conference reports

THANKS TO OUR SUMMARIZERS

18th USENIX Security Symposium

Montreal, Canada
August 10–14, 2009

OPENING REMARKS AND AWARDS

Summarized by Rik Farrow

Fabian Monrose began with the statistics: 176 papers were submitted to the symposium. One was withdrawn and three rejected for double or triple submissions to conferences (resulting in the papers being automatically rejected from all the conferences). Three or more people reviewed each paper, with Monrose reading the vast majority of these papers. Many submissions were being edited right up to the deadline. By the time of the PC meeting, there were only 62 papers left to discuss. All PC members attended the meeting in Chapel Hill, and Monrose said there were some real battles there (beside the NCAA tournament). By the end of the meeting, 26 papers were accepted.

There were 84 applications for student grant support; the USENIX Board provided $50,000. The two Outstanding Student Paper awards were "Compromising Electromagnetic Emanations of Wired and Wireless Keyboards" (Martin Vuagnoux and Sylvain Pasini, LASEC/EPFL) and "Vanish: Increasing Data Privacy with Self-Destructing Data" (Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy, University of Washington).

KEYNOTE ADDRESS

- ***Android: Securing a Mobile Platform from the Ground Up***
  Rich Cannings, Android Security Leader, Google

  Summarized by Italo Dacosta (idacosta@gatech.edu)

With the increase in the adoption of smartphones as well as in our reliance on these devices, they will undoubtedly become the next target of cybercriminals. This makes the security of the mobile operating systems an area of critical importance. Rich Cannings described the main features of Android, Google's open source mobile OS, and pointed out that its openness differentiates Android from other popular mobile OSes. In Android, any user can develop applications, because there is no centralized software signing authority, but this openness also makes Android more vulnerable to malicious software. Being aware of this risk, Android developers follow a security strategy based on four components—prevent, minimize, detect, and react—to protect Android's core components, applications, and user data.

To prevent possible attacks based on the exploitation of unknown vulnerabilities, Android has partnered with security experts to target high risk areas such as remote attacks and vulnerabilities in media codecs. Being

an open source OS, Android does not rely on obscurity techniques for its security. In addition, well-known security mechanisms such as stack overflow protection (ProPolice) and heap protection (dlmalloc) have been implemented. Address Space Layout Randomization (ASLR) has not been implemented yet due to some platform constraints but is expected to be added in the future.

Cannings said that not only is it unfeasible to prevent all the possible security vulnerabilities in Android, but attackers do not always even need vulnerabilities to compromise an OS; social engineering techniques and bugs can also be used to install malware. Therefore, it is important to minimize the impact of compromised applications. For this, Android tries to extend the Web security model to the OS, using an application sandbox model for separation of privileges (each application runs with its own UID and virtual machine). Applications are locked down to their minimal functionality, and permissions are required to grant more access to resources, a whitelist model. Users decide to give permissions or not to the applications when they are installed. A challenge in this area is to determine the right number of permissions that should be asked of the user (granularity) because too many questions could cause the user to ignore this mechanism. In addition, media codec libraries are given lesser privileges than in other OSes, given the long history of vulnerabilities in media codecs.

To detect attacks, Android uses activities such as developer education, code audits, fuzzing tests, and honeypots. Because Android is an open source OS, anybody can detect and report security problems: users, developers, security researchers, etc. External reports from members of the security community have helped Android's developers fix several security problems. Also, users are encouraged to report suspicious applications in the Android Market. Reported applications are analyzed by Android personnel and removed from the Market if they are considered malicious.

Android relies on auto-updates to distributed security patches to fix critical security vulnerabilities. Android uses an over-the-air update system where user interaction is optional and no additional cables or computers are required, resulting in a high update rate. However, the main challenge to apply security updates is the testing of the updates and the coordination with different mobile network providers. For mobile carriers, updates are a concern because they could affect the availability of a great number of devices, resulting in financial and customer service problems. Therefore, before being released, security updates should be carefully tested and approved by each mobile carrier, but this process can delay the release of the update considerably.

During the Q&A, Rik Farrow asked about the prevention of privilege escalation attacks. Cannings answered that one way to mitigate this type of attack is to reduce the number of processes running with root privileges. In Android, only ping and zygote (the application launcher) run with root privileges. Regarding the support of security hardware mechanisms, Cannings commented that they are evaluating the use of the execution prevention bit and other hardware mechanisms. How many of Android's 5 million lines of code were written in type-safe language? Most of the Android code is written in Java, not only for security but also for compatibility purposes. Finally, Gary McGraw asked what percentage of the vulnerabilities discovered in Android were discovered externally versus internally. The number of vulnerabilities discovered internally was several orders of magnitude greater than those discovered externally.

ATTACKS ON PRIVACY

Summarized by Shane Clark (ssclark@cs.umass.edu)

- ***Compromising Electromagnetic Emanations of Wired and Wireless Keyboards***
  Martin Vuagnoux and Sylvain Pasini, LASEC/EPFL

  Awarded Best Paper!

According to Martin Vuagnoux, the authors chose the keyboard as an attack vector because it is the first device in a system that handles sensitive data such as passwords electronically; security is not a priority in their design. They chose to examine electromagnetic emanations because many other attack vectors for I/O devices have been demonstrated in the past, such as electromagnetic leakage from displays and acoustic emanation from keyboards.

To provide background for the work, Vuagnoux next introduced radiative emanations and their capture. Radiative emanations are those requiring the source to act as an antenna. These are the emanations of interest for attacks, as others require physical contact with a wire. Radiative emanations can be further broken down into direct emanations (those caused by a keypress or other action) and indirect emanations (those from carrier signals, modulation schemes, etc). To observe these emanations, the authors chose to attempt to capture the entire spectrum of interest simultaneously in order to capture the maximum amount of information without scanning. They found that they were able to achieve this using a large conical antenna and a 5 GSa/s oscilloscope. After capturing the signals, they examined the Fourier transform of each to identify interesting characteristics.

Vuagnoux moved on from background and acquisition methodology to describe three attacks that are effective only against PS/2 keyboards. The first attack relies on the fact that PS/2 keyboards modulate a scan code onto a clock signal by pulling down a data line repeatedly. It is possible to exploit this direct emanation by observing the series of falling edges that this creates in the modulated signal, but this results in aliasing among the scan codes. The authors constructed a table of all keys and their corresponding signatures, which allowed them to reconstruct words typed based on the sequence of key presses. The second attack simply filters the same information and computes a distinct threshold in order to remove aliasing. The final PS/2 attack actually demodulates the captured signal in order to remove noise from the clock. The only USB attack relies on

the use of a matrix scan loop to poll pressed keys. Which key has been pressed can be discerned from the delay associated with scan reporting. This attack is also effective against PS/2 and wireless keyboards. All of the attacks were effective in realistic environments, including through walls, though at ranges sometimes less than a few meters. Surprisingly, all attacks were much more effective when tested in an apartment building, owing to construction features such as common grounds and water pipes.

Xiaofeng Wang (IBM) asked what the implications are of multiple users typing in the same space simultaneously. Vuagnoux responded that they were able to fingerprint individual keyboard models based on clock signal differences. Perry Metzger asked a similar question, emphasizing the problem of separating users whose input is captured simultaneously, to which Vuagnoux responded that this should be possible in theory based on keyboard fingerprinting, but presented a technical problem because they were unable to actually trigger data capture accurately with current hardware. This is something that the authors are still working on. Had the authors attempted similar attacks against mice? They hadn't and, in fact, removed mice during their experiments to minimize noise. Had the authors attempted any countermeasures, such as wrapping a keyboard in shielding material? A few keyboards implement shielding based on the results of the TEMPEST program, but they cost hundreds of dollars and the authors were not able to purchase one without a military affiliation. Attempts by the authors to shield a keyboard themselves sometimes resulted in more visible emanations.

- ***Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User Systems***
  Kehuan Zhang and XiaoFeng Wang, Indiana University, Bloomington

Kehuan Zhang reported a new shared information vulnerability present on multi-user UNIX-like systems and presented an example attack on Linux. Zhang started by introducing legitimate uses of shared information on Linux systems and procfs, the mechanism for this sharing. It is common for users on a system to run commands such as top in order to see which users are logged in to a system, what processes are running, and what resources they are consuming. This information-sharing is enabled by the process file system, procfs, which is a pseudo file system that is globally readable. It contains per-process data such as image name, starting address of the stack, and current stack pointer (ESP).

Zhang next discussed the attack, which performs keystroke inference using the information available in procfs. Before the attack can be mounted, the attacker must analyze the victim process offline and build a trace of the ESP variation in procfs as a result of user input. The attack also requires a multi-user system, the ability to execute programs, and a multicore CPU. These capabilities are necessary because the attacker must run a shadow process concurrently with the victim process in order to observe changes in procfs. The shadow process produces a partial ESP trace (as it is not possible to catch all changes reliably), which is then converted into a longest common subsequence problem in order to extract keystroke timings. The timings are then given as input to a Hidden Markov Model (HMM) to perform key inference. Multiple timing traces are produced in order to increase HMM accuracy.

Zhang concluded by presenting performance results and discussing countermeasures. He noted that the percentage of keystrokes detected decreases rapidly as CPU usage increases for some applications, but with CPU usage under 5%, all of the tested applications were vulnerable. Zhang showed server traces indicating that three test machines averaged under 4% CPU usage, to illustrate the feasibility of the attack on real-world systems. He next presented results for password inference using 50 keystroke captures. The authors' keystroke inference system was able to reduce the password search space to between 0.05% and 7.8% of the initial space. Zhang noted that a kernel patch to remove the compromising information leakage is the short-term solution, but suggested a complete evaluation of information leakage through shared information channels. Someone asked if this attack can be used to capture SSH keys, and Wang answered that it can.

- ***A Practical Congestion Attack on Tor Using Long Paths***
  Nathan S. Evans, University of Denver; Roger Dingledine, The Tor Project; Christian Grothoff, University of Denver

Nathan Evans gave a talk on a new Tor attack which allows the attacker to determine the path data travels through the network. The Tor system is the most popular free software used to achieve anonymity on the Internet. Tor uses *onion routing*, which forwards data through the network, peeling off a layer of encryption at each node. Each node in the network knows only the previous hop and the next hop. This is a key security goal for Tor, as the discovery of a complete circuit through the network makes it easier to deanonymize the originator of the traffic. Evans noted three design choices made by the Tor project that are relevant to his attack. First, no artificial delays are induced on any connection. Second, path length is set at a small finite number (3). Third, paths of arbitrary length through the network can be constructed.

Evans described the attack and countermeasures in more detail. The attacker must first operate a Tor exit node that is in use by the victim. Next, the attacker uses a malicious client to create a long loop in the network before connecting to the requested server. This allows the attacker to load the intermediate nodes as desired. Finally, the exit node injects a JavaScript ping command into the traffic that reports back to the malicious client and is used to measure the latency along the circuit as the attacker loads possible first hop routers. Based on the observed latency, the attacker can determine which node is the first hop. Since the attacker also operates the exit node, she can determine what server the victim is connecting to. Evans showed that attack runs are clearly distinguishable from normal Tor traffic in testing and that the attack is effective even against high bandwidth

routers. Finally, he presented several possible countermeasures, including the prohibition of infinite path lengths, which the Tor developers have implemented.

INVITED TALK

- ***The Building Security in Maturity Model (BSIMM)***
  Gary McGraw, CTO, Cigital, Inc., and Brian Chess, Chief Scientist, Fortify Software

  Summarized by Salvatore Guarnieri (sammyg@cs.washington.edu)

The Building Security in Maturity Model (BSIMM, http://bsi-mm.com/) ranks your corporation's security practices against those of other corporations. This work differs largely from previous work in that it does not advocate security practices based on what seems like a good idea; it doesn't actually recommend anything. The model simply compares corporations' security practices. It is up to the users of the model to determine if they want to be like the organizations they are being compared to.

BSIMM is based on a study of nine large companies: Adobe, Depository Trust and Clearing Corporation (DTCC), EMC, Google, Microsoft, QUALCOMM, Wells Fargo, and two anonymous companies. BSIMM analyzed what these corporations were doing for software security and found some expected and some unexpected results. Two basic and expected findings were that security was an emergent property of the entire system and that secure software requires deep integration with the Security Development LifeCycle (SDLC).

Since BSIMM is a model that compares company practices, one would think that the companies one is compared to would be important. For example, an independent software vendor (ISV) would have different security concerns and practices from those of a financial institution. The BSIMM study actually found that this is not the case. Financial institutions and ISVs have approximately the same software security model. Additionally, the size of the companies in the study ranged from hundreds to thousands of software developers. In all the companies, the size of the Software Security Group (SSG) was 1% of the total software developers. This doesn't mean that the correct SSG size is 1% of developers, but if you like the security of these nine companies, maybe 1% is a pretty good target size for your SSG.

The model is a set of over 100 activities. You mark which activities your company does and then compare your results to the average of the nine studied companies. Each activity has a ranking associated with it that describes how easy it is to do. This ranking is also interpreted as how serious or mature a company is in a certain area of security. The end result is a simple comparison, but the speakers have developed a visualization that easily shows how one organization compares to the average. The model is available from the BSIMM Web site for free under a creative commons license.

There were a few surprising discoveries from the study, including the top 10 most unexpected results. These are all available on the Web site, but there were a few very interesting things that everybody does. First, everyone is doing code review, using tools and, most importantly, looking for ways to automate the process. Second, SSGs do architectural analysis. Architectural analysis is difficult, and product teams have a hard time doing it, so SSGs need to help out. Third, every organization has an SSG, but each one had a different way of starting its SSG.

More companies need to be studied. Nine is a good starting point, but few statistics are valid with only nine data points. They are already up to 17 companies with their current work and they keep looking to expand. As they get more companies, they can start to say more interesting things, such as comparing big companies to small companies.

MEMORY SAFETY

Summarized by Stephen McLaughlin (smclaugh@cse.psu.edu)

- ***Baggy Bounds Checking: An Efficient and Backwards-Compatible Defense against Out-of-Bounds Errors***
  Periklis Akritidis, Computer Laboratory, University of Cambridge; Manuel Costa and Miguel Castro, Microsoft Research, Cambridge; Steven Hand, Computer Laboratory, University of Cambridge

Periklis Akritidis described a technique called baggy bounds checking, which aims at increasing the efficiency of array bounds checking. Because type-unsafe languages such as C do not perform array bounds checking, previous research efforts have been made to add it to the language. Traditional backwards-compatible techniques (e.g., splay trees) require several memory accesses per check or use too much memory. To address this issue, the authors suggest allocating strategically sized buffers to make bounds checks more efficient.

The presented technique, baggy bounds checking (BBC), pads objects upon allocation to a size that is a power of two. Bounds checking is then performed, not on the object boundaries but on the allocation boundaries, which can be calculated from a pointer into the object and a single table lookup. Because BBC partitions memory into slots that are powers of two in size, the base address of an allocation can be found by clearing the lowest-order lg(size) bits of a pointer to an object, where lg is the log base two. A pointer to an object is used to index a global array that contains the log base two of the size of the containing slot, requiring only one byte to track the bounds of each allocation. Of course, this technique cannot detect memory accesses that are within an allocation but outside an object. This is not a problem, as the padded regions are cleared upon allocation to remove any sensitive data from previous allocations.

BBC is implemented as a compiler extension that works with the intermediate representation of a C program to modify memory allocation and add bounds checks. Heap allocation is modified to use buddy allocation at runtime, while globals are modified at compile time and the heap

allocator is modified in the library. BBC was evaluated for memory and performance overhead using the Olden and SPECINT 2000 benchmark suites on Windows. Each benchmark was compiled using both BBC and splay tree bounds checking. Most surprisingly, BBC had a smaller average memory overhead than splay tree checking on both suites, although the splay tree did slightly better on most SPEC benchmarks. On the Olden tests, the splay tree version created 170% memory overhead, while BBC sometimes performed better than the default Windows allocator. Both of these effects are a result of the metadata overhead caused by the Olden benchmark's many small allocations. The ability of BBC to detect bounds errors was tested utilizing 18 buffer overflows from a suite of benchmarks used to test for memory errors. BBC detected 17 out of 18 errors. In the one exception, an array was allocated inside a structure. An out of bounds access to the array caused another pointer in the structure to be overwritten.

Someone asked how this scheme differed from SoftBound, which was presented at PLDI 2009. Akritidis said that SoftBound requires eight bytes to be stored for each pointer, causing high memory overheads.

- ***Dynamic Test Generation to Find Integer Bugs in x86 Binary Linux Programs***
  David Molnar, Xue Cong Li, and David A. Wagner, University of California, Berkeley

Dave Molnar presented work on generating better test cases for finding integer bugs with fuzz testing and compared it to a black box fuzz tester running in Amazon's Elastic Compute Cloud. A large number of software errors are caused by integer bugs such as over- and underflows, non-value preserving conversions, and signed and unsigned conversion errors. Typical black box fuzz testing does not deal with integer bugs, which may only occur for particular integer values. Molnar described SmartFuzz, a fuzz test generation tool that uses constraint solving to more quickly find inputs that should cause a program to crash.

SmartFuzz performs a symbolic execution of the program under test, yielding a set of constraints on integer variables. These constraints may then be solved to determine the set of inputs that should either be rejected or trigger a bug. In order to generate constraints on signedness, SmartFuzz uses a four-type system in which an integer is either unknown, signed, unsigned, or bottom. If at some point in execution the inferred type of a variable is bottom, SmartFuzz will search for a constraint to assign a negative value to that variable to test for a signed/unsigned conversion bug.

SmartFuzz gives fuzzed inputs to programs running in Valgrind, which will detect any memory errors caused by fuzzing. While effective at determining whether an input causes a bug, this use of Valgrind results in different test cases discovering the same bug and different bugs being discovered by the same test case. This results in multiple reports being filed for the same bug on different test cases. The solution Molnar presented is a fuzzy stack hash which maps the first three frames of a stack trace to a bucket for a single bug. Then a single report is generated for each bucket.

MetaFuzz is run in the Amazon Elastic Compute Cloud (EC2), where CPU time can be rented for 10 cents per hour. The metric used for evaluating fuzz testers in this environment is dollars spent per bug found. The evaluation compares SmartFuzz against zzuf, a block box fuzz tester. The two fuzz testers were run against six programs: mplayer, ffmpeg, convert, gzip, bzip2, and exiv2. SmartFuzz achieved a lower cost per bug than zzuf on two out of six programs and found two bugs in gzip, in which zzuf found none. The metafuzz framework can be accessed at http://metafuzz.com.

Someone asked why the black box fuzzer, zzuf, found more bugs than SmartFuzz on four of the six programs tested. Molnar explained that this is because zzuf changes much more of the fuzzed inputs between tests. This will find more bugs in unrefined code, whereas SmartFuzz is aimed at finding more obscure bugs in mature code. When were most bugs found? Molnar said, early on in the process. How difficult does the analysis become for programs that require complex inputs and user interaction? There was no relationship between complexity of inputs and difficulty, but there are engineering issues that need to be overcome to fully automate the testing of interactive programs.

- ***Nozzle: A Defense Against Heap-spraying Code Injection Attacks***
  Paruj Ratanaworabhan, Cornell University; Benjamin Livshits and Benjamin Zorn, Microsoft Research

Ben Zorn described NOZZLE, a software detection method for heap-spraying attacks. Heap spraying is a method for achieving code execution in the face of address space layout randomization (ASLR). The goal is to place many instances of malicious code on the heap, then jump to some address in the heap with the injected code. Many instances are needed, as ASLR prevents the calculation of the correct address of the malicious code.

NOZZLE provides protection against JavaScript-based heap-spraying attacks in which the malicious code is placed on the heap through the allocation of objects, usually strings. NOZZLE does this by inspecting objects on the heap to determine if they seem malicious (e.g., if they contain a no-op sled, a long series of no-op instructions that lead to executable code). If a percentage of objects are malicious beyond a certain threshold, the offending script is stopped. The simplest way to check if an object is malicious is to check for the presence of a no-op sled, but this technique produces too high a false positive rate. Instead, an object is marked as malicious if it contains a sequence of instructions that looks sufficiently like executable code. This is a hard task in itself, as virtually any byte sequence can be interpreted as x86 instructions.

To detect executable code, NOZZLE uses program flow analysis on objects to determine their attack surface area (SA). The SA of each potential code block in an object is the likelihood that the block is reachable if execution occurs in

its containing object. The surface area is then propagated throughout the control flow. Blocks that contain invalid opcodes, such as those that must be executed in kernel mode, have zero SA. NOZZLE exhibits zero false positives and zero false negatives when tested on the 150 Web sites and 12 known heap-spraying attacks, respectively. Note that this is with a 100% sampling rate. In the case of full sampling, NOZZLE causes a maximum overhead of two times normal page-load time, and around 5–10% overhead with a 5% sampling rate. Zorn concluded the talk with a live demonstration in which NOZZLE successfully detected heap spraying.

Avi Rubin pointed out potential means for circumventing NOZZLE, including runtime-initialized objects and code obfuscation with junk data. Zorn pointed out that jumps to code in different objects is another way to trick the surface area calculation, and that they are exploring mitigations for all of the described escalations. Adam Barth (UCB) asked whether NOZZLE would be effective against a less aggressive heap-spraying attack in which only 10% of objects are malicious. Zorn explained that NOZZLE would not detect such an attack, as the malicious surface area is too small, and that NOZZLE is not effective if an attacker is willing to settle for a low success rate.

INVITED TALK

- ***Toward a New Legal Framework for Cybersecurity***
  Deirdre K. Mulligan, School of Information, University of California, Berkeley

  Summarized by John Brattin (jbrattin@student.umass.edu)

Deirdre Mulligan spoke about the difficulties involved in designing laws to help protect end users from cyber-attacks. Her main ideas were: a public health analogy may be fitting, and a new legal framework for cybersecurity could benefit from this approach; by using tactics such as mandatory information disclosure, the law could be more flexible than technical standards in regulating software; and because we have a participatory government, people with computer security expertise should get involved in helping design a new legal framework for cybersecurity.

Lack of adoption is a major problem in computer security. There's no point in coming up with new, stronger security practices if no one will bother to use them. "Security in the marketplace is remarkably below what known best practices could provide." In many cases, it isn't even a technical problem—we have the theories, and we even have the theories implemented in software, but people choose not to use the software. Many people use virus protection software but don't update definitions. Many people don't download critical security patches.

Mulligan notes that law is a somewhat unpopular channel for effecting change in the cybersecurity community. People think law moves too slowly, lagging significantly behind changes in technology. Currently, cybersecurity law focuses on deterrence: "increasing the celerity, severity, or certainty of punishment for criminal activity." However, certainty of punishment is remarkably difficult to increase, as cybercriminals are notoriously difficult to identify and, once identified, are frequently not under our jurisdiction. For these reasons, deterrence seems like a poor choice of policy.

Another option is to "incentivize the good guys" to use more secure practices. One way we could get developers to use secure practices is by using notification laws: when a company emits a large volume of toxic waste, they must report it to the government, and eventually the information becomes public. This may lead companies to limit emissions in order to avoid bad publicity. This "mandatory reporting" method prevents the government from directly interfering, but creates incentives for developers to address security concerns.

Cybersecurity will continue to be an important issue; as technology changes and improves, so, too, do technological attacks. We will never completely stop these attacks. We also put ourselves at risk by having an open flow of communication—just as you can defend yourself from biological viruses by staying in your house, you can defend yourself from computer viruses by avoiding the Internet. Another parallel between public health and computer health is that viruses spread in a monoculture. If we had more diversity in systems, particularly operating systems, viruses might not spread as easily.

Public law is a useful channel through which to combat cybercrime, primarily because it is more flexible than using rigid technical standards. By using public law, we can guarantee that a certain problem is addressed by software, or we can guarantee information disclosure that results in greater security, without enforcing the use of any particular system that may quickly become outdated. The law gives another layer of abstraction, in essence. However, people with technical know-how should participate in the construction of appropriate laws.

A member of the audience suggested that people don't patch because patching requires the user to restart, which is time-consuming. She also suggested that it is difficult to enforce security when there aren't clear standards. Mulligan proposed a checklist strategy: if a developer has in some way addressed every problem on the list, she's done her job. Another strategy would be to let developers come up with their own standards and merely report when those standards are violated. Another audience member noted that although diversity may slow the spread of viruses, software becomes more useful the more people use it. He then asked if the government should somehow regulate the development of patches, to make sure nothing breaks. Mulligan expressed doubt that the government would ever interfere so directly in development.

NETWORK SECURITY

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

- ***Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine***
  Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, and Alexander G. Gray, Georgia Tech; Sven Krasser, McAfee, Inc.

Shuang Hao spoke about SNARE, a system that uses network-level (as opposed to content-level) features of email transmissions to detect spam. Hao cited familiar figures on the cost of worker productivity lost to spam and the prevalence of spam; he also pointed out that spam is increasingly being used as a vector for malware. He stressed the high cost of content filtering in terms of both network traffic and human time. IP blacklists, commonly used in addition to content filtering, are incomplete, and many spam senders are newly infected machines without reputations. The authors' approach in SNARE incorporates empirically derived heuristics to classify email transmissions, given as little as a single packet. Hao asserted that network-level features may be cheaper to analyze than content-level features, because they require less input data; they may also be more difficult for a spammer to vary. Hao offered the basic intuition that, because over 75% of spam is thought to come from botnets, the sending patterns of spammers should be distinguishable from those of human non-spammers.

SNARE uses a set of thirteen features to classify spam. Hao focused on five heuristics that classify messages based on the receipt of a single packet (and, in some cases, using auxiliary knowledge gained from previous interactions with the sender). First, Hao asserted that most legitimate email does not need to travel a long distance geographically from the sender's computer to the recipient's; according to their data set, 90% of legitimate messages travel 2,500 miles or less. Second, because clients participating in botnets tend to share network space with other botnet participants, spam often arrives directly via IP addresses that are numerically close to others that have submitted messages; legitimate messages tend not to exhibit this pattern. Third, legitimate senders and spammers exhibit different sending patterns throughout the day, with spam traffic peaking slightly later than legitimate traffic. Fourth, while legitimate email tends to arrive via mail servers that listen on standard mail-related ports, spam does not; Hao noted that 90% of the spam senders in their data set had none of the standard mail-related ports open. Finally, because some ISPs are more spam-friendly than others—the top 20 ASes in their data set hosted 42% of the spamming IP addresses—the sender's AS may provide a clue to the legitimacy of the message. Hao concluded that SNARE is capable of providing an effective first line of defense against spammers.

George Jones asked whether SNARE consulted lists of known dynamically assigned IP addresses; Hao answered that it did not. Had the authors considered ways to improve on the linear function used to score messages? They considered improvements to the classifier a separate problem. Michael Sirivianos expressed doubt that the geodesic distance feature was equally valid for non-US recipients; Hao agreed that different regions might exhibit different characteristics, and he remarked that the use of several different features made SNARE more robust. An audience member asked which of the classifying features was the best predictor of spam in the authors' experiments. Hao replied that it was the sender's AS number.

- ***Improving Tor using a TCP-over-DTLS Tunnel***
  Joel Reardon, Google Switzerland GmbH; Ian Goldberg, University of Waterloo

Joel Reardon spoke about a way to improve the performance of Tor. Reardon introduced Tor as an overlay system that grants anonymity to anyone on the Internet, most importantly to people who are subject to Internet censorship. The authors propose a change to the way in which Tor routers handle concurrent connections; their change reduces packet delivery latency and, according to the authors, makes Tor more usable.

The authors studied latency in Tor in an attempt to find bottlenecks. Reardon remarked that communication delays—that is, those imposed by throughput limitations—were negligible compared to overall latency. By running a Tor node at a university and exchanging several pieces of data, they eventually found a bottleneck in the buffering strategy Tor uses to multiplex connections. While input buffers drained quickly, output buffers occasionally required packets to wait a long time to be sent. Because Tor uses one socket per router-router link and because the underlying asynchronous communication library, libevent, waits to send on a socket until the operation is guaranteed not to block, data queues up in the output buffers waiting for the socket to become writable. The authors investigated further and found that TCP congestion control was the primary cause of such blocking: if circuits A and B are multiplexed along a link E, then congestion control on E will affect A and B regardless of the respective traffic on each. Reardon showed several graphs illustrating how output buffers on a Tor router changed over time. As an alternative to multiplexing, the authors implemented a scheme in which each circuit that traversed two routers received its own TCP connection between the routers. To avoid several problems (e.g., information leaks) with using TCP directly, the authors tunneled TCP over UDP streams with Datagram Transport Layer Security (DTLS). To prevent clients having to modify their kernels, the authors implemented a user-space TCP stack that can assemble packets suitable for sending via DTLS. Each router advertises a single UDP socket that multiplexes data for all incoming connections; congestion control is performed on a per-circuit basis in the user-space TCP stack. Reardon showed performance graphs demonstrating that Tor with TCP-over-DTLS exhibits much less latency under load than unmodified Tor. Reardon discussed future work involving Tor's new user-space TCP stack and rethinking Tor's buffering strategy.

Michael Sirivianos asked whether it would make sense to make Tor an IP-level service with congestion control only at the entry and exit nodes. Reardon remarked that such a strategy would decrease throughput, because congestion control does not work well on long paths.

- ***Locating Prefix Hijackers using LOCK***
  Tongqing Qiu, Georgia Tech; Lusheng Ji, Dan Pei, and Jia Wang, AT&T Labs—Research; Jun (Jim) Xu, Georgia Tech; Hitesh Ballani, Cornell University

Tongqing Qiu spoke about LOCK, a system that locates IP prefix hijackers by using PlanetLab to monitor traffic to hijacked networks. The Internet comprises tens of thousands of networks (called autonomous systems, or ASes) that exchange packets according to an inter-network routing protocol called the Border Gateway Protocol (BGP). BGP's lack of authentication allows any AS to announce ownership of any other AS, which means any network can hijack, or steal, another network's traffic. Qiu described three kinds of hijacking: blackholing, in which an attacker drops all traffic destined for the victim; imposture, in which the attacker pretends to be the victim such that the victim never receives the hijacked traffic; and interception, in which the attacker transparently interposes her own AS into the chain of networks leading to the victim. Previous approaches to the problem of prefix hijacking have been stymied by the difficulty of changing the Internet's routing infrastructure or have otherwise been focused on recovering the network without pinpointing the source of the error. Qiu claimed that his team's work was the first study of the hijacker location problem. Their system, called LOCK, aims to locate hijackers automatically in order to minimize the effort required for mitigation and recovery.

LOCK uses monitoring software on PlanetLab nodes distributed around the world. Given a network prefix P (owned by a specific AS) to monitor, LOCK's constituent nodes periodically observe the AS paths from their own networks to P. If some monitoring nodes detect that their respective paths to P have changed—Qiu called such nodes "polluted"—they follow an algorithm that infers the location of the hijacker. The algorithm finds the ASes within one hop of the prefix P in a public database of AS relationships. It then considers all the neighbors of ASes on the new paths to P. Because a hijacker cannot manipulate the path to her own AS that traverses her upstream providers, her AS will appear in the set of neighbors, so the algorithm restricts its search to that set. Because paths from polluted monitors (which are distributed diversely around the Internet) to the hijacker naturally converge around the attacker's AS, simply ranking the ASes in the neighbor set by the number of times they appear in paths from polluted monitors allows a quick whittling of the search space. Qiu showed experimental evidence that, for real and synthetic hijacking events, LOCK correctly ranked the hijacker's AS in the top spot up to 94.3% of the time.

George Jones remarked that, although LOCK's detection mechanisms appear sound, the design sidesteps the basic issue that no authoritative central registry of AS relationships is kept up-to-date, thereby making it impossible to determine with total certainty whether a new AS announcement is good or bad. Qiu agreed that the lack of a central registry was a fundamental problem. An audience member asked whether an attacker couldn't simply prepend arbitrary AS numbers into the AS path it announces, and whether that affected LOCK's ability to infer the hijacker's neighborhood. Qiu remarked that some existing routers implement sanity checks that would flag such announcements, but agreed that a hijacker might be able to foil the neighborhood inference by including arbitrary AS numbers in its announcement.

INVITED TALK

- ***Modern Exploitation and Memory Protection Bypasses***
  Alexander Sotirov, Independent Security Researcher

  Summarized by Martim Carbone (mcarbone@cc.gatech.edu)

Security researcher Alexander Sotirov, well known for his work on offensive techniques, gave a very instructive talk on the past, present, and future of memory exploitation. His talk also covered the other side of the game by analyzing the evolution of countermeasures, from virtual inexistence to techniques such as Data Execution Prevention and Address Space Layout Randomization (ASLR). Overall, his presentation gave useful information and insight to the audience on the nature of this arms race that has been going on for many years and shows no sign of stopping.

Sotirov started by introducing some basic concepts, such as what exactly constitutes a memory corruption vulnerability and an exploit. The latter is defined as a way to make the target process execute arbitrary code by exploiting the vulnerability. Although these two concepts are commonly coupled, the difficulty of finding a vulnerability and that of effectively exploiting it are not closely related. Sotirov explained that the distance between the two has varied over time, as our understanding of memory exploitation and countermeasures has increased.

In the late '90s, finding a vulnerability could be a hard task, but once found, building a working exploit for it was trivial, given the absence of mitigations. As time passed, techniques for finding new vulnerabilities were systematically improved, reaching a climax in the summer of 2004. At that time, several classes of vulnerabilities were known with no effective mechanisms to counteract them, along with effective techniques for automatically finding such vulnerabilities, such as fuzzing. Examples mentioned by Sotirov include the infamous stack overflow, structured exception handler (SEH), heap overflow, and format string exploitation techniques. As he pointed out, all these exploitation techniques relied on assumptions made about the target process's execution environment. Examples are the fixed locations of code and data regions in memory, a well-known stack layout, and the fact that data placed on a program's stack/heap can be executed as code. The "golden age" of exploitation came to an end as operating systems started

being shipped with some basic mitigation mechanisms that invalidated some of these assumptions.

Windows XP Service Pack 2, released in August of 2004, was the first attempt at mitigation and included support for features like disallowing code execution at the stack and heap of programs by leveraging new hardware support (the NX bit), safe heap header unlinking to prevent using unlinking to control execution flow, and stack cookies. Sotirov explained some of the workarounds that were developed to counteract these mitigation techniques and the other mitigation techniques that were developed in response. For example, a simple way to circumvent stack cookies was to no longer rely on overwriting the return address but to use other variables and function arguments. In response to this, compilers started supporting variable reordering in the stack by placing all the buffers at the end of the local stack frame. However, exploitation was still possible by overflowing into other buffers or even the stack frame of the calling function. Exploitation and mitigation techniques involving SEHs were also discussed.

Two state-of-the art mitigation techniques were given special focus in the presentation: Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR). The first effectively closes the window for code injection attacks by disallowing code execution in the program's data regions. The second operates by loading executables at randomized locations in virtual memory, preventing an exploit from correctly guessing the address of its payload. Sotirov explained that DEP and ASLR are only useful when deployed together. Alone, DEP can be circumvented through return-into-libc-style attacks and return-oriented programming, techniques that execute the program's own code in a controlled manner for malicious purposes. If ASLR is used alone, exploitation is still possible by using "heap spraying," which fills the program's heap with copies of the malicious payload, to the point that an exploit writer can say for sure that a certain address will contain a copy of it, despite the randomization. Although included in Windows Vista, these two mitigation techniques had limited impact, since DEP was disabled by default and ASLR was only used for a small set of system services. It is expected that Windows 7's implementation of DEP and ASLR will have much better support for third-party applications.

Due to these new mitigation techniques, the situation at the moment is the opposite of that of the late '90s: finding vulnerabilities has become a relatively easy task, whereas exploiting them now sometimes requires many man-months of hard work, according to Sotirov. In light of this, he moved on to discuss new possibilities in memory exploitation as well as interesting research directions. These include the development of techniques to disclose memory content, which would allow ASLR to be bypassed, as the secrecy of a program's location in memory would be lost. Another one relies on partially overwriting the low-order bytes of pointers, giving an exploit access to the region of the address space occupied by the target process. This works because ASLR randomizes only the 16 high-level bits of addresses, i.e., programs are still 64K-aligned. Entropy attacks against ASLR are also possible. In these, an exploit is executed many times until all possibilities for a program's location in memory are covered. Sotirov also mentioned the possibility of corrupting a program's non-control data as a way to manipulate its internal logic. This attack would require a more detailed understanding of the program's semantics, though. Finally, he proposed as a research direction the use of program analysis techniques to better understand and control a process's memory layout, as a way to oppose ASLR.

Sotirov concluded by arguing in favor of current mitigation techniques, as they significantly raise the bar against exploitation. The question of whether they are enough is hard to answer, but it is likely that the arms race will continue for the time being. And, as he pointed out, "we will always have the Web and all of its brokenness to look at."

Rik Farrow wondered why format string vulnerabilities had disappeared so quickly. Sotirov replied that it was because they were so easy to find. Peter Kristic asked whether using virtual machines (like VMware) makes any difference with regard to exploitation techniques. It makes no difference, since full virtualization replicates the execution environment of a real machine. Ben Zorn asked whether Sotirov had thought about any new mitigation techniques which might help to defend against some of the new attacks, to which he comically replied, "Certainly, but I will not tell you what they are." Sotirov also mentioned (citing the example of Microsoft) that as a result of this arms race, programmers' awareness about writing secure code had increased, but he cautioned the audience never to underestimate the potential of developers to introduce new vulnerabilities into code. And in the unlikely circumstance that all memory corruption vulnerabilities are found and fixed, the Web will always be there as a fertile ground for future exploitation, with whole new classes of vulnerabilities.

JAVASCRIPT SECURITY

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

- ***Gatekeeper: Mostly Static Enforcement of Security and Reliability Policies for JavaScript Code***
  Salvatore Guarnieri, University of Washington; Benjamin Livshits, Microsoft Research

Ben Livshits explained that Gatekeeper statically analyzes JavaScript code to check for violations of security and reliability policies. Statically analyzing JavaScript is difficult because it offers many ways to accomplish any given task. For example, to materialize an alert box one can call simply call alert(), one can use document.write() to write a call to alert(), one can create an alias of document.write() and call it, one can use eval() to write a call to document.write() that writes a call to alert(), and so on. Gatekeeper allows administrators to set simple policies that disallow certain JavaScript features. It uses a whole-program static analysis

approach that is, according to the authors, general enough to be used for purposes other than policy enforcement.

Gatekeeper recognizes two subsets of JavaScript: JavaScript_{GK}, which lacks several JavaScript features including eval(), and JavaScript_{SAFE}, which further lacks several more features. The subsets are such that the SAFE variant is fully statically analyzable without runtime checks, while the GK variant requires basic instrumentation at runtime to aid policy enforcement. Livshits described the authors' experiments on over 8,500 JavaScript widgets from three major Web sites owned by Microsoft and Google, noting that the majority of those widgets were already in the SAFE subset or GK subsets without any need for modifications. Given a program in one of the JavaScript subsets, Gatekeeper uses points-to analysis to track object relationships, thereby ensuring that object aliases do not confound the policy checker. Livshits said that Gatekeeper's points-to analysis is sound, meaning that its policy checker finds all violations it knows to look for. To illustrate the syntax of policy declarations, he showed an example of a Datalog rule that recognizes calls to document.write(). Finally, he offered experimental results: with nine security policies and two reliability policies in hand, Gatekeeper found 1,341 policy violations across 684 of the 8,500 widgets; it also found 113 false positives spread across only two of the widgets.

Adam Barth asked whether Gatekeeper had to parse HTML in order to catch violations; Livshits responded that disallowing document.write() was sufficient to make parsing HTML unnecessary, and that they did not test their system without disallowing it. The session chair, Lucas Ballard, expressed appreciation for the authors' choice of a small, tractable data set (viz., widgets), and asked whether they had attempted to apply their techniques to more complex content. Livshits remarked that analysis of such content was one of the authors' long-term goals, but that most large applications use some of the constructs Gatekeeper flags as suspicious. Livshits suggested that by-hand annotation of legitimate uses of such constructs would be a reasonable way to allow them without confusing Gatekeeper. Ballard proceeded to ask how the authors' work relates to JavaScript strict mode, to which Livshits replied that there were various connections. He remarked that current approaches required subsetting JavaScript and expressed hope that, for the sake of simplicity, some of the current approaches would be implemented directly in the browser.

- ***Cross-Origin JavaScript Capability Leaks: Detection, Exploitation, and Defense***
  Adam Barth, Joel Weinberger, and Dawn Song, University of California, Berkeley

Joel Weinberger spoke about using JavaScript heap-graph analysis to find a previously unnoticed class of browser vulnerabilities. The JavaScript security model includes a notion of contexts, which are separate containers for separate collections of objects. Such separation is designed to prevent private information from leaking between pages or page elements, specifically those with different origins; for example, an advertisement from an ad network should not be able to steal cookies from the page that embeds it. The policy of separating objects from different origins is commonly referred to as the same-origin policy. Weinberger claimed that the authors' work uncovered a new class of vulnerabilities in browsers' enforcement of the same-origin policy.

Weinberger pointed out that browsers implement two concurrent—and different—security models when it comes to JavaScript. Although the Document Object Model (DOM) that exists in each JavaScript context has a reference monitor and a concomitant same-origin policy enforcement mechanism, the JavaScript engine is a separate entity that uses a separate capability-based policy: if you hold a reference to an object, you are granted access to that object. The authors call the circumvention of the DOM's policy in favor of the permissive one a cross-origin JavaScript capability leak: if context B somehow obtains a reference to an object in context A (e.g., if context A passes a reference to context B, or if such a reference leaks), then context B is allowed to access the object without obtaining permission from context A's reference monitor. To detect these capability leaks, the authors instrumented WebKit's JavaScript engine with calls into an analysis library at object creation, destruction, and reference. As the program executes, the library fills out a heap graph.

Weinberger showed several heap graphs of increasing complexity, then described their automated heap graph analysis as a tree traversal which flags edges that span multiple contexts. Running their heap graph analysis on the security-related tests from the WebKit test suite revealed two new vulnerabilities, of one of which Weinberger showed a graphical example. The same technique found several major flaws in the open-source CrossSafe cross-domain JSON request library. Finally, Weinberger suggested access control checks on every object property access as an in-browser defense mechanism, and he remarked that the results of the added checks could be cached using a mechanism that already exists in modern browsers.

Ben Zorn pointed out that JavaScript benchmarks have tight loops that result in access control checks being handled primarily from cache, and he asked whether the authors have tested the overhead of their proposed defense mechanism on code other than test suites. Weinberger responded that the authors have tested other code informally and found their mechanism's performance to be qualitatively good. Lucas Ballard asked Weinberger whether any Web page could exploit the WebKit vulnerabilities to gain access to any other Web page, and Weinberger remarked that before WebKit was patched such exploitation had been possible.

- ***Memory Safety for Low-Level Software/Hardware Interactions***
  John Criswell, University of Illinois; Nicolas Geoffray, Université Pierre et Marie Curie, INRIA/Regal; Vikram Adve, University of Illinois

Nicolas Geoffray spoke about SVA-OS, a system that identifies memory safety violations in low-level software-hardware interactions with the goal of defanging kernel bugs. He defined software-hardware interactions as sequences of instructions that manipulate hardware resources. Such interactions, even when expressed in perfectly valid code free of type errors, can circumvent the execution environment's memory safety guarantees or corrupt the hardware resources they manipulate. Geoffray cited processor state, I/O objects, and MMU mappings as examples of manipulable hardware properties whose misuse can result in security violations. As a set of enhancements to the SVA compiler-based virtual machine, SVA-OS comprises a Linux 2.4 instance plus low-overhead compiler analysis and runtime checking of hardware accesses.

Geoffray presented details about how SVA-OS intervenes in several hardware interactions; the interventions are implemented either as special instructions in the SVA VM or as runtime checks. To prevent a task's processor state (e.g., the program counter) from being manipulated before it is properly restored by a context switch, SVA-OS adds an instruction that, instead of temporarily storing a task's processor state in memory where it can be manipulated at rest, atomically swaps one task's processor state for another's. To ensure that memory-mapped I/O operations behave properly, SVA-OS adds I/O-specific load and store instructions whose operation parallels that of regular memory loads and stores; it then segregates memory operations into those that do and those that do not affect I/O. Further, SVA-OS adds runtime checks on MMU updates to ensure that kernel memory is not mapped into user space and that physical memory is not remapped to incorrectly typed virtual pages. Geoffray reported that SVA-OS caught bugs that SVA did not: they tested two real-world MMU exploits, which SVA-OS disallowed; they injected errors into their Linux kernel and observed that SVA-OS prevented crashes; and they discovered that SVA-OS would have caught a serious bug in an Ethernet driver for Linux 2.6 that disabled many network cards. Geoffray showed several performance graphs demonstrating that SVA-OS imposes negligible overhead compared to SVA.

An audience member asked how SVA-OS preserves type safety during MMU remapping and wondered whether SVA-OS maintained type information for physical memory. Geoffray responded that SVA (rather than SVA-OS) did so by segregating physical memory by type.

INVITED TALK

- ***How the Pursuit of Truth Led Me to Selling Viagra***
  Vern Paxson, EECS, University of California, Berkeley, and Senior Scientist, International Computer Science Institute

Summarized by Todd Deshane (deshantm@clarkson.edu)

Vern Paxson, a self-proclaimed empiricist, admittedly loves data. The reason he loves data so much is because he has a thirst for the truth and also a phobia about being fooled. In this invited talk, Dr. Paxson described over two decades of Internet measurements and how the changes have been both incredible, at times surprising, and often unpredictable. He started with a general description of network characteristics and then talked about some early manual attacks, followed by the emergence of worms, botnets, and spam. He explained how this led him to begin a campaign to pretend to sell Viagra.

There are three invariants throughout his study of Internet data: growth, explosive onset, and diversity. Between the time when Vern applied to graduate school in 1988 to the publication of his paper "Growth Trends in Wide Area TCP" in 1994, the Internet grew from about 56,000 Internet hosts to about 3 million. The growth was attributed to the explosive commercial use of the Internet, exemplified by WWW traffic doubling every eight weeks from late 1992 to 1994.

Dr. Paxson's first demonstration of explosive onset appears in his quest to understand some seemingly anomalous data that he received regarding USENET bulletin board traffic. His data from 1986 to 1994 shows exponential growth of USENET usage (80% growth per year). Plotting this data on a log linear graph shows a perfect fit to the line. The only problem was that the data ends in 1994, but Vern really wanted to follow up on the data. He conjectured that it couldn't keep growing exponentially; generally, data like that breaks downward (fades gradually before coming to an end), but it turned out that two new data points showed the contrary. After some investigation, he determined that between 1994 and 1996, abuse, in the form of piracy and porn, arrived on USENET and the Internet as a whole. That abuse broke a decade-old invariant (the consistent exponential data growth) upward and not down as would have been expected.

In the mid-1990s, Internet abuse started becoming a major concern. The operators Paxson was in contact with at Lawrence Berkeley National Laboratory (LBL) wanted to know if he could use the data he was collecting to give some insight into the intrusions. Not only did he think that it was possible, but he thought it could be done in real time. This led him to create the Bro Intrusion Detection System (an open source, UNIX-based project that is still actively worked on by Vern and others), which was running 24 hours a day and 7 days a week starting in 1996. Much of the data presented in the rest of the talk was gathered by Bro.

The ability to use Bro at LBL and the ties with LBL operational deployment were "research gold." In particular, from host-scanning data Paxson was able to describe in detail the traffic changes starting with the emergence of the Code Red worm and the beginning of the worm era in 2001. Worms such as Code Red, Nimba, and Blaster were just the beginning, however. Again, using the scanning data, he was able to describe the emergence (around 2002) of what he refers to as auto-rooter tools, more commonly known as bots. At this time there was another significant increase of traffic, which he attributes to malice. Another interesting phenomenon he described was the diversity of the attacks, both in terms of the services attacked and the patterns of when the attacks occurred. For instance, ports scanned included common well-known ports as well as more obscure ports (such as the Sasser backdoor). The patterns of when scanning occurred ranged from heavy traffic during the day, to consistent scanning traffic regardless of time, to scanning the entire Internet at a certain time of day, every day.

In the second part of the talk, he described how he led an effort to infiltrate the Storm botnet and run a spam campaign. The inspiration for the spam campaign was the fact that he had studied the enemy and understood that profit was the motive of the botnet masters. The shift from curiosity and fame to an underground botnet-based economy had begun. He showed screenshots of professional spam software, sites that auction stolen eBay accounts, sites that sell social networking bots (with separate services that would integrate CAPTCHA bypassers), and affiliate programs that allow people to refer others and get a cut of the profits on these malicious tools. He realized that a large part of the business model was based on turning exploits into bots, then turning the bots into spam worker threads, then converting user clicks into sales. The spam campaign is described in further detail in "Spamcraft: An Inside Look at Spam Campaign Orchestration" presented at LEET '09. He continued by highlighting the fact that spam-filtering software and blacklisting spam bots filtered much of the spam to junk mail folders, which meant that only a small percentage of the spam was actually seen by users in their inboxes. During what he calls their spam conversion experiment (counting fake sales of Viagra) they were able to instrument 1.5% of the Storm botnet workers. They estimate that if they had been able to instrument the entire botnet army, they would have been able to make around $3 million. He notes that there was a lot of FUD (Fear, Uncertainty, and Doubt) about the Storm botnet in the news, where the media made claims of very large profits from Storm (orders of magnitude larger than reality) that are erroneous due to flaws in measurement methodology.

Paxson concluded with some reflections on the enormous changes he has seen in the Internet in just a couple of decades, especially in cybercrime (for profit) and the latent threat of cyberwarfare. He emphasized that measuring is easy, but measuring in a meaningful and sound way is hard (full of unfun grunt work dealing with messiness and error). Despite the challenges, he argues, it is the only way to get the truth and you can even run into some very interesting surprises (including diversity, exponential growth, unexpected threats, and rapid changes in the landscape). He encouraged the students in the audience to take on the challenge, as there is a deep fundamental need for well-grounded empirical data in the computer security field.

An audience member asked whether he thought that more success might come with more waves (repeat customers) of the spam campaign, to which he agreed that it was possibility, but he also noted that there is a tension over whether the botnet would be able to go after these follow-up sales or if the pharmaceuticals themselves would follow up. Steve Bellovin wondered about the ethics and IRB process involved. Paxson responded that he had lawyers look at the experiment, but that it didn't go through the IRB process, although he admits that it should have. A second follow-up study is currently going through a long IRB process, he noted. He also mentioned that there is an upcoming workshop that focuses on ethics at Financial Cryptography and Data Security '10. An admirer of the Spamalytics paper asserted that spam makes a lot of money, to which Paxson responded that he would have thought it would have been more (not only around $2 million per year as according to his data). He recommended that people think about the problem of network saturation (a "tragedy of the commons" scenario). Were the phishing attacks from the same players? He didn't know and speculated about the structure of the attackers, whether there were one or a few kingpins, or if there were, instead, a lot of ankle biters.

RADIO

Summarized by Italo Dacosta (idacosta@gatech.edu)

- ***Physical-layer Identification of RFID Devices***
  Boris Danev, ETH Zürich, Switzerland; Thomas S. Heydt-Benjamin, IBM Zürich Research Laboratory, Switzerland; Srdjan Căpkun, ETH Zürich, Switzerland

RFID chips are important components in the security of systems such as electronic passports (ePassports) and identity cards. Three security mechanisms have been defined to protect ePassport RFID chips, but only one is required by the standards. On the other hand, multiple attacks against these mechanisms have been published by security researchers. These attacks against ePassports prompted the authors to determine whether RFID chips can be uniquely identified based on their physical-layer features and the accuracy of the identification techniques. As Boris Danev noted, this work attempts to achieve a form of hardware biometrics. A direct application of this technique will be the prevention of cloning attacks against ePassports.

Danev described the experimental setup and the different experiments used to collect features from the RFID chips. A total of 10 ePassports and 50 Java cards were analyzed. The authors used three techniques to analyze the data collected: time, modulation shape, and spectral features analysis.

From these techniques, the analysis of modulation shape and spectral features were found to be the most effective (spectral features in particular). Based on the analysis, the authors were able to identify the ePassports' country and year of issuance, and some model and manufacturer chips design. The techniques evaluated also showed good accuracy: a 95% successful identification rate (5% equal error rate). In addition, Danev said that the accuracy can be improved dramatically through the combination of burst and sweep techniques. Finally, the authors have done some preliminary work to determine how hard it is to reproduce the fingerprints to defeat the identification techniques proposed.

Danev was asked if environmental conditions such as temperature and age could affect the fingerprinting of RFID chips. Current work is trying to determine if aging can affect the proposed identification techniques, and more work is needed to analyze the impact of other environmental factors. Several members of the audience were concerned about the privacy risks of fingerprinting RFID chips, such as the remote profiling of individuals. Danev mentioned that such privacy attacks may be possible but not by using the features analyzed in this work, because such features cannot be measured reliably from a distance. Another question related to the use of physical protection mechanisms (i.e., metal shields) in current ePassports. The author mentioned that some techniques are being implemented, but they vary from country to country. Finally, in response to a question regarding the relationship between the quality and the variability of an CRFID tag, Danev mentioned that the cheaper the design the more variability an RFID chip will have and, therefore, the easier it will be to identify.

- ***CCCP: Secure Remote Storage for Computational RFIDs***
  Mastooreh Salajegheh, Shane Clark, Benjamin Ransford, and Kevin Fu, University of Massachusetts Amherst; Ari Juels, RSA Laboratories, The Security Division of EMC

Computational RFID (CRFID) tags introduce a lot of interesting possibilities due to their additional components: a micro-controller, flash memory, and one or more sensors. However, these devices are also affected by hardware constraints: small memory size, tiny energy reservoir, and reboots every few seconds. Using the fact that radio transmissions are cheaper than writing to flash memory, the authors proposed outsourcing storage to a reader to save energy. However, using remote storage presents several security challenges: the data is transmitted over the air and the reader may not be trusted. Shane Clark introduced a new protocol, Cryptographic Computational Continuation Passing (CCCP), that adds the minimum security guarantees to allow CRFID tags to use a reader as a remote storage mechanism in an energy-efficient and secure way. The main goals of the protocols are to use remote storage to get real computational progress out of these devices and to eliminate Sisyphean tasks that result from the short power cycles of the CRFID tags.

The authors outlined a set of security goals: confidentiality, integrity, authentication, and data freshness. Based on these security goals and the CRFID tag constraints, the authors defined some basic and efficient security primitives: the use of stream ciphers (XOR operations) for confidentiality, and universal-hash-function-based MAC (UMAC) for integrity and authentication. To support the use of stream-keys, the authors introduced pre-computation when the tag is idle (good power season) to create stream-key bits. For data freshness, the authors used a unary encoding technique (hole punching) which allowed a counter in memory to be updated more efficiently.

Clark described the experimental testbed used to evaluate CCCP and the methodology followed. Energy was chosen as the most appropriate metric during the evaluation. The main result of the evaluation was that using radio for secure remote storage is cheaper than using local storage up to a data threshold size of 96 bytes. Finally, Clark suggested some future work in this area: the development of more efficient CRFID tags, extensions for long-term storage, work on WOM codes, and a public key system for CRFID tags.

Clark was asked about his expectations regarding flash memory costs in the future. Clark commented that backscatter transmissions used only one transistor, while flash memory operations used several, and that he was quite confident that in the near future flash will not be cheaper than radio. What about atomicity issues with data transmission in the CCCP protocol? A solution to this issue was to increase the counter in two steps: one before and one after data is sent.

- ***Jamming-resistant Broadcast Communication without Shared Keys***
  Christina Pöpper, Mario Strasser, and Srdjan Čapkun, ETH Zurich, Switzerland

It is a well-known fact that RF communications are vulnerable to jamming attacks. Traditional defenses against this type of attack are the use of spread spectrum (SS) techniques such as frequency-hopping SS or direct sequence SS (DSSS). These techniques rely on the use of a shared code to spread the transmitted messages. However, these techniques do not work well on broadcast communication scenarios where a sender wants to broadcast one or more authenticated messages to a potentially large number of receivers and some of the receivers may be unknown or untrusted (e.g., emergency and navigation systems). This paper presents a novel technique, Uncoordinated-DSSS (UDSSS), to solve this problem. UDSSS uses DSSS communication but releases the requirement of a shared secret key by using randomization and the following key observation: "Whatever has arrived unjammed at the receiver can be decoded." To an attacker, UDSSS looks similar to DSSS. The difference is that the code sequence used to spread the messages is chosen randomly from a set of public code sequences that both the sender and the receiver know. The receiver records the spread messages and tries to de-spread them using

the public code sequences in a trial-and-error fashion. For successful de-spreading of the messages, it is important to choose the same public code sequence used by the sender, as well as the right synchronization.

Pöpper described the prototype implementation of UDSSS, based on Universal Software Radio Peripherals (USRP) and GnuRadio, as well as the experimental setup and methodology used. Several adversaries were considered during the analysis, using the jamming probability with respect to a given message transmission. Also, message transmission time was used as the main metric during the evaluation. The results show that increasing the processing gain is much more harmful for the message throughput than increasing the size of the public code set, and that message throughput increases with the use of large message sizes. While UDSSS has lower performance than DSSS, it can be enhanced to achieve similar performance to DSSS in the absence of jamming; through the use of two parallel transmissions, one using a single code sequence and the other using normal UDSSS. Pöpper also suggested an optimization that is not described in the paper: using UDSSS to transmit only the spreading code and not the message, which allows faster decoding times and larger message sizes. Finally, Pöpper described a practical application of UDSSS in a navigation broadcast system.

INVITED TALK

Summarized by Salvatore Guarnieri (sammyg@cs.washington.edu)

- ***Designing Trustworthy User Agents for a Hostile Web***
  Eric Lawrence, Senior Program Manager, Internet Explorer Security Team, Microsoft

Eric Lawrence learned a lot while working on Internet Explorer 8 (IE8), and he talked about how IE was designed, where the current threats are, and the future of Web security.

Internet attacks are always evolving, so mechanisms to prevent or limit the effectiveness of these attacks must evolve as well. In Internet Explorer 7 (IE7), the goal was to reduce the attack surface on the local machine. This meant that IE7 had fewer vulnerable areas than previous versions of the browser. Now the local machine isn't as valuable a target. Much data lives in the cloud, so cross-site scripting (XSS), cross-site request forgery (CSRF), and other similar attacks are major problems. IE8 tries to address these new types of attack that don't necessarily target the local machine but, rather, the way in which confidential or high-integrity data is handled in the browser.

Security is difficult for the Web because the space is very complex. The browser needs to be secure and Web developers need to produce secure Web sites. This is a problem, since some Web developers don't even understand what same-origin policy is. Furthermore, security for the Web was largely an afterthought, and many of the interesting security models for the Web don't fit how the Web is being used today. Finally, since Internet Explorer (IE) has been around for a while, users expect that things that worked in an old version will work in a new version. This means that changes to the browser cannot break backward compatibility. It turns out that if backward compatibility is broken, developers don't update their sites to work in the new browser, so users simply refrain from upgrading and are left with a less secure browser visiting possibly insecure pages.

The security team for IE is focused on three areas: (1) security feature improvements, (2) secure features, and (3) security and compatibility. Security feature improvements are new features (e.g., the XSS filter) that exist solely to improve security. "Secure features" refers to the process of ensuring that new features (e.g., IE8's accelerators) are secure and do not increase attack surface. Security and compatibility focuses on ensuring that the security features are compatible with Web sites. Users need the security feature to work on the Web sites they visit or they will roll back to a less secure browser that does work.

IE8 has some features that make attacks much less effective. Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR) are turned on by default. This is a huge help, because they greatly limit the exploitability of memory-related vulnerabilities in the browser. Some new features improve security by reducing the amount of native code needed in the system. Plug-ins such as browser toolbars provide some functionality to users but are usually written in conventional languages like C and often contain security bugs. IE8 has accelerators and Web slices, which allow third parties to enhance the user experience without introducing potentially buggy native code.

No matter how secure the browser is, there is still a user involved in the system. Users are faced with social engineering attacks, such as phishing and malware-distribution sites, that trick them into granting the attacker permission to perform unwanted actions. People have tried a lot of things to warn users about potentially unsafe actions. IE8's SmartScreen Filter feature can provide a bold, unambiguous warning to block known-unsafe actions because Microsoft actively searches out dangerous content on the Internet and flags it using the URL Reputation Service.

SECURING WEB APPS

Summarized by Shane Clark (ssclark@cs.umass.edu)

- ***xBook: Redesigning Privacy Control in Social Networking Platforms***
  Kapil Singh, Georgia Institute of Technology; Sumeer Bhola, Google; Wenke Lee, Georgia Institute of Technology

Kapil Singh gave a talk about the problems with privacy control in social networking platforms and xBook, a system designed to allow fine-grained and reliable control over applications for such platforms. Singh noted that social networking sites such as Facebook, Twitter, and Orkut are still

growing rapidly in popularity. Many sites are also evolving into platforms that expose APIs to third-party developers, giving any developer's application access to almost all stored user data. A user must generally agree to this access when adding an application, but there is no guarantee that the application will use exposed data appropriately or that it will only access data that is necessary for its advertised functionality. The goal of xBook is to provide privacy protection to users without requiring changes to the browser or any discernible usability changes from the user's standpoint.

Singh next introduced the high-level approach taken with xBook. All applications are run from within the trusted xBook domain to allow mediation. Applications are then monitored at runtime in the browser. While they are still allowed access to any user data, applications are required to make the use of such data explicit to the user; xBook is able to enforce this policy by tracking information flow. xBook also requires applications to be split into client and server components that are confined appropriately. For example, JavaScript must be written in a safe subset of the language called ADsafe and not have access to a page's DOM elements. To enable necessary DOM accesses, xBook implements a DOM wrapper that gives client-side components access to only those elements that belong to the same application. Even when the components of a single application are communicating among themselves, the data that they are able to exchange is subject to the explicit restrictions specified by the application developer and agreed to by the user. Application components are also subject to these restrictions when communicating with external entities. The authors implemented xBook as a Facebook application to demonstrate its effectiveness, and they also implemented two applications that ran on top of xBook.

An audience member asked if the authors have performed any experiments to test usability. Singh answered that they have not completed any experiments but have designed the system to be easy to use. Another audience member wanted to know why users should be more willing to trust xBook with their data than any given application. He was also concerned that application developers may not be willing to target xBook because they want direct access to user data without added restrictions. Singh responded that users already trust a large number of other applications, all with the same privileges. Trusting xBook only requires users to add a single application, minimizing the number that must be trusted overall. If users choose to use xBook, developers will be forced to target xBook with their development in order to achieve widespread use. Finally, an audience member pointed out that developing applications for xBook is not necessarily an indicator of ease of use. He asked whether the authors had attempted to port an existing application and, if so, what their experience was. Singh countered that the two applications they implemented mirror the functionality of existing apps, but he acknowledged that there may be a learning curve for their cross-platform APIs.

- ***Nemesis: Preventing Authentication & Access Control Vulnerabilities in Web Applications***
  Michael Dalton and Christos Kozyrakis, Stanford University; Nickolai Zeldovich, CSAIL, MIT

Michael Dalton presented his work on Nemesis, a system designed to automatically prevent authentication and access control vulnerabilities in Web applications. Dalton first introduced some of the failings of Web authentication with an illustrative example. If a user Bob uploads a photo to a Web application, for example, that photo is typically stored in the database using the credentials of the Web server user. The database has no knowledge of Bob, and the Web application's user account necessarily has privileges equal to or greater than those of any user that exists in the application. According to Dalton, this semantic gap fundamentally breaks Web authentication by requiring the application programmer to accurately insert access control and authentication checks before every file-system or database operation. This must be done perfectly in order to adequately secure any Web application.

Dalton next introduced the two classes of attacks that Nemesis addresses and the approach to preventing each of them. Authorization bypass vulnerabilities occur when a user is able to access a resource without authorization, often as the result of a missing or incorrect authorization check. Authentication bypass vulnerabilities occur when successful authentication occurs without valid credentials, often caused by a poor URL/cookie validation method or simply weak cryptography. Nemesis stops both classes of vulnerabilities by inferring when authentication is performed correctly using dynamic information flow tracking (DIFT), also known as taint tracking, and automatically enforcing access control lists (ACLs) for Web application user accounts (as opposed to Web server processes). To support the use of Nemesis, DIFT functionality must be added to the language interpreter in use, and ACL enforcement must be added to the language's core library. Nemesis does not require the modification of the application code.

A Nemesis prototype has been implemented for the PHP language and Dalton presented some examples of vulnerabilities in real-world applications that Nemesis is able to prevent. The prototype stopped authentication and authorization bypass vulnerabilities in six popular PHP applications without any discernible performance overhead, but does suffer from some limitations. There is currently no SQL query rewriting support, forcing the authors to manually insert code to perform ACL and authentication checks in some cases. The current prototype also lacks automatic ACL generation, which leaves application administrators to write their own. Both SQL query rewriting and automatic ACL generation based on logs are slated for future work.

Bryan Parmo asked if Nemesis could be easily extended to work with sites that use authentication methods other than passwords. Dalton answered that Nemesis currently relies on a byte-by-byte comparison of two strings in order to infer successful authentication, but as long as there is some

authentication mechanism that Nemesis can be made aware of, it should be possible to support that method. Parmo followed up by asking if Nemesis is effectively removing the responsibility for authentication from the Web application and handling it completely. This is possible but Nemesis actually requires some amount of ground truth in the application that it can trust as a valid authentication technique. Finally, David Wagner asked if Dalton believed that Nemesis is applicable to other popular Web application languages and if he foresees any particular challenges in supporting these other languages. Dalton answered that many popular languages have had DIFT support implemented for them in the past and it should be straightforward to add Nemesis support. The authors chose to use PHP because they found the interpreter easy to modify, and there are many insecure PHP applications currently in use.

- ***Static Enforcement of Web Application Integrity Through Strong Typing***
  William Robertson and Giovanni Vigna, University of California, Santa Barbara

William Robertson noted that Web application vulnerabilities make up more than half of all reported vulnerabilities over the past two years, according to Symantec. The majority of these Web vulnerabilities are either cross-site scripting (XSS) or SQL injection vulnerabilities. He acknowledged that there are a number of existing solutions such as application firewalls, automated code analysis, and penetration testing, but argued that these are clearly insufficient measures, considering the continued prevalence of XSS and SQL injection vulnerabilities.

Robertson said that a major source of Web application vulnerabilities is the treatment of Web documents and database queries as unstructured character sequences. Because there is no knowledge of appropriate structure and content at the language or framework level, developers are responsible for manually sanitizing this content. Developers will inevitably fail at this task, as correctness requires perfect code. In order to address this problem, the authors implemented a language-based solution intended to explicitly denote structure and content using a strong type system. This approach shifts responsibility for integrity enforcement from the developer to the language and removes the need for separate testing or analysis to ensure that an application is secure.

The authors' prototype solution took the form of a Haskell-based application framework, and they presented evaluation results. The framework enforces integrity by requiring the application to construct a tree of document nodes that it then "renders" into serialized raw text for display. Each node of the tree is sanitized during the rendering process in order to prevent XSS vulnerabilities. The framework is able to remove SQL injection vulnerabilities by simply requiring applications to exclusively use prepared statements where possible. When prepared statements are not possible, the framework enforces integrity using a node rendering approach similar to that for document nodes. Robertson claimed that their framework achieved full sanitization coverage and that they had confirmed correct sanitizer operation using a large number of randomly generated test cases. Finally, Robertson showed that their framework's performance was similar to that of Tomcat and Pylons. They found that the performance of their prototype fell between the other two frameworks in testing and is thus acceptable.

Adam Barth asked if the authors have investigated the possibility of removing XSS vulnerabilities stemming from client-side code such as JavaScript. Robertson answered that they are investigating this problem. One promising approach might be to compile a representation of the client-side code on the server side into JavaScript, similar to Google Web Toolkit's approach. Benjamin Livshits asked about using symbolic execution to verify sanitizers, as opposed to random testing. The authors have not looked into this, but it seems like a useful approach. Finally, David Wagner complimented Robertson on the system's removal of the ability to even express some server-side vulnerabilities and asked if the authors have looked into any other classes of vulnerabilities. They chose to focus on SQL injection and XSS because they can be simply expressed and prevented. The authors are considering other vulnerabilities, but they may be more difficult to address.

INVITED TALK

- ***Compression, Correction, Confidentiality, and Comprehension: A Modern Look at Commercial Telegraph Codes***
  Steven M. Bellovin, Professor of Computer Science, Columbia University

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

Steve Bellovin delved into historical archives to present an enlightening investigation of telegraph codes, looking back with a modern perspective and uncovering remarkable similarities to our current network structures and protocols. An example of these surprising findings was discovering that telegraph systems contained protocol stacks—the link layer was well defined—and error correcting codes. Some mechanisms were different, e.g., the job of a router was performed by an operator who would relay telegraphs to the correct destination link, but many of the problems were similar to those we currently face. Steve focused on four areas: compression for reducing the cost of transmission; detection and correction of coding errors; confidentiality to deal with operators seeing messages as they are transmitted; and comprehension of the culture of the time, which is strikingly conveyed in how the codes were structured.

The talk described a succession of codes from Sir Home Popham's naval code in 1805, which was expressive enough to allow conversation, through to the telegraph era, where codes were compiled not only for general use but for specific industries as well. As an example, theater codes differentiated between specific capabilities desired of chorus girls, each description compressed to a single code word. It was noted

that we still use domain-specific compression: Lempel-Ziv doesn't work as well on multimedia as JPEG or MP3. Early codes did not provide redundancy, and techniques such as terminal indices and mutilation tables were created. These would provide potential endings for words where the first two letters were known, and usually about five possibilities were available. Based on the sentence, the code could either be retransmitted or disambiguated contextually.

Confidentiality was an issue, identified in an 1870 document that described a threat model for communication security. However, it was not particularly well implemented in many codes. Techniques included mono-alphabetic substitution and constant additives. Even the US Government's code consisted of constant additives built on top of the Western Union code, and this was state of the art until around World War I. Bloomer's Commercial Cryptograph was a more successful code, a "holocryptic cipher" that was unreadable without a key. This was a user-created two-part code.

Some interesting details also appeared about threats to confidentiality, such as Britain requiring cable companies to turn over copies of all international telegrams, and copyright, where the US was not a signatory to the Berne Convention until 1976, causing publishers to often publish their works first in the US to ensure copyright there before publishing in the rest of the world. A particularly interesting reference to copyright infringement as "piracy" appeared in a code book from 1936.

Telegraph codes were only recently supplanted in many places, with China using them regularly until around the year 2000. Much of what can be found in modern codes is an evolution of what was done then, with formalized and mathematical models now created, but the basis was set for these techniques at least 75 years previously. A draft paper of these findings can be found on Steve's Web page at http://www.cs.columbia.edu/~smb/papers/codebooks.pdf.

In the Q&A session, Matt Blaze pointed out that some early military codes were based on poetry, and that codes are only as secure as the people compiling them. Steve added to this by discussing how early Russian OTP codes were not in fact random; their randomness depended on the people generating the pads.

APPLIED CRYPTO

Summarized by Andres Molina (amolinaf@nsm.umass.edu)

- ***Vanish: Increasing Data Privacy with Self-Destructing Data***
  Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy, University of Washington

  Awarded Outstanding Paper!

Roxana Geambasu made the case that currently, when data is communicated over the Internet, it lives forever. Moreover, while it is in transit, different providers create multiple copies of the data. The users do not know where these copies reside or for how long they will be kept. Some of this data may be sensitive and resurface in a manner that is unwanted by the user; for example, by means of a subpoena. The speaker referred to this type of attack as a retroactive attack on archived data. She also pointed out that current solutions such as PGP do not protect against such attacks, because the key to recover the archived material can be obtained by legal means. This problem also occurs in centralized services in which a provider maintains the data in an encrypted form. The speaker gave the example of Hushmail, an encrypted email provider that, in at least one case, allowed access to email content as required by a subpoena.

Vanish is a system that combines distributed hash tables (DHTs) together with secret sharing techniques to accomplish the goal of destroying data. The data gets encrypted using a key that is split using Shamir shared secrets, gets stored using the DHT in key-value pairs, and gets destroyed after the data is successfully encrypted. Properties of DHT systems such as member churn and discarding of key-value pairs after a timeout is reached allow the proper destruction of the shared key after an expiration date without requiring any action by the user. With Shamir shared secrets, a threshold amount of the shared key must be recovered, and once that threshold can no longer be obtained the key is gone forever. An attacker would need to capture the key before or while it is being distributed, but this only applies to premeditated attacks, not retroactive attacks.

The system has currently been realized using a Firefox plug-in as the front end, allowing users to interact with services such as Gmail, Hotmail, Facebook, Google Docs, or essentially any other Web application that deals with plain text. However, the speaker mentioned that Vanish can easily be implemented as a plug-in to Thunderbird or other email clients. The presentation only addressed some aspects of the evaluation relevant to security, referring the audience to other details of evaluation criteria in the paper.

A member of the audience asked about the selection of the parameters from the user's perspective. Geambasu responded that the current implementation provides the user with reasonable default parameters that can be modified to balance security and performance. Someone noted the reference in the talk to the need to use encryption techniques like those employed in PGP to protect the data in transit. Geambasu said such encryption techniques should be used in addition to Vanish, not instead of it. Someone pointed out that there are other existing solutions to the problem, such as the Tahoe distributed file system, although this system could pose a major drawback in terms of usability. Geambasu was not aware of Tahoe and agreed that usability was a major consideration in the design of Vanish. How does Vanish protect the anonymity of the recipient, since if the recipient is known, it would be possible to subpoena the recipient's computer in order to obtain the contents of the caches? The Vanish model assumes that the attacker would not know about the sender or recipient until after the data

has expired and enough time has elapsed for the data to be destroyed.

For more information and to obtain the application, see http://vanish.cs.washington.edu/.

- ***Efficient Data Structures for Tamper-Evident Logging***
  Scott A. Crosby and Dan S. Wallach, Rice University

Scott Crosby started his talk by noting that everyone has logs and that there are many applications that require storage using tamper-evident techniques. Some of the examples that were given included HIPAA regulations and credit card payment contracts. Scott also pointed out that current commercial solutions to this problem rely on the correct operation of the appliances and are too slow.

Crosby then proposed the use of efficient data structures for tamper-evident logging. This approach would allow a third party to prove the correctness of the solution using known cryptographic techniques. Additionally, this solution offers logarithmic performance in all operations, instead of the linear performance that previous solutions offer. The proposed solution based on history trees would offer performance of 1,750 events per second, including digital signatures, and 8,000 audits per second.

Crosby explained that audits are essential for this type of application. If a malicious logger could anticipate that a particular log entry was not going to be audited, then he could easily replace that entry. For that reason, it is necessary to ensure that every event has a non-zero probability of being audited. Furthermore, the author distinguished two types of audits: audits to verify correct insertion of events and audits to verify consistency among commitments. This new paradigm requires that both insertions and audits are cheap in terms of CPU, communication, complexity, and storage. The solution proposed makes this possible by using Merkle binary trees that avoid having to compute linear chains of hashes. The solution allows the probabilistic detection of tampering by only verifying a subset of events. Additionally, this solution allows the computation of Merkle aggregates and the performing of safe deletion by using tree pruning techniques.

The system was evaluated with a syslog implementation, logging 4 million events using 11 hosts over four days. The experiment was repeated 20 times, using DSA signatures and SHA1 hashes. While the results were reasonable (1,750 events/sec; 8,000 audits/sec), approximately 83% of the runtime was spent computing signatures. Also, compression reduced performance by about 50%. The authors suggested that concurrency and replication would improve performance. Additionally, given that the major bottleneck is due to the computation of signatures, a performance gain is expected if only a subset of the commits are signed, a faster public key encryption scheme is used (such as ECC), or the computation of the signatures is offloaded to other servers. Using the latter approach, it is possible to process up to 10,000 events/sec.

Someone asked if it would be possible to perform data reduction preserving the tamper-proof properties. Yes, this is possible by employing tree pruning techniques and appropriately using Merkle aggregation predicates with each event. Another member asked if the technique would account for trees with different depths, to which the author responded affirmatively.

- ***VPriv: Protecting Privacy in Location-Based Vehicular Services***
  Raluca Ada Popa and Hari Balakrishnan, Massachusetts Institute of Technology; Andrew J. Blumberg, Stanford University

Raluca Popa presented VPriv, a system that can be used in various transportation systems to compute functions over driver paths, such as usage costs. These computations are performed while preserving users' privacy. Current systems such as EZPass do not offer adequate levels of privacy, as it is possible to track the times and places where an EZPass has been used. The solution came from observing that it is possible to compute costs associated with a path without actually knowing all the details of the path, by using zero-knowledge proofs and secure multi-party computation. VPriv was designed from the start to be efficient by exploiting the properties of homomorphic encryption and families of random functions. The system relies on the ability to compute functions on tuples containing times, locations, and random tags unique to each tuple.

The security model considers two parties: a client, such as a driver, and a server. The client is not trusted, as he or she has incentives to alter the protocol in order to reduce personal costs. The server, on the other hand, is partially trusted. That is, the server will be trusted to perform the protocol correctly, but it will not be trusted to preserve the privacy of the client. The speaker mentioned that the paper also discusses a protocol dealing with a malicious server; in other words, a server that is not trusted even to perform the protocol correctly. While the talk focuses on the application of usage tolls, the speaker briefly explained how the system can easily be customized for other applications such as "pay-as-you-go" insurance and speeding violations. The protocol was divided into three phases. In the first phase, referred to as registration, random tags are created to which the client commits. The second phase involves the uploading of the tuples containing the random tags generated in the previous phase, together with the locations and times. Finally, the reconciliation phase is when the client would compute the owed cost and allow the server to verify the result without compromising the privacy of the client.

The evaluation of the protocol showed linear performance on the number of tags and the number of tuples uploaded to the server. The system is three orders of magnitude faster than Fairplay, a state-of-the-art general-purpose secure multi-party computation compiler. Raluca mentioned that it is possible to serve roughly a million customers with 21 server cores. The system was evaluated on Cartel, an MIT project, with 27 taxis and 4,826 one-day paths. The experi-

ments showed that the enforcement scheme is effective and efficient. The authors hope to see this system implemented in the car sharing company Zipcar.

In a system like this, will it be hard for a customer to dispute a bill when there is no actual record of the trips on the server side? This is handled by the reconciliation phase, in which the fees due are actually computed by the customer and only verified by the server. Is it possible to leak some information by allowing the server to identify where the tuples are uploaded from? The system may be complemented with the use of an anonymizing system, such as TOR, to upload the tuples. Ian Goldberg asked if more complex functions could be used to compute costs beyond simple additions. It is possible to compute conditional functions, and, in theory, any polynomial function could also be computed using similar techniques, but the details would have to be explored in future work. In the security model, although the server is trusted to perform the protocol correctly, it may also have incentives to prevent the protocol from being performed correctly. The paper includes an extension to the system that takes into account a malicious server. In such a case, efficiency would be affected.

INVITED TALK

- ***Top Ten Web Hacking Techniques of 2008: "What's possible, not probable"***
  Jeremiah Grossman, Founder and CTO, WhiteHat Security

  Summarized by Stephen McLaughlin (smclaugh@cse.psu.edu)

Jeremiah Grossman presented the top 10 Web hacking techniques of 2008. Tenth is Flash parameter injection (FPI). FPI uses embedded Flash media as a vector for accessing the HTML content of the containing page. One method for FPI requires that an attacker assign values to global variables which are initialized from values in HTTP requests.

Another technique that leverages a browser plug-in, ActiveX repurposing, comes in at number nine. In this attack, a malicious Web page takes advantage of the update functionality in the Juniper SSL-VPN client to first save a malicious configuration file to a known location, then place a path to it in the ActiveX control's INI file.

The eighth Web hack, tunneling TCP over HTTP over SQL injection, uses the reDuh server and client to give an attacker control of a remote machine behind a firewall. In this attack the reDuh server, which tunnels TCP traffic over valid HTTP commands, is uploaded to an SQL server behind a firewall that only allows HTTP traffic. The SQL server is used as a gateway between the attacker and the internal network. While Grossman pointed out that database hardening is one defense against this attack, the reDuh server may also be uploaded to an application server as a JSP page.

Web hack number seven can be used to determine if a victim is logged in to a given Web site or by observing the stylesheets from another Web page. A link to a target inline stylesheet from a malicious page can be used to obtain the cross-domain style definition. The malicious page can then check for style properties indicating the user is logged in to the target site. The sixth Web hack, abusing HTML 5 client-side storage, uses any cross-site scripting (XSS) vulnerability to either leak information from or inject code into HTML 5 client-side stored objects. Web hack five is a different Opera. The goal of this attack is to execute code in the Opera browser running in the opera:* context, which will give an adversary the ability to modify browser settings with opera:config. This is achieved by a cross-site request forgery to opera:historysearch, which already contains an XSS from a previously visited page.

Clickjacking, the fourth Web hack, uses the CSS opacity property and JavaScript to place an invisible button pulled from a form on a target page directly under the user's mouse. This causes the user to click the invisible button instead of some visible part of the page. If the user has already authenticated to the target site, this will result in the site taking some action on the attacker's behalf. Examples of target sites may be CPC advertisements, Digg links, and options on DSL router configuration pages. Grossman went on to say that this exploit may be used to trick users into enabling a laptop's camera and speakers through a Flash applet, allowing for remote surveillance through a Flash application. Grossman had a recommended countermeasure to the camera-enabling attack: placing tape or a sticker over the camera lens.

The third Web hack uses a feature of Safari in which any file of a type not recognized by the browser is saved to the desktop. This allows malicious site to effectively "carpet bomb" the target machine with garbage files or malware.

Coming in at second place is an attack on Google Gears which bypasses the cross-origin security policy. This is possible if an attacker can insert Google Gears commands into a file uploaded to a target site. These commands are likely to pass input filtering, as they lack suspicious tokens like <script> tags. A malicious Gears-enabled site then executes the commands in the context of the target site in the victim's browser.

The number one Web hack of 2008 is the GIFAR, a concatenation of a GIF image and a Java JAR archive. Because GIF files are parsed from the first byte and any garbage at the end is ignored, and JAR files are the exact opposite, appending a JAR file to the end of a GIF creates a GIFAR, which will pass input validation for file uploads while allowing the JAR file to be embedded in a page that will run in the targeted site's context.

Several of the questions after the talk concerned classifying Web hacks by type, technique, and trends. Grossman said that the trend in 2008 was toward attacks against the browser, while in 2009 we are likely to see a shift back toward servers with HTTP parameter pollution attacks. Also, according to Grossman, as diverse as 2008's Web exploits are, they can likely all be classified according to MITRE's Common Weakness Enumeration (CWE). He gave the ex-

ample that clickjacking could be classified as a UI redressing attack. Rik Farrow asked if Grossman used NoScript as a countermeasure, and Grossman said that he did. When Farrow then asked the same question of the audience, very few hands went up. (See p. 16 of this issue for Grossman's article on Web hacks and p. 21 for an article on NoScript.)

POSTER SESSION

Posters below summarized by Kalpana Gondi (kgondi@cs.uic.edu)

- ***An Examination of Secure Programming Practices Through Open Source Vulnerability Patch Characteristics***
  Mark Plemmons, Andrew Falivene, Jonathan Peterson, Adam Wenner, Will Quinley, Jing Xie, and Bill Chu, University of North Carolina at Charlotte

Mark Plemmons presented a study to understand secure programming practices by analyzing characteristics of patches for vulnerabilities in open source applications. Authors analyzed Linux kernel vulnerabilities and their patches for the period of 2006–2008 (from kernel.org) to learn characteristics such as number of files and lines changed. For the buffer overflow vulnerabilities, patches were localized to a small number of files. The methodology followed was software vulnerability cognitive complexity (SVCC), where SVCC = Lines of Code added + Lines of Code removed.

- ***The Impact of Structured Application Development Framework on Web Application Security***
  Heather Lipford, Will Stranathans, Daniel Oakley, Jing Xie, and Bei-Tseng Chu, University of North Carolina at Charlotte

Will Stranathans presented this work about the effects of structured application development frameworks on Web application security. The authors have studied practically, by training a few members to observe the behavior of the software they have written after taking the training in application development frameworks (especially struts). There were two groups of people, those with and without the knowledge of application development framework, who wrote the software. The software was tested against attacks like SQL injection, XSS, and system information leaks, and there was a major difference in defense against XSS and information leaks between the software developed with and without the application framework knowledge. The authors observed that the knowledge of frameworks will help in writing the secure code.

- ***Toward Enabling Secure Web 2.0 Content Sharing Beyond Walled Gardens***
  San-Tsai Sun and Konstantin Beznosov, University of British Columbia

San-Tsai Sun discussed enabling secure content sharing where there are no proper mechanisms in Web 2.0 to provide content sharing among individuals in a controlled manner across content-hosting or application service provider (CSP) boundaries. The design includes OpenID email protocol and RT policy service. OpenID email protocol enables OpenID identity providers to use email as an alternative identity. RT policy provides services for Internet users to organize their role-based trust-management access control policies and for CSPs to make access decisions. The two key concerns are usability and interoperability.

- ***A Self-Certified Signcryption Scheme for Mobile Communications***
  Ki-Eun Shin and Hyoung-Kee Choi, Sungkyunkwan University

Ki-Eun Shin pointed out that securing mobile communications is challenging because communicating entities (i.e., mobile devices) are resource-constrained, and mobile networks restrict Internet access. Hence, cryptosystems developed for mobile communications should be efficient, and overhead associated with the security protocol needs to be minimal. The intervention by the conventional trusted authority should also be minimized because of the restricted access to outside networks. The authors propose a self-certified signcryption scheme to withstand such obstacles in mobile communication.

- ***Developing Security and Privacy Requirements for a Local Area COllaborative Meeting Environment (LACOME)***
  Fahimeh Raja, Kirstie Hawkey, and Kellogg S. Booth, University of British Columbia

Fahimeh Raja explained that they tried to look beyond usability toward security and privacy. The authors have developed software to provide a multi-user platform to support sharing of co-located and collaborative work in the same platform. This is more useful for group discussions in the corporate world or any group, for that matter, discussing and sharing information on a particular topic. The key challenges here include: authentication, client authentication and authorization, and unintended disclosure of private data. They are focusing on performing one-on-one interviews and focus groups to get the end users' privacy and security requirements and to implement security and privacy controls, especially for those in front of a large audience. Finally, they want to test the scheme in real meetings.

Posters below summarized by Prithvi Bisht (bishtspp@yahoo.com)

- ***Comprehensive Redaction for Neurological Imaging***
  Alex Barclay, Laureate Institute for Brain Research; Nakeisha Schimke and John Hale, University of Tulsa

Alex Barclay presented a technique to preserve privacy of patients' data captured in neurological imaging. According to Barclay, researchers often share imaging data for collaboration and research purposes. However, it may compromise the privacy of patients, e.g., by reconstructing the face, time of capturing the image, etc. He presented a technique to preserve privacy by deleting certain data from images and presented a disk level algorithm to achieve this.

- ***Java Data Security Framework (JDSF) and Its Applications: API Design Refinement***
  Serguei A. Mokhov, Concordia University

Serguei A. Mokhov presented a framework to evaluate/compare various implementations of security algorithms in a homogeneous environment, primarily in Java. According to Mokhov, this framework provides interfaces to implementation providers. Further, according to him, such a framework enables evaluation of disparate implementations of an algorithm on dimensions and metrics set by the framework, such as runtime and memory usage.

- ***Towards Investigating User Account Control Practices in Windows Vista***
  Sara Motiee, Kirstie Hawkey, and Konstantin Beznosov, University of British Columbia

Sara Motiee presented a study plan to investigate patterns of usage for administrative or normal user accounts in Windows Vista. Specifically, the planned study aims to discover if users prefer admin/non-admin accounts, what challenges are faced in non-admin accounts, and how these patterns vary across disparate user groups.

- ***Large-scale Multitouch Interactive Network Visualization***
  Cody Pollet, George Louthan, and John Hale, The University of Tulsa

George Louthan presented an approach to graphically represent network traffic in large-scale networks. The goal is to allow a human to understand and reason about the real-time condition of the network. He further discussed plans to integrate a multi-touch screen in such a system to provide collaboration among different users.

- ***FloVis: Flow Visualization System***
  Diana Paterson, Joel Glanfield, Chris Smith, Teryl Taylor, Stephen Brooks, and John McHugh, Dalhousie University; Carrie Gates, CA Labs

Diana Paterson presented an approach to render the network traffic through visual artifacts. A real-time traffic visualization may allow security administrators to more quickly discover patterns of hostile activity and diagnose compromised hosts. It may also be useful in maintenance: e.g., heavy traffic on a few nodes may represent an in-progress denial-of-service attack as well as a heavily loaded subnet.

Posters below summarized by Asia Slowinska (asia.slowinska@gmail.com)

- ***Beatrix: A Malicious Code Analysis Framework***
  Christian Wressnegger

Beatrix is a framework designed to reduce the effort required to build a prototype for a malware detection technique. The framework introduces a plug-in infrastructure that divides the entire analysis process into six disjoint subtasks: e.g., input, extracting, or formatting. This architecture enables utilizing existing components, which lets the system conduct significant parts of the examination of malicious binaries and thus reduces programmers' efforts. Future work includes extending the set of modules available.

- ***SAND: An Architecture for Signature-Based Automatic Network Protocol Detection***
  George Louthan and John Hale, The University of Tulsa

George Louthan targeted the problem of network traffic classification, and proposed a content-aware method based on the actual contents of packets. His solution overcomes the prevalent lack of adherence to standard port numbers, which could result in incomplete traffic identification in port-based network monitors. The general SAND strategy for identifying a stream involves matching a set of string signatures describing a known protocol format. For example, the SSH identifier finds the strings "SSH-" and "CR LF" and takes the string between them to be the protocol version. Future work includes a thorough analysis of the performance and effectiveness of the system.

Posters below summarized by Patrick Wilbur (patrick.wilbur@gmail.com)

- ***Securing the Application Acquisition Chain: Security Concerns & Human Factors of Application and System Acquisition in the Enterprise***
  Eric Goldman, Rochester Institute of Technology

Eric Goldman explained that this work examines to what extent, if any, security is considered in an organization's selection of information technology software and hardware. They focused on small- to medium-sized organizations, which generally lack the resources, time, and experience to adequately address security. He considered both the processes of acquisition in information technology and the psychology of good decision-making, and he concluded that all businesses and individuals, regardless of size, deal with sensitive and valuable data to varying degrees, and that those organizations and individuals that do not keep up with security concerns become easy prey as others advance their security focus, despite those organizations' seemingly small size and value.

- ***Exploring the Human-Behavior Driven Detection Approach in Identifying Outbound Malware Traffic***
  Huijun Xiong, Chih-Cheng Chang, Prateek Malhotra, and Danfeng (Daphne) Yao, Rutgers University

Chih-Cheng Chang noted that outbound malware network traffic is unintended by the user and does not always correlate with a user's actions on the system or the user's intended actions. In this work, both user inputs and outbound traffic are semantically examined to see if a valid correlation exists between actions and outbound traffic. This work assesses the actions a user performs and exposes invalid outbound traffic, which could signify malware traffic that is undesired by the user.

- ***Towards Improving Identity Management Systems Through Heuristic Evaluation***
  Pooya Jaferian, David Botta, Kirstie Hawkey, and Konstantin Beznosov, University of British Columbia

Pooya Jaferian pointed out that good identity management is absolutely critical in an organization's access control framework. However, this work finds that identity management systems can host significant usability problems. The goal of this work is to expose usability problems in identity management that could result in mistakes being made while using or administering those systems, which, in turn, could result in the improper granting of access to resources.

- ***A Spotlight on Security and Privacy for Future Household Robots: Attacks, Lessons, and Framework***
  Tamara Denning, Cynthia Matuszek, Karl Koscher, and Tadayoshi Kohno, University of Washington

In the future, robots and automation will become increasingly ubiquitous in the household. Tamara Denning and Karl Koscher presented this examination of various household robots for security vulnerabilities, which include hard-coded passwords, lack of strong encryption, and other easy-to-avoid weaknesses. Although each individual household robot proved to be fairly innocuous on its own, this work reveals several complex attack vectors spanning multiple, differently purposed (and differently skilled) household robots that could lead to more serious attack payloads (e.g., combining dexterity-rich and vision-rich robots to hold and scan your keys for duplicates to be made by an attacker).

- ***Performance Testing the Vulnerability Response Decision Assistance (VRDA) Framework***
  Art Manion, CERT/Coordination Center; Kazuya Togashi, JPCERT/CC

Art Manion and Kazuya Togashi presented this work about the Vulnerability Response Decision Assistance (VRDA) framework, a decision support system used for predicting an organization's responses to vulnerability report stimuli. In order for the VRDA framework to be an effective tool, it must accurately predict the organization's responses relative to vulnerability reports. This work analyzes errors in VRDA predictions using several techniques (hit rate, off-by-one, mean of squared errors, and mean of the errors) in order to quantitatively evaluate the effectiveness of the VRDA framework. With the help of the Vulnerability Analysis team at the CERT/CC, this work offers a multitude of numerical error assessment results on various tasks for which the VRDA framework can predict responses.

MALWARE DETECTION AND PROTECTION

Summarized by Andres Molina (amolinaf@nsm.umass.edu)

- ***Effective and Efficient Malware Detection at the End Host***
  Clemens Kolbitsch and Paolo Milani Comparetti, Secure Systems Lab, TU Vienna; Christopher Kruegel, University of California, Santa Barbara; Engin Kirda, Institute Eurecom, Sophia Antipolis; Xiaoyong Zhou and XiaoFeng Wang, Indiana University at Bloomington

Clemens Kolbitsch introduced his talk by mentioning the weaknesses of existing malware detection solutions that rely on binary signatures or on the detection of artifacts. Kolbitsch suggests that a better solution to this problem would be to look for patterns of malicious behavior. He claims that a system based on detecting these patterns is harder to obfuscate and is more stable. A major contribution of the presented work is to provide a solution to detect malware that combines the effectiveness of behavior-detecting techniques with the efficiency of previous solutions based on binary signatures.

First, in a detection phase in which the characteristics of the malware's behavior are determined, the malware is executed in a full system emulator called Anubis. During this phase, the system monitors the interaction with the operating system. This observation allows the tool to perform a detailed analysis in order to generate behavior graphs. These detection graphs describe a sequence of required system calls leading to the security-relevant system activity, together with the dependencies to the related previous calls. Finally, the end host is monitored and all the system call activity of unknown executables is logged and matched against the behavior graphs obtained. The speaker described how this would work, using a trojan horse as an example. Clemens described their proposed way of matching behavior graphs using recorded execution semantics and then extracting data propagation and manipulation formulas.

The evaluation of the system covered aspects of the effectiveness and efficiency of the system, showing that the behavior detection is fast enough for the end hosts and that the approach is robust against any kind of binary obfuscation, including polymorphism and metamorphism. Furthermore, in some cases the system can detect malware variants that were never seen by the graph generator.

A member of the audience said that it is difficult to distinguish installers from malicious behavior, especially given that trojans seem to be the most common type of malware these days. Kolbitsch pointed out that trojans actually download other malware following very precise patterns, and the system described already detects these trojans. He noted that the authors plan to explore this kind of detection further in future work.

- ***Protecting Confidential Data on Personal Computers with Storage Capsules***
  Kevin Borders, Eric Vander Weele, Billy Lau, and Atul Prakash, University of Michigan

Billy Lau began his talk by noting that many users need to work with sensitive data, such as financial records, while using a PC that is not trusted. Lau proposed a solution to this problem using storage capsules. He started by describing a typical workflow for a solution using TrueCrypt, pointing out that such a solution should not be considered safe, because when the document is open it also becomes

available to malware. He said that Trusted Boot, another existing solution, is also not completely appropriate because all software is required to be verified before installation, and verifying documents is even more complex.

The authors' solution consists of dividing the modes of operation into normal and secure modes. In the normal mode, a user would face no restrictions but should perform only non-sensitive operations and would not have any storage protection. In the secure mode, network output would be prevented, but editing sensitive documents would be possible and changes would be encrypted to storage capsules. The speaker said that from the user's perspective the workflow should be very similar to using a solution such as TrueCrypt. The solution with storage capsules is implemented by running a primary virtual machine and a secure virtual machine. The I/O is restricted across different modules between the Secure VM, the Primary VM, and the VMM that handles the physical device drivers, while the system is in secure mode.

Paul Van Oorschot from Carleton University questioned the speaker's claim to offer a system with better usability, given that the authors did not conduct a study to evaluate this claim. Another member of the audience also pointed out that since the primary OS is not trusted and the viewer is run in this system, it may be hard for a user to trust the viewer. Lau and one of the co-authors responded that the viewer is not trusted in the proposed model, and it can be thought of as a malicious client. How would their solution compare to simply rebooting the machine into another more trusted OS, given that entering and exiting the trusted modes takes a couple of minutes? The transition in their system would still be faster than rebooting into another system. Would all covert channels be mitigated to avoid leakage from the Secure VM in this system? The authors did not claim to eliminate all covert channels. Instead, the intention was to eliminate as many as possible at the layer of abstraction provided by the authors. An audience member pointed out that buffer overflows may have to be addressed by this architecture. Lau noted that, while this point should be taken into account, the system is currently implemented in Python, a type-safe language.

- ***Return-Oriented Rootkits: Bypassing Kernel Code Integrity Protection Mechanisms***
  Ralf Hund, Thorsten Holz, and Felix C. Freiling, Laboratory for Dependable Distributed Systems, University of Mannheim, Germany

Ralf Hund noted that the kernel has elevated privileges and that not even virtualization solutions allow the detection of malicious programs at lower privileged levels. Hund argued that it is necessary to prevent malicious programs from executing in the first place. The key idea is that current integrity protection mechanisms do not protect against attacks in which the attacker re-uses existing code within the kernel to perform malicious computations. He explained how this procedure can be automated by providing a framework with three core components: a constructor, a compiler, and a loader which can currently be used in 32-bit Windows operative systems running IA-32.

Hund described the instruction sequences that are useful for performing these types of attacks. He also briefly described the designs of the so-called automated gadget construction, the compiler, and the loader. Hund gave a demo of the rootkit implementation, which can be used to traverse the process list and remove a specific process. This rootkit is only 6KB in size, and, as shown in the demo, can be used to eliminate the program Ghost.exe from the Windows Task Manager while the process is still running.

Hund concluded by claiming that the problem is malicious computation, not malicious code, and, therefore, code integrity itself is not enough to provide a secure OS. Future work would include the implementation of the rootkit in other operating systems in order to show its portability and to develop some countermeasures against this sort of attack.

A member of the audience asked if address randomization could prevent this kind of attack. Hund responded that indeed this could be effective in mitigating the attack but not in the way it is currently implemented, not even in Windows Vista, where the randomization is done in user space. Could something be done to the kernel compiler to prevent these attacks? Although possible, this approach would require the recompilation of every single system component.

INVITED TALK

- ***Hash Functions and Their Many Uses in Cryptography***
  Shai Halevi, IBM Research

  Summarized by John Brattin (jbrattin@student.umass.edu)

Shai Halevi divided his talk on cryptographic hash functions into four main parts. The first described some standard uses for hash functions; the second explained a motivating application for hash functions; the third described how hash functions should be used; and the last described some of the considerations taken into account during the implementation of Fugue, the IBM submission for NIST's SHA-3 hash function competition.

Traditionally, hash functions have been used to compress, encrypt, and authenticate data, or for error-checking. Hash functions can be used in digital signature algorithms, in message authentication codes, in pseudo-random number generators, and in key derivation functions. Halevi described briefly how one would go about implementing each of these things.

The important properties of a hash function are also the defining characteristics of a "random function": each output is equally (im)probable; collisions (e.g., a and b such that H(a)=H(b)] are hard to find; fixed points [e.g., a such that a=H(a)] are hard to find; and so on. The "random oracle paradigm" is a method of creating a cryptosystem in which you assume you have a random function, design a system

around that function, prove that that system would be secure, and then replace the ideal random function with a practical, somewhat nonrandom hash function. In most cases, this paradigm is successful, resulting in secure cryptosystems. When the paradigm fails, it is due to some nonrandom property of the hash function.

However, every hash function has some nonrandom properties which can be exploited, because every hash function is computable. There are degenerate cases of cryptosystems designed using the random oracle paradigm, which are provably secure with a random function but trivially breakable with any hash function. According to Halevi, the best way to minimize the effect of the nonrandom properties of hash functions is to rely on very weak security assumptions, i.e., to claim that a cryptosystem is secure only when several external conditions are met. Halevi described several systems with weak security assumptions, including enhanced target collision resistance (eTCR).

The final section of the talk concerned Fugue, IBM's submission for the NIST hash competition. Many modern hash functions rely on the Merkle-Damgard construction, which is an iterative method that hashes the first part of the message, then hashes that result with the next part of the message, and so on. However, this construction has very little intermediate state. At each iteration, a nontrivial amount of work is done to hash part of the message, but that work is all thrown away and only the hashed result is preserved for the next iteration. Alternatively, Fugue carries along 120 bytes of intermediate state, making it harder to find internal collisions, which have been used in attacks on previous hash functions, eventually leading to real collisions.

A member of the audience asked if a PRG could be used to extend a short salt into a longer salt in the eTCR scheme. Halevi replied that it could be done but that the resulting salt may be easier to guess—if the attacker knows the PRG and can guess the seed, then they've generated your salt and it's easier to break the eTCR. Dan Boneh noted that Halevi talked about Fugue's collision resistance but didn't mention other nice properties of random functions—fixed points, low Hamming weight, and so on. Halevi responded that second-preimage analysis worked out to be very similar to the analysis he gave, and pseudo-randomness would also work out to be similar in some ways. He hadn't looked into analysis of fixed points.

BROWSER SECURITY

Summarized by Todd Deshane (deshantm@clarkson.edu)

- ***Crying Wolf: An Empirical Study of SSL Warning Effectiveness***
  Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor, Carnegie Mellon University

In the talk, Joshua Sunshine described a user study done at Carnegie Mellon University (CMU), in which the goal was to study the effectiveness of SSL warnings presented to users by various browsers and also some custom warnings devised by their research group. Most, if not all, modern browsers warn about SSL certificate problems in the cases of domain mismatch, unknown certificate authority, or expiration. These warnings happen in two types of cases, either as a final protection against man-in-the-middle attacks or when a user is contacting a legitimate server. One example of a legitimate server that would issue an SSL warning is the Carnegie Mellon Library Web site, which uses the Carnegie Mellon certificate authority. Since most browsers don't have the Carnegie Mellon authority added by default, the users will see an SSL warning.

Josh showed three native browser warnings, from Firefox 2 (FF2), Internet Explorer 7 (IE7), and Firefox 3 (FF3), as well as two custom warnings, a single-page and a multi-page warning. He noted the change from popup style warnings (pre-2007) to warnings that take up the entire page (post-2007). He also described the shift from a default action of ignoring the warning and continuing anyway to browsers that make it very difficult to ignore the warning, often forcing the user to make more than a single click to get by. In the FF3 case, four steps are necessary to ignore the warning and get through. In fact, in one of the alpha releases of FF3 there were 11 steps.

Next, Josh described his team's principled approach to designing two custom warnings based on an online survey and some warning science guidelines. The outcomes of the online survey taught them that content sensitivity (what the user is currently doing) is important to take into account. Also, they learned that users will ignore warnings out of habit (they ignored the warnings in the past and nothing bad happened). From the survey, they also found that people confused the SSL warnings with much less serious warnings, such as sending unencrypted information over the Internet via a Google search. The warning science guidelines they used, as taken from applied psychology literature, were to avoid warnings when possible, clearly explain the risk, and provide straightforward instructions for avoiding the risk. In their custom multi-page warning, they first asked the user what type of Web site they were visiting (bank, e-commerce, other, don't know). If the user chose bank or e-commerce, then a second page appeared wgucg gave a severe warning, based on the FF3 phishing warning and using the most severe, red Larry icon. The only two buttons on that second page were "Get me out of here" and "Why was this blocked?" with a small link in the bottom right to ignore the warning. If the user chose "other" from the previous warning page (as the type of site), they bypassed the severe warning page and were directed right to the destination site. Their custom single-page warning, as opposed to the multi-page warning, was simply the red, severe warning page, without asking users the type of site they were visiting.

The user study had 100 participants, all CMU students, who were randomly assigned to the different warnings (FF2, FF3, IE7, custom single-page, and custom multi-page). SSL

warnings were shown to the user in two of the four tasks, the bank task and the library task. For each task, the users were presented with an alternative method for completing the task, such as calling or using a different Web site. The bank task required them to enter their credentials for the bank, while the library task didn't require any sensitive information. The hypotheses of the study were that the IE7 and FF2 warnings would be ignored at both Web sites, participants would likely obey the FF3 and the custom single-page warning on both sites, and participants who saw the multi-page warning would obey on the bank Web site but continue on the library site.

The hypotheses turned out to be validated, but with some interesting findings along the way. First, although it was less likely that users would continue to the bank Web site, even in the best case (the single-page warning), 45% of the users visiting the bank Web site still ignored the warning! Also, the FF3 warning turned out to be a significant deterrent, even in the library task. Finally, it was surprising that even though fewer people continued through the bank task than the library task, the difference was slight, generally only one or two users. Based on an exit survey, the authors found that their single-page warning was more effective at conveying the risk and intent of the warning. They also noticed that 4 out of the 20 users of FF3 confused the warning with the "Page not found" (404) error. The FF3 warning also caused the most (10 times as much) hesitation actions (clicking back, refreshing, re-typing URLs, etc.) than any other warning.

The CMU team feels there are weaknesses that need to be addressed with their multi-page warning, such as better context-sensitive information. Finally, they conclude that forcing users to do the right thing (such as using systems like Perspectives and ForceHTTPS) is the correct way to go.

Someone asked about the backgrounds of the participants. Josh responded that as students at CMU they were rather technical, many of them in technical master's degree programs, such as information networking and software engineering. What is the general applicability of the results? The results should be applicable to the general population; technical expertise did not seem to play a role in the actions of the users. What is the statistical significance of the graph comparing users who ignored warnings on the library task vs. those who ignored warnings on the bank task? The only two cases where there was statistical significance was for the single and multi-page warnings. Another questioner wondered about the recommendation to force safe practices and in turn get rid of self-signed certificates. Josh said that in an ideal world he would recommend getting rid of self-signed certificates and use Perspectives and ForceHTTPS, although this avenue may be too costly, and he surmised that IE developers would not support it. Had the users entered their actual credentials, which would imply that they could have captured 45% of users' actual credentials? Josh confirmed both were true. Did users have an alternate way to succeed in the task? Josh said that they were allowed to use the phone or visit a different Web site.

- ***The Multi-Principal OS Construction of the Gazelle Web Browser***
  Helen J. Wang, Microsoft Research; Chris Grier, University of Illinois at Urbana-Champaign; Alex Moshchuk, University of Washington; Samuel T. King, University of Illinois at Urbana-Champaign; Piali Choudhury and Herman Venter, Microsoft Research

Alex Moshchuk acknowledged the trend of users to store information in the cloud as opposed to on their own computers. He explained that the existing browser technologies, such as IE8 and Google Chrome, take the approach of protecting valuables on the desktop via sandboxing and other similar techniques, but fail to protect Web sites from stealing data from each other. Therefore, the authors set out to design their Gazelle browser to apply operating system concepts directly in the browser's security model: for instance, moving all of the complicated resource allocation, protection, etc., into a small, trusted, and simple browser kernel. Taking the OS analogy one step further, instead of treating users as principals they treat individual Web sites as the principals by putting them into protection domains. The real challenge arises when dealing with Web sites that embed content from other Web sites, as is done with mashups.

The design of Gazelle builds on the concept of same-origin policy by labeling content based on origin and isolating Web site origins into Web site principals. Further, they separate principals into principal instances if, for example, multiple browser tabs from the same origin are open. In their design, principal instances would run in their own processes, but finer-grained methods of separation (e.g., based on type-safe code) could be applied. The architecture of the system is a single browser kernel that mediates all resource access of the principal instances, including network, storage, and user display. Gazelle requires that principal instances use the Gazelle API to interact with the system and each other. Even Flash (and other plug-ins) would run in its own process and would interact via Gazelle function calls.

The Google Chrome and IE8 browsers combine content from different origins on the same tab into the same process, where a malicious Web site could compromise data from another in the same tab. In contrast, Gazelle separates each of the principals from different origins into its own process, thus protecting the data of another site even if one of the principals is compromised. The goals of Chrome and IE8 are reliability and stability (keeping the browser going even if another tab crashes), while the goal of Gazelle is to introduce more security by protecting principals from each other.

Next, Alex briefly discussed the backward compatibility vs. security trade-off and noted that it is a policy issue. He also discussed in detail the concept of display in Gazelle

and explained how to use display access control to limit both the creator window (landlord) and embedded content (tenant) in order to allow proper control of data or display. The concepts exist to some extent in modern browsers, but are intermixed too much and the complexity leads to bugs. Gazelle is designed to more easily protect against CSS history stealing and clickjacking by carefully processing events and maintaining the proper protection between principals in the browser kernel.

The implementation is in C# and makes use of the Trident rendering engine of IE. The evaluation described in the talk showed that Gazelle could perform on par with other modern browsers, specifically IE7 and Chrome, for single-origin pages and has a fair amount of overhead for multiple-origin pages. He noted, however, that Gazelle is still a research prototype and has not been tuned for performance. He noted several optimizations that could bring performance closer to Chrome and IE7, and he mentioned Xax and Native Client (NaCl) as examples.

The first two questions concerned the overhead and the feasibility of using Xax and NaCl. Alex responded that he didn't have numbers for the overhead involved, but he didn't think it would be a problem, based on the overhead of system calls they had been experiencing. He mentioned that they are considering implementing a new renderer for Gazelle so that they can take advantage of Xax or NaCl-style sandboxing. He considered the process of adding sandboxing support to be more of an engineering effort, since XaX, NaCl, and Google Chrome had already accomplished it. There was a question regarding plug-ins; Alex responded that they are prioritizing support for a Flash plug-in. Rik Farrow asked about their decision to support transparent frames (thus opening up the possibility for clickjacking). Alex explained that they are studying the top sites and that they considered backward compatibility issues. He noted that the policy of allowing same-origin transparent frames was just one possibility and that other policies are being considered by his team.

INVITED TALK

- ***DNS Security: Lessons Learned and The Road Ahead***
  David Dagon, Georgia Institute of Technology

  Summarized by Michalis Polychronakis (mikepo@ics.forth.gr)

During 2008, the DNS infrastructure underwent a major disruption due to a new and improved DNS cache poisoning attack. David Dagon presented a timeline of the events that led to the public release of the attack details, including the response steps the DNS community took to mitigate the attack, and discussed current and emerging DNS security issues that remain open.

In a poisoning attack, a recursive DNS server is forced to perform a lookup for which it does not have any cached answer and thus needs to ask an authoritative resolver. In the meantime, the attacker floods the recursive server with spoofed answers using the source address of the authoritative server. If the race between the misleading and the legitimate responses ends in favor of the attacker, then the relevant DNS cache entry will be changed to point to any malicious server the attacker chooses.

In order to succeed, one of the spoofed responses has to match the 16-bit query ID used in the original request. In the past, DNS poisoning was not very effective, because correct answers were cached for a long period, so the window of opportunity for the attacker was very limited. However, in early 2008 Dan Kaminsky disclosed a new poisoning technique that is not affected by the retention of legitimate responses and thus has no "wait" penalty. The technique exploits the fact that name server (NS) locations are communicated through updates included in the DNS responses. The attacker first sends a query for a random child label, such as abcd.example.com, and then floods the server with responses containing a malicious NS update. If the query ID field is not matched, the attacker does not have to wait, but repeats the attack immediately by requesting a different child label, e.g., wxyz.example.com, in essence making the attack only bandwidth-limited.

The response of the DNS community was immediate. Since enhancements like DNSSEC cannot be deployed in a very short period, most vendors chose an opaque mitigation technique that increases the size of the query ID by adding 16 more bits of randomness through source port randomization. With an unprecedented simultaneous effort, most DNS server vendors had released patches before any exploit details were released. However, port randomization is not a perfect solution, since it adds a considerable resource overhead, and in many cases it is negated by NAT devices that do not preserve the source port.

Almost simultaneously with the disclosure of the Kaminsky attack, David Dagon and his team proposed 0x20 encoding, a practical technique that makes DNS queries more resistant to poisoning attacks. DNS-0x20 increases the randomness of the query ID by mixing upper- and lowercase in the domain name in the query. Since almost all DNS servers preserve the mixed-case encoding in their answers, attackers now have to also guess the correct mixed-case encoding.

Port randomization and DNS-0x20 have been widely adopted. However, David presented results of studies which suggest that about 20% of the DNS servers remain vulnerable. Unpatched servers can be exploited by attackers for various malicious purposes, including email message interception.

DNS prefetching is another emerging issue that has started affecting the DNS infrastructure. DNS prefetching aims to improve the responsiveness of client applications, especially Web browsers, by aggressively resolving host names in advance, before the user actually requests them. For example, after loading a page, the browser can resolve the host names of all URLs in the page before the user actually clicks on any of them. In another example, as a person types in the

address bar, the browser can predict matching domain names and attempt to resolve them with the hope that the correct response will have arrived before the user actually presses the return key.

The spurious requests made due to DNS prefetching can have a major impact on the DNS infrastructure. For example, while typing a domain name ending in ".com", a request for the ".co" TLD may be sent before the letter "m" is typed, incurring extra overhead to the ".co" TLD servers. Furthermore, DNS prefetching can be abused for spam delivery verification using unique domains seeded in spam messages, or as a DoS attack multiplier, through nonce child labels in the victim authority that are posted in popular forums or spam messages.

DNS is also widely misused by malware. An old technique is to change the resolver or manipulate the "hosts" file of the victim computer to redirect traffic to a malicious server. Recently, malware uses "DNS agility" for botnet command and control operations. For instance, the Conficker worm used 50,000 unique domain names per day, making tracking the actual C&C servers challenging.

Finally, David presented the current state and future of DNSSEC deployment. DNSSEC uses public-key asymmetric cryptography to validate content in a zone, providing message integrity. This enables other innovative uses besides DNS mappings, such as distribution of PGP keys and email addresses.

In the Q&A session, Niels Provos asked about the actual severity of the DNS poisoning problem. David mentioned that the remaining unpatched servers are being exploited, but it is hard to actually detect an attack since the observer should be on path with the poisoning in order to detect it. However, the cost for mounting a successful HTTP redirection attack is significant, since it requires additional steps besides attacking a DNS server.

Bill Cheswick mentioned that another similar case of massive cooperation among vendors to fix a severe problem was the SYN packet attack in 1996. (See the article on p. 35 in this issue.)

WORK-IN-PROGRESS REPORTS (WIPS)

WiPs below summarized by Kalpana Gondi (kgondi@cs.uic.edu)

- ***Full-Datapath Secure Deletion***
  Sarah M. Diesburg, Christopher R. Meyers, and An-I Andy Wang, Florida State University

Sarah Diesburg presented this work focusing on erasing file data securely and completely. There exist previous solutions to secure delete, but these cannot guarantee fine-grained deletion. Diesburg is trying to come up with an extension to secure deletion solutions using a module for the Linux kernel to keep track of blocks to be deleted. The proposed approach is to delete not only the files but also the metadata of the files, so that no traces are left in the hard disk. Also, the module takes care of deleting the data residing at multiple locations (e.g., on flash drives). Complete removal is necessary for users' privacy when a large amount of data is to be deleted permanently—individual files from bank accounts, for example.

- ***Cacophony: BitTorrent over VoIP***
  Rhandi Martin and Angelos Stavrou, George Mason University

Rhandi Martin described Cacophony, a usability tool to enable information sharing when there are legal restrictions. It is like steganography where the information sharing is not noticeable to network traffic analyzers. The authors are proposing this solution especially for BitTorrent traffic, which may be blocked by educational institutions or ISPs. The main idea is to hide the BitTorrent traffic in VoIP. As a result this traffic will not be noticeable to the network administrators/packet inspectors who can watch the traffic to block any data. As the authors acknowledge, the main limitation would be to conceal the large amount of BitTorrent traffic given the number of connections. To resolve the size limitation, authors propose the use of multiple vectors such as teleconferencing traffic and relaying.

WiPs below summarized by Prithvi Bisht (bishtspp@yahoo.com)

- ***Easier Passwords for the iPhone***
  Bill Cheswick, AT&T Research

Bill Cheswick presented a system that enables users to quickly and securely enter passwords on the iPhone. Cheswick mentioned that hard passwords are, ironically, hard to remember. This problem is aggravated for mobile phones due to their small form factor, and may make entering such passwords difficult and insecure (e.g., with increased time-to-enter, users can fall prey to shoulder surfing attacks). According to Cheswick, this work attempts to bring "usability" and "security" together. To retain strong entropy of passwords, Cheswick proposed using a combination of multiple, easy to remember, dictionary words as passwords. To allow users to quickly enter the password, Cheswick proposed the iPhone's spell checker feature to reduce the burden on users to enter all words correctly and, hence, reduce the time needed to enter passwords.

- ***Lightweight Information Tracking for Mobile Phones***
  William Enck and Patrick McDaniel, Penn State University; Jaeyeon Jung and Anmol Sheth, Intel Labs Seattle; Byung-gon Chun, Intel Labs Berkeley

William Enck presented an approach to implement information-flow tracking in mobile phones. He briefly touched upon the lack of security in existing mobile phones and presented an approach to enable taint tracking in Android Virtual Machine. According to Enck, mobile platforms are amenable to such taint tracking, as most sensitive information is retrieved from single-purpose interfaces and thereby allows for automatic tagging. He further indicated the possibility of enforcing precise security policies in the pres-

ence of system-wide taint information. Preliminary results indicate that the taint tracking does not adversely impact the performance of the system.

WiPs below summarized by Patrick Wilbur (patrick.wilbur@gmail.com)

- ***The OSCKAR Virtualization Security Policy Enforcement Framework***
  Todd Deshane and Patrick F. Wilbur, Clarkson University

User applications and virtual appliances (applications packaged within virtual machines) can be difficult to secure and, upon distribution, can potentially be run in environments that are ill-equipped to meet their unique security requirements. Furthermore, without clear environmental awareness of the specific needs and behaviors of applications and virtual appliances, malware can exploit vulnerabilities and wreak havoc on the installed system and others. This work develops an extensible policy enforcement framework and contract specification—where the application package maintainer who knows best can clearly define the application's needs and behaviors—to mitigate malware problems by eliminating the majority of their payload.

- ***An Introduction to LR-AKE (Leakage-Resilient Authenticated Key Exchange) Project***
  SeongHan Shin and Kazukuni Kobara, Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST), Japan

SeongHan Shin described Authenticated Key Exchange (AKE) protocols as a core cryptographic primitive, and these are used for establishing both mutual authentication and secure channels of communication. Traditional AKE protocols assume shared secrets are and will remain secure; in practice, however, these shared secrets are often leaked, resulting in the cryptographic system becoming insecure. The Leakage-Resilient Authenticated Key Exchange Project works to design a new Authenticated Key Exchange method that mitigates the risks associated with the leakage of shared secrets, while also recognizing the threat of dictionary attacks on traditional password-based protection of shared secrets.

- ***Towards Exploitation-Resistant Trust Models for Open Distributed Systems***
  Amirali Salehi-Abari and Tony White, Carleton University

Amirali Salehi-Abari noted the importance of trust models in establishing agent reputation within distributed systems, where the reputation of an agent is formed from information collected from those who have interacted with the agent in the past. This work adds exploitation resistance to the traditional trust model criteria. This requires the trust model to also protect against the exploitation of the system by an adversary who understands the internal workings of the trust model in use. This work thwarts trust exploitation by an individual agent by cautiously incrementing trust after defection and issuing larger punishments after each defection.

- ***Scalable Web Content Attestations***
  Thomas Moyer, Penn State University

As Web content is received, a Web user can tell which server the content is being sent from, but a user generally has no indication of the integrity or authenticity of the content they are viewing. This work attempts to better inform the user of the level of content integrity by leveraging TPM and integrity measurement technologies. This work provides a clear binding between the state of a server hosting content and the content being served, and does so efficiently despite the slowness of conducting TPM operations.

- ***Exploring the Trusted Computing Base of User Applications***
  Hayawardh Vijayakumar, Penn State University

The Trusted Computing Base (TCB) of an operating system is large, consisting of both the kernel and trusted programs, despite the fact that an individual user application might only need to interact with parts of the kernel and some trusted applications. Based on the hypothesis that typical user-space applications use only a small fraction of the TCB that runs on a typical OS, this work looks at applications and attempts to find what that fraction is. This work then deploys an application in separate virtual machines equipped only with the dependencies the application needs.

- ***Further Improving Tor's Transport Layer***
  Chris Alexander, University of Waterloo

Tor's current transport layer employs a user-level TCP stack and Round Robin to transfer data and multiplex data across common paths, which poses problems with resending portions of multiplexed data as well as fair congestion control in accordance with the primary purposes of Tor. This work replaces the user-level TCP stack with user-level Stream Control Transmission Protocol (SCTP) in order to decrease memory requirements and allow customization of the congestion control mechanism. Furthermore, this work replaces the round-robin scheduling so that bursty traffic (e.g., HTTP) can compete with large, steady traffic (e.g., BitTorrent), which is fairer for traffic closer to Tor's primary purposes (i.e., anonymity and censorship circumvention).

WiPs below summarized by Asia Slowinska (asia.slowinska@gmail.com)

- ***Challenges in Sinkholing a Resilient P2P Botnet (Is It Possible or Not?)***
  Greg Sinclair, iDefense/UNCC; Brent Hoon Kang, UNCC

Brent Hoon Kang presented the layered architecture of Waledac, a P2P botnet, and described the resilient protection mechanisms that Waledac has employed to protect the botnet against common mitigation efforts. The hierarchy introduces a number of layers, starting from spammer nodes at the bottom. Above these come repeater nodes, which forward messages to/from higher parts of the hierarchy. Subsequently, the TSL layer protects the bot master located at the top. The research conducted demonstrates that Waledac introduces a number of mechanisms preventing it from

being sinkholed. For example, taking down the TSL layer nodes may result in converting some of the repeater (or new) nodes to new TSL nodes. Also, unlike the Storm botnet in the past, the new information about new TSL server lists is signed by the private key of the upper layer, making any modification to TSL information impossible.

- ***Advanced Metering Infrastructure Security Analysis***
  Steve McLaughlin and Patrick McDaniel, Penn State University

The Advanced Metering Infrastructure (AMI) is a set of smart meters and communication networks forming the basis of smart grid. These smart meters observe one's energy consumption and communicate that information to customers, the local utility, and the grid. Steve McLaughlin presented his research on the security implications of the meter functionality. The penetration testing that was conducted employs an attack tree methodology, where the root of the tree defines a goal, e.g., committing energy theft. The subgoals are defined as the child nodes. Finally, the leaves determine the types of attack to mount. The results identify numerous vulnerabilities and exploits possible: by unplugging a meter from a phone connection at the right time, for example, one will be able to impersonate the meter using a regular computer and forge demand values. The work is published at CRITIS 2009.

- ***Enhancing the Formal Cyberforensic Approach with Observation Modeling with Credibility Factors and Mathematical Theory of Evidence***
  Serguei A. Mokhov, Concordia University

Serguei Mokhov presented his work on refining the formal cyberforensics approach by Gladyshev to model cybercrime investigations, evidence, and witness accounts in order to reconstruct the event and verify whether a given claim agrees with the evidence. With the invention and use of an intensional programming language called Forensic Lucid, he improves Gladyshev's finite state automata (FSA) approach in order to increase the usability of the entire system. Current work is being conducted to enable defining credibility of witnesses or evidence in a case using the Dempster-Shafer mathematical theory of evidence.

- ***Decompiling Android Applications***
  Damien Octeau, Penn State University

Damien Octeau presented his approach to decompiling the Android application bytecode back to Java source code. He claimed that this might prove useful for several reasons: for instance, to unearth security policies buried in the source code, which is inaccessible if one holds a binary only. Android applications are written in Java but run in a virtual machine called Dalvik VM, which significantly differs from the traditional JVM (Dalvik VM is register-based, for example, as opposed to the stack-based JVM). Octeau built a tool for converting Dalvik executable files (.dex) into new Java bytecode files (.class), which can be further processed by existing Java bytecode tools to recapture the original source code. Initial results show that the method is effective.