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

July 29, 2008
San Jose, CA

SECURING SYSTEMS

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

- ***Towards Application Security on Untrusted Operating Systems***
  Dan R.K. Ports, MIT CSAIL and VMware, Inc.; Tal Garfinkel, VMware, Inc.

The first talk of the day was given by Dan Ports. The motive for his research was the need to run critical applications on commodity operating systems. These OSes may be quite complex, leading to a large trusted-computing base and weaker overall security. Recently, researchers have begun investigating methods for protecting applications from a malicious underlying operating system with a trusted lowest-layer module that encrypts application memory and protects execution state. However, execution state and memory are only part of the story. This paper explores what can happen when the operating system attacks an application by providing unexpected system call behavior. There are a number of system calls on which applications rely for secure and correct execution. This includes, but is not limited to, file system, synchronization, clock, random number generator, and inter-process communication calls.

The authors propose to fix incorrect system call behavior from a compromised operating system by having a separate trusted module verify system call correctness. For file system calls, this may mean storing a hash value alongside each data block. For synchronization calls, the module could verify the correctness by making sure a lock is only given to one thread. In general, the amount of work needed to verify correctness of system calls is significantly less than reimplementing the calls. For example, the trusted module would not have to handle scheduling and fairness to verify synchronization calls.

Various members of the audience asked about the difficulty of verifying correctness for all system calls in practice. It may be hard to check results from calls such as ioctl() which have a wide range of parameters and expected behaviors. Furthermore, results coming from the OS may be correct but could compromise the application by returning unexpected values that the application does not handle properly.

- ***Digital Objects as Passwords***
  Mohammad Mannan and P.C. van Oorschot, Carleton University

Mohammad Mannan described a new method for creating passwords, motivated by the inherent inability of people to select and remember good passwords. The goal of this research is to create a strong password that is also easy to remember, similar to the way that a personal question, such as your mother's maiden name, is memorable. The solution that Mannan presented is using an object on your computer or on the Internet as your password. The system will compute a hash value of the object that, when combined with a short salt, will yield a secure text password of a predefined length. The password is both easy to remember, because it is associated with a logic object, and secure, because it is derived from the hash of a large object.

Mannan also discussed limitations of the object-as-password approach. First, looking over someone's shoulder is much easier. For network objects in particular, the system is also vulnerable to snooping. If an attacker can see what objects you are looking at, then the space for a password dictionary attack is fairly small. This line of attack is difficult to prevent in general, but might be mitigated by using Tor to anonymize traffic. Another limitation of using objects as passwords is a lack of portability. It is more difficult to carry objects around when you are in a remote location. The fail-over solution suggested by the authors is writing down the long and difficult-to-remember password generated from the object in this case.

The audience raised concerns about whether users would gravitate toward the same types of objects as passwords, especially if selecting them from the Web, and thus reduce security by still choosing bad passwords. A usability study would be necessary to test the true security of an object-as-password approach.

A Firefox plug-in implementing the presented research can be found at http://www.ccsl.carleton.ca/~mmannan/obpwd.

- ***Security Benchmarking using Partial Verification***
  Thomas E. Hart, Marsha Chechik, and David Lie, University of Toronto

David Lie began by pointing out that software has a lot of vulnerabilities. There is no current way to measure them other than waiting for vulnerability disclosures. It would be nice to have a system for preemptively estimating the potential number of vulnerabilities based on code analysis. In this paper, the authors present a metric for classifying a particular piece of code's susceptibility to so-called mechanical vulnerabilities—those that are independent of program logic, such as buffer overflow or SQL injection. The long-term goal of this research is software security through quantitative measurement.

The security benchmarking system in this paper involves automatically pre-pending all potentially insecure operations with assert statements that check program state to see whether an exploit is possible. For a buffer overflow, this would be a check of the buffer length. The next step is to use a theorem prover to see which of these assertions are always true, and which cannot be verified. Finally, a program would be scored based on the quantity of unverifiable assertions.

A member of the audience pointed out that it is important to have a good theorem prover; otherwise, programs may be deducted for code that is safe. Another person asked whether coding style has an effect on the count of unverifiable asserts and thus the count might be unfairly prejudiced against certain styles, even if they are not less secure. However, in the experience of the authors, code that is easier to verify is also easier to read, and thus it is probably more secure by nature.

EXPLORING NEW DIRECTIONS

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

- ***Securing Provenance***
  Uri Braun, Avraham Shinnar, and Margo Seltzer, Harvard School of Engineering and Applied Sciences

Uri Braun from Harvard University gave this presentation about securing provenance. Provenance is metadata that represent the history of an object—the "when" of knowledge. Examples can be found in financial transactions, a lab notebook, etc. Provenance has some important properties: (1) It is append-only (i.e., you cannot change history); (2) it can be represented by a directed acyclic graph (i.e., you cannot have a circular dependency in time); and (3) you can add attributes to old nodes. The guiding principle behind this research is that provenance has different security properties from normal data, and it thus needs different control mechanisms.

The primary example used in the presentation is construction of a performance evaluation for an employee that has multiple anonymous contributors. It is important to be able to verify that the report was generated by multiple sources, but the employee should not be able to tell who they are. A variant on this is a graduate school application, where the student should be able to control who writes recommendations but should not be able to read the results of the recommendations. Security policies for these situations are complicated and not easily satisfied by current approaches, such as access control.

Audience members asked whether provenance wasn't just a special case of the general metadata handling problem. Causality and history are just some metadata that needs to be addressed by security policies. The database community has done some work in this area in the form of mechanisms for handling complex conditions but has not directly addressed securely handling provenance, but existing work may be applicable to solving these problems.

- ***Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security***
  Tamara Denning, University of Washington; Kevin Fu, University of Massachusetts, Amherst; Tadayoshi Kohno, University of Washington

Tamara Denning presented some follow-on research to recent work on attacking implanted medical devices (IMDs). A variety of such devices exist, and securing them from attack is essential for maintaining the wearer's safety. Implantable medical devices are particularly challenging from a security perspective, because of their limited capabilities. They have little power with which to perform cryptography or resist a battery-draining attack. An even more significant limitation is that any security system for IMDs must allow emergency personnel to override protection. For this reason, today's IMDs are fairly open, leaving wearers susceptible to serious attacks.

This research explores a number of potential solutions for securing IMDs. First, the presenter discussed approaches that are insufficient either because they would be too closed in the case of an emergency or would offer only weak protection. Case-by-case access controls would be safe, but would be too closed for emergencies. A user alert when communication with the IMD is taking place would be too open, because the wearer may not be able to respond adequately to an attack. Requiring very close proximity for communication would also be too open, because an attack could take place at close range. A member of the audience asked about combining these approaches to come up with the right solution, but a combination including case-by-case access would still be too closed. Another possible approach that would be too closed is carrying a password card for access to the IMD.

Consideration of other alternative design options led the authors to their proposed solution: have a device known as a cloaker that suppresses access when worn by the patient. The cloaker could have a wristwatch form factor that would allow emergency crews to remove it. Interesting research problems remain for the proposed solution. What is the best way for the cloaker and IMD to communicate? The presenter and audience also briefly discussed usability and psychological factors associated with wearing a cloaker. Such a device may serve as a reminder of the IMD's presence and be undesirable for some wearers. There also may be cases where emergency staff cannot reach the cloaker (e.g., if the person's hand is trapped in a car). Overall, a cloaker-based approach to securing medical devices gives desirable openness and safety properties, but there is significant research left to be done on the effects of IMD cloaking devices.

- ***Research Challenges for the Security of Control Systems***
  Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry, University of California, Berkeley

Alvaro Cardenas presented research on securing control systems for physical processes. These systems are responsible for keeping our critical infrastructure—the power grid, sewage treatment plants, and others—up and running. Security is essential for physical control systems because a compromise can lead to physical damage and danger. However, there have not been any recorded attacks, so why should we care? The reason is that everything is increasingly connected and complex, exposing new vulnerabilities. Another motivation is cybercrime. Reports from the CIA have alluded to the occurrence of extortion based on attacks on physical control.

An important question to consider is whether the problem of securing physical control systems is any different, fundamentally, from securing conventional computers. One notable dissimilarity is that physical control systems can be designed with algorithms that are resilient to attack by software. However, a sustained denial-of-service attack, which, as a member of the audience mentioned, is probably easy, can also lead to unsafe conditions. Research on physical control systems is needed to characterize vulnerabilities and come up with realistic active attack models.

ADVERSARIAL SECURITY

Summarized by Alexei Czeskis (aczeskis@cs.washington.edu)

- ***Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications***
  Alexei Czeskis, David J. St. Hilaire, Karl Koscher, Steven D. Gribble, and Tadayoshi Kohno, University of Washington; Bruce Schneier, BT

Alexei began his presentation by showing that the state of the art in file privacy—whole-disk encryption—was no longer sufficient, because of recent legislation that requires individuals to give up their laptops and any electronic media to customs agents without cause. Furthermore, he said that passwords could be extracted from the user via such coercive means as fines, jail time, or even physical torture. Next he told us that privacy advocates are suggesting the use of a software package called TrueCrypt that offers a deniable file system feature (also called a steganographic file system), which hides the existence of data from an attacker. Alexei then explained that TrueCrypt provides a deniable file system by allowing the user to undetectably create an arbitrary number of nested, encrypted, and hidden containers within an encrypted container. Each nested container could only be read if the appropriate password was provided. The existence (or nonexistence) of a nested container could not be proved by looking at the properties of memory, allowing the user to claim that no data existed. Alexei said that although this may be true if one solely looks at the bottom layer of a system, it does not hold if one considers the large ecosystem of operating system and applications in which a user interacts with the files in a hidden container.

Alexei said that his team analyzed TrueCrypt v5.1a and found that the system leaked enough information for an attacker to determine that the system had a hidden volume installed on it. Information leaks could be grouped into the following categories: (1) operating system; (2) primary applications; (3) nonprimary applications. Primary applications are ones the user interacts with daily; nonprimary applications may run in the background and be supplemental to the user's overall goals while using the system. The analysis only considered an attacker that has one-time access to the system. Although stronger attackers could have more frequent access to the user's system, Alexei explained that this work tries to show that the state-of-the-art methods for hiding data do not protect against even the weakest attacker.

The operating system (Microsoft Vista) leaked information via the recently used shortcuts list, revealing the real file's name, location, creation time, modification times, access time, volume type, and serial number. The primary application (Microsoft Word) leaked information via automatically generated auto-recover files that were not securely deleted and were recoverable even after a power cycle. The nonprimary application (Google Desktop) leaked information by caching and indexing files from the hidden container.

Alexei concluded by stating that this problem was not specific to TrueCrypt's implementation. Rather, he mentioned that it is very difficult to hide the existence of data on a system while at the same time providing a usable system in which there is a balance between isolation of components that must stay separate and sharing of components that must coexist for usability. Finally, he gave several examples of other methods a DFS may implement: using tainted data flow in the OS, a selective bootloader (implemented by TrueCrypt 6.0), and hard-drive firmware that will fake corrupted sectors until a particular sequence of reads permits them to be unlocked.

- ***Panic Passwords: Authenticating under Duress***
  Jeremy Clark and Urs Hengartner, University of Waterloo

Jeremy first showed a clip from a Hollywood film in which a secret agent is forced, under gunpoint, to call her superiors in order to obtain confidential information. As the agent's superiors ask her for a password, the audience is shown that she has two possible answers: one to indicate a legitimate authentication and the second (called a panic password) to indicate that she is authenticating under duress. Next, Jeremy formally defined panic passwords (or distress password/duress codes) as schemes that allow a person to indicate that the authentication attempt is made as a result of some coercive action. Although commonly used as a part of larger systems (e.g., home alarm systems), panic password schemes are rarely discussed in patents and in academia. Jeremy then presented a threat model, examined a common panic password scheme, and explained why it failed to fully succeed in its objective. He proposed three new schemes and described their associated analyses.

Jeremy's analyses were based on the assumptions that an attacker: (1) knows how the system works; (2) is able to observe the communications; (3) can force the user to iterate the process some finite number of times; (4) can force the user to disclose passwords in any particular order. Furthermore, each analysis was characterized based on the following parameters: (1) attacker's persistence (i.e., how long an attacker can interact with the user); (2) communicating parties' responses (i.e., whether it may be indicative of the legitimacy of a given password); (3) the attacker's goal (i.e., whether to know that a panic password was given or to force the user at some point to reveal the real password); (4) screening versus signaling (i.e., how well the user can trick the attacker into thinking that he or she entered a legitimate password and vice versa).

These parameters were then used to examine several panic password schemes. The most ubiquitous scheme, called 2P, involves the existence of two passwords: one good and one bad. However, this scheme is defeated by forced randomization: asking the user to enumerate the known passwords and then choosing one of them for the user to enter, thus giving an attacker a 50% success ratio, which means that the user loses since the threat model permits the attacker to iterate any number of times. If the attacker is nonpersistent, then a possible solution to this problem is the 2P-Lock, in which an alarm is triggered if two different passwords are used within a short period of time. Another scheme, called P-Complement, assumes one legitimate password and all other responses result in an alarm. This approach suffers from a high false-negative rate. The last approaches, called 5-Dictionary and 5-Click, involve the user entering five words and clicking five images in a particular order, respectively. All other entries far enough from the legitimate password (using some distance metric) are defined as panic passwords. That is, a password with one typo does not result in a panic; rather, it is just deemed an invalid password.

Jeremy concluded by pointing out that all of these scenarios seem like Hollywood stories, but they do have applicability to home security systems, intelligence agencies, and electronic/on-line voting. One questioner mentioned that human reactions play a larger role, indicating that he reacts differently when he lies. Jeremy agreed, but said that most likely no one will be holding a gun to his head for his vote.

- ***Bootstrapping Trust in a "Trusted" Platform***
  Bryan Parno, Carnegie Mellon University

Bryan began his presentation by telling the audience that he saw a pop-up notice for an update for a trial version of a key logger on the screen of a computer he was about to use in an Internet café. The rest of his presentation dealt with the question, "How can you trust any given computer?" Bryan made the following assumptions: (1) we have a trusted mobile device; (2) someone will be able to vouch for the physical safety of the system in question (i.e., the hardware will do what it's supposed to do). Bryan's proposed solution to how we might bootstrap trust in a system revolves around the use of a Trusted Platform Module (TPM)—a security chip equipped with a public/private key pair that can be used in conjunction with hashes (stored in the TPM) of installed software on the system to attest to the software state of the system.

Trust in a system can be bootstrapped iteratively, with the user's mobile device checking the computer's bios, the bios checking the bootloader, and the bootloader checking the kernel, which then checks all applications. However, this approach falls prey to the cuckoo attack, in which malware can reroute communication between the local machine and its TPM to a different machine, which the attacker controls and in which the attacker can modify hardware. The logic framework for analyzing this boot process is presented in the paper and was not presented by Bryan. One solution may be to cut network traffic during the trust bootstrapping procedure. However, this is also problematic, because malware may act as a fake TPM with a legitimate private key obtained from a different TPM that an attacker possesses.

The root of the problem seems that the user has no secure channel to the TPM. Bryan presented two methods for solving this. The first revolves around the "seeing is believing" principle, in which the public key of the TPM could be contained on a sticker affixed to the exterior of the computer. A second approach is more blunt: requiring a special-purpose interface to communicate directly with the TPM. Bryan suggested that the first be used in the short term but that the latter be adopted as a more solid solution.

NETWORK FORENSICS

Summarized by Dan Ports (drkp@mit.edu)

- ***Towards Quantification of Network-Based Information Leaks via HTTP***

Kevin Borders, Web Tap Security, Inc.; Atul Prakash, University of Michigan

Kevin Borders discussed the problem of detecting unauthorized disclosure of confidential information via the network. Current data-loss prevention systems scan outgoing network traffic for known patterns of sensitive data, and so are easily foiled by encryption or obfuscation. Instead, he proposed detecting suspicious behavior by quantitatively measuring the amount of outbound information flow and comparing it to a baseline value.

Kevin observed that, although the raw outgoing byte counts for HTTP traffic are large, the actual information content is much smaller. For example, a form submission contains many lines of header information in addition to the submitted form values, and even the values themselves may simply be default values.

Formally, the problem is to compute an upper bound on the amount of outgoing user-originated information, using network measurements and protocol knowledge. This involves measuring the size of the outgoing request but discounting expected values, such as HTTP headers that remain unchanged from the previous request or Referer headers containing the URL of a previous request. For GET requests, the address being fetched may leak information. The full length of the URL is counted if it was previously unseen; the information content of links followed from a previously accessed page is proportional to the logarithm of the number of links on that page, unless the accesses are for mandatory links (e.g., images) in the proper order. For form submissions, the edit distance between the submitted and default values is measured. Active JavaScript applications may send custom HTTP requests, which are currently counted by measuring the edit distance from frequent requests; analyzing the JavaScript to better understand its network behavior will be a goal of future work.

Kevin showed that these techniques substantially reduce the amount of measured information flow. On several typical Web browsing sessions, the new techniques gave a 94%–99% reduction in byte count relative to simply measuring raw traffic volume and a 75%–99% reduction relative to the simpler techniques from Kevin's earlier work on Web Taps (in CCS '04). The best results came from pages with minimal JavaScript usage.

In response to a question about whether this technique could be applied to other protocols, Kevin responded that it would work well for other protocols where most of the data is predetermined, such as instant-messaging protocols. It would not be as helpful for protocols such as SMTP, where most content is actually user-generated.

- ***Principles for Developing Comprehensive Network Visibility***

Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, and Nicholas Weaver, ICSI

Vern Paxson proposed a design for a network monitoring system based on the principle of unifying the analysis process across time—combining analysis of past history with real-time monitoring—and space—integrating information from many different sources. The monitoring system would operate primarily at the scope of an administrative domain, such as an enterprise or department. This scope is broad enough to provide interesting information but narrow enough to make collecting and understanding data practical.

The key to unifying analysis across space is combining events recorded from many different sources into a common data model. This information would span different abstraction levels: An event might represent a packet being seen, a TCP connection being established, or a URL being fetched. The data should be policy-neutral, such as recording packets rather than IDS alerts, in order to provide more flexible analysis. Because many attacks take place over long time intervals, the monitor needs to keep extensive history. Making this feasible requires discarding some data; Vern proposed discarding most of the bytes from the relatively few large connections that consume most traffic (i.e., keeping only the first 20 kB of each connection), then gracefully degrading history over time, making it more coarse-grained.

With this data aggregated and stored in a common data model, operators can then develop queries to analyze the data. Vern advocated using a common framework to develop queries that can be used both to perform retrospective analyses and to analyze a stream of events as they arrive. Besides eliminating the need for parallel development of two different programs, this enables "what-if" analysis to better understand the effectiveness of newly developed rules.

Vern also discussed extending this approach to perform analysis beyond a single site. Most proposals assume a global scale, which brings with it many trust issues (e.g., one site might not trust another with its network logs or might worry about the other site providing false data). Instead, he proposed limiting the scope to sites with co-aligned threat models and administrative ties, which may already work together informally today. One site would be able to send a query to another site, which could return the results of past analysis or install it as a trigger to detect future activity. Many attendees were concerned that this might cause sensitive data to be leaked to another site, but Vern explained that data itself is never shared, and each site's operators can decide on a per-query basis whether to allow another site's query. Essentially, this is a more structured version of the ad hoc coordination that often occurs between sites via telephone and email.

■ ***Challenges and Directions for Monitoring P2P File Sharing Networks—or—Why My Printer Received a DMCA Takedown Notice***

Michael Piatek, Tadayoshi Kohno, and Arvind Krishnamurthy, University of Washington

Michael Piatek began by observing that availability of copyrighted data on peer-to-peer networks has not gone unnoticed by the media industry, which crawls peer-to-peer networks to identify infringing users and take legal action against them. However, most current monitoring techniques are inconclusive and can be manipulated.

In the BitTorrent protocol, all clients interested in downloading a particular file contact a tracker to obtain a list of other peers and add themselves to the list. They then communicate directly with the other peers to download the file data. Monitoring agencies working for the media industry have two main approaches for identifying the IP addresses of offenders: direct identification, where they actually contact peers and download data from them, and indirect identification, where they rely on the tracker's word that a particular peer is sharing the file. Indirect identification is most common because it is substantially less expensive, but it may lead to false positives.

Michael and his colleagues at the University of Washington experienced this firsthand while conducting a measurement study of BitTorrent traffic. Their measurement involved a crawler that connected to many BitTorrent trackers to obtain membership lists, but it did not actually upload or download any traffic. Nevertheless, they received a number of DMCA takedown notices. Following this result, they conducted a second study to determine whether they could falsely implicate a different IP address in file-sharing and cause it to receive DMCA takedown notices. This was sometimes possible, because some trackers allow a joining client to register under a different IP address from that of their network source address, to aid in NAT traversal. Using this technique, they were able to attract 18 complaints for IP addresses associated with hosts that were not running BitTorrent, including printers and wireless access points. However, they also received many more complaints for the machines being used to launch the attack, indicating that most trackers do not support this protocol extension. Someone asked whether network-level spoofed source addresses could be used to frame a different IP, but Michael responded that this was not possible, because tracker connections either use TCP or a two-way handshake protocol with UDP.

Michael concluded by likening the world of peer-to-peer monitoring and enforcement to the Wild West. Enforcement agencies detect copyright violators using arbitrary techniques and report them to ISPs, who respond with arbitrary penalties. More accurate techniques are available, but they are costly. Monitoring organizations should use direct identification, but this increases the bandwidth costs by a factor of 10 to 100. ISPs should involve more human intervention and sanity-checking in the enforcement process, but instead the current trend has been to increase automation to reduce costs. Finally, this work considered the problem of identifying infringing IP addresses, but even if this is accomplished perfectly, it remains challenging to reliably associate an IP address with a user.