applications and allows too much access for Web applications located on the same domain. To address this problem, Hsu proposes blocking access to everything, and then treating pages as objects to enable necessary interaction between them.

- ***The Cost of Free Calls: Identifying Accents in Encrypted Skype Traffic***

Paul DiOrio

Despite encryption, you can extract a fair amount of information from VoIP traffic based on variable bit rates. Previous research shows that the language and specific phrases can be identified in this manner. DiOrio's research looks at detecting different accents based on the encoding bit rate. Preliminary results show that the average accuracy of differentiating accent pairs is 73%, with the best being Italian/Japanese at 91% accuracy.

- ***Mementos, a Secure Platform for Batteryless Pervasive Computing***

Benjamin Ransford

Batteryless computing is hard; you have no battery, little time to compute, and very few resources. The goal of Mementos is to enable long-running computations on batteryless devices. The idea is to execute a little bit and then write results to nonvolatile storage before losing power.

- ***Debian, OpenSSL, and SSL Certificates***

Hovav Shacham

There was a bug in OpenSSL where the Debian folks accidentally removed code to generate entropy, leading to keys in a 32,000-value space based solely on process ID. Shacham conducted a survey of SSL keys installed on popular Internet sites four days after the vulnerability was disclosed and found that 279 out of 43,491 certificates contained bad keys, including many key collisions.

- ***An Enhancement of Windows Device Driver Debugging Mechanism for VMM-based Live Forensics***

Andy Ruo

There is no direct way to transfer information between a virtual machine and the host operating system. Ruo is working on a system for mapping certain regions of memory from the guest VM to the host OS to enhance debugging.

- ***Botnet Enumeration: The Nugache Case***

Sven Dietrich

Nugache is a bot that uses peer-to-peer communication with encryption for command and control. Dietrich queried Nugache bots for information about connected peers, version, etc. One particularly interesting result was that the

number of reachable nodes on the botnet declined significantly following each "patch Tuesday" every month.

The accepted WiPs abstracts can be found at <http://www.usenix.org/events/sec08/wips.html>.