conference reports

THANKS TO THE SUMMARIZERS

Kevin Butler

Ming Chow

Jonathon Duerig

Serge Egelman

Boniface Hicks

Francis Hsu

Stefan Kelm

Mohan Rajagopalan

CONTENTS OF SUMMARIES

14th USENIX Security Symposium

Baltimore, Maryland
July 31–August 5, 2005

Keynote Address

- *Computer Security in the Real World*

Butler W. Lampson
Summarized by Stefan Kelm

As in the past, this year's keynote was given by someone well versed in dealing with security issues. Butler Lampson opened his talk by comparing real-world security to computer security. Real-world security is not usually about locking things (or people) up but, rather, is about risk, locks, and deterrence. Risk management, Lampson argued, is important there, since the main issue often is how to recover from an incident at an acceptable cost. Part of this is accountability: unless you can identify the bad guy, you will not be able to deter him. Accountability needs to be enforced at the "end nodes," i.e., "all trust is local."

Senders of network packets need to be held accountable for their actions. ISPs, for example, should cooperate when trying to stop DDoS attacks. "How much security?" Lampson asked, and argued that the main goal should be feasible security, stating that "perfect security is the worst enemy of real security." Applications or operating systems must not become unusable due to bad user interfaces.

Lampson then began a lengthy and fairly technical discussion on access control. His main example was that of someone wanting to access a Web page securely. Authentication and authorization are very often confused, he said, but need to be clearly differentiated. He said that fine-grained access control was a mistake. Moreover, there is a need for solid auditing mechanisms, which one especially needs for deterrence.

He also discussed secure channels, which in his usage do not refer to physical network channels or paths but to a more general concept. He provided a few examples, such as SDSI/SPKI and ACLs. Closely related is the issue of securely authenticating programs upon loading. Being with Microsoft, Lampson brought up NGSCB/TPM and surprised the audience by saying that "it's been put on the shelf" ("I do not believe in the DRM stuff at all," he said), especially since nobody has figured out how to keep the TCB small, a key requirement.

Some of the questions and answers focused on access control and the problems of humans giving away their identity. Curiously enough, Lampson's reply to one question, "If you want your machine to be moderately secure you need some form of remote administration," seems to contradict his earlier "all trust is local" statement. To a question about being sure one's configuration is correct, Lampson replied, laughing, "You want perfection and you're not gonna get it!"

His talk can be downloaded at http://www.usenix.org/events/sec05/tech/lampson.pdf.

For more information, see his home page at http://research.microsoft.com/lampson.

Refereed Papers

SECURING REAL SYSTEMS

Summarized by Kevin Butler

- *An Analysis of a Cryptographically Enabled RFID Device*

Steve Bono, Matthew Green, Adam Stubblefield, and Avi Rubin, Johns Hopkins University; Ari Juels and Michael Szyydlo, RSA Laboratories

- ***Awarded Best Student Paper!***

Steve Bono presented his group's work on analyzing Texas Instru-

ments' (TI) Digital Signature Transponder (DST). This is a passively powered device used in vehicle immobilizers by automobile manufacturers such as Ford. It is also used in the ExxonMobil Speedpass, a device that can be used in lieu of cash or credit cards at gas pumps. The DST provides security based on a challenge-response protocol, where a 40-bit key challenge is issued from the reader to the transponder and a 24-bit response is returned by the transponder, along with its 24-bit serial number. The serial number can only be written by the manufacturer, and the response is encrypted by a 40-bit secret key.

Bono outlined the methodology used to examine the security of the DST system. They set out to discover whether it was possible to recover the proprietary secret algorithm used by the device, purchasing an evaluation kit from TI and testing against the device with structured bit patterns for the challenge issued. A diagram published by TI on the protocol was used as a general schematic to verify against, and through experimentation, the group verified the diagram and made tables outlining the operation of the substitution boxes therein. In this manner, the entire cipher was uncovered.

The 40-bit key used was found to be small enough to be vulnerable to a brute-force attack. While general-purpose CPUs proved to be slow, requiring about 31 days to uncover the key, the JHU team put together 16 FPGAs in parallel and were able to uncover the key in about 35 minutes. Real-world applications were shown by using the evaluation kit in a briefcase and getting close enough to a person to retrieve the response from a challenge, effectively making it possible to scan victims for the RFIDs. Additionally, the team built a transponder to circumvent an engine immobilizer and spoofed a Speedpass signal to purchase gasoline. To their surprise, there was little push-back from Ford, who made some phone calls but no legal threats, or TI, who did not want proprietary information published but did not threaten to sue.

It was noted during the Q&A that the cost of the FPGAs used in the attack have dropped to $150 each, making this even more economically feasible. Bono expanded on this by observing that the decoder chip itself cost a mere $12. Rik Farrow asked how much cryptanalysis was performed to uncover the algorithm, and Bono responded that because the key was so weak, no cryptanalysis was necessary at the time, although it was performed formally when the protocol was broken.

- ***Stronger Password Authentication Using Browser Extensions***

  Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C. Mitchell, Stanford University

  Collin Jackson presented the password "phishing" problem, where users cannot reliably identify fake sites set up for purposes of stealing credit card and other identity data. In particular, the problem of protecting passwords used in multiple venues was addressed. Some passwords are used for low-security sites, such as high school reunions, while others, oftentimes the same password, are used for sites requiring high security, such as banks, where revelation of the password has drastic consequences. If the same password is used at both types of sites, breaking a low-security site could reveal the password to a high-security site. Jackson and his group investigated ways, as transparent to the end user as possible, to ensure that high-security passwords were not revealed.

  The solution proposed, called PwdHash, is a lightweight browser extension. It generates a unique password that is a hash of the password employed and the domain name of the Web site visited. This provides a modicum of protection against phishing, as the HMAC will be different for the password given to a spoofed site compared to the real one, due to different domain names. While other password hashing schemes exist, Jackson asserted that PwdHash was the only one that remained invisible to the user. One particular problem not addressed by many solutions, however, is the spoofing problem, where a malicious site employs JavaScript or Flash in such a manner that the user thinks he is entering information into an encrypted password field, but the password is sent in the clear, circumventing the hashing mechanism. To handle this, the tool is set up so that the original password never touches the Web site itself, with keystrokes being intercepted by the browser extension and the hashed result sent to the site. A password prefix (in this case, "@@") is used to activate the browser extension. This is the best method for securing users, as they do not have to decide when to make a trust decision.

  Challenges in this scheme include password resets, use in Internet cafes, and dictionary attacks. Jackson clarified that this tool does not protect against spyware or DNS poisoning. To allow password resets, the user must enter the unhashed password into a change page. Use of the password prefix facilitates this, however, as the prefix ensures that old passwords will not be hashed and new ones automatically will be. Because users cannot install the software at Internet cafes, an interim solution set up by the authors is to create the hashes from a secure Web page (http://www.pwdhash.com). It was asserted that dictionary attacks work about 15% of the time, so if the password was retrieved from a low-security site and the attacker knew the domain name, their odds of retrieving the password are much lower than the 100% rate currently achievable. The ultimate

solution would be to use a better authentication protocol.

In the Q&A period, a question was raised about how to handle policy requirements for different sites (e.g., minimum number of password characters and use of numbers or caps). Jackson responded that the best way would be to create a policy repository for all sites. Another way is to look at the user password itself, but this gives up some security. A following question raised concerns about Javascript focus-stealing attacks, where a user could think they are using the extension but the keystrokes are being hijacked by a script. This is a difficult problem to solve, but, theoretically, one could find all ways in which focus-stealing may occur and eliminate them; using longer passwords is also beneficial. Another question had to do with user-interface issues. The group found that, above all, end users favored simplicity and ease of use over any other factor.

- ***Cryptographic Voting Protocols: A Systems Perspective***

Chris Karlof, Naveen Sastry, and David Wagner, University of California, Berkeley

An analysis of two new cryptographic voting schemes was presented by Chris Karlof. DRE (direct recording electronic) voting machines are popular for a variety of reasons, such as their ability to display multiple languages and allowance for disabled people to vote more easily, as well as providing quick counts. However, the software and hardware must be fully trusted and the process transparent, none of which is guaranteed by current DREs. Allowing a voter-verified audit trail (VVAT) can be done by issuing a paper receipt. Election officials can use these to verify recounts, but individual voters cannot verify their vote. David Chaum and Andrew Neff have each proposed verifiably cast-as-intended protocols, where the voter can later check that their vote was as they registered it. The ballot is encrypted but can be verified later on public bulletin boards by the voter. The analysis from Karlof's group focused on Neff's scheme, but is applicable to Chaum's as well. The DRE makes a pledge that the row chosen by the voter on the ballot (where a row consists of a certain pattern of 1's and 0's) is the one they chose, and later the voter can match the candidate openings with the pledge made. To circumvent vote-buying, all candidate rows on the ballot are opened, not just the one corresponding with the chosen selection.

Karlof explained that both protocols were subject to information leakage through subliminal channels. The DRE can embed information within the pledge values, constructing a ballot where a certain bit pattern indicates the user's choice. Someone knowing the encoding pattern could then look at a ballot and know who the voter selected, threatening privacy. An analysis of the attack found, in the worst case, it was possible to encode up to 51KB per ballot through a subliminal channel, enough to provide plentiful information on the voter. The only solution appears to be making the ballot preparation more deterministic. Another possible attack is to use humans as cryptographic agents. Humans are not generally good at detecting subtle deviations, and a DRE can produce a false ballot that looks essentially similar to what the voter would expect. Because the protocols specify that the user makes his pledge before the DRE offers a challenge, the DRE is susceptible to cheating, as it can offer a receipt with small differences that the user will ignore. There is no clear mitigation strategy other than user education and testing during elections.

Finally, these schemes can only detect DoS attacks, not mitigate them, though that is still better than what DREs are capable of doing today. A simple attack from which recovery is impossible is to plant a trojan horse in every DRE, such that nationwide, the machines selectively delete ballots and perform ballot stuffing. Alternately, a machine can deny service selectively, such as only when a chosen candidate is losing. Such activities would be enough to cast entire elections in doubt, representing a threat to the entire voting system. Flexible recovery strategies including the use of VVATs are required. In summary, while the protocols examined are a large improvement over current implementations in DREs, some issues remain to be ironed out.

Invited Talk

- ***Human-Computer Interaction Opportunities for Improving Security***

Ben Schneiderman, University of Maryland

Summarized by Ming Chow

Professor Ben Schneiderman first reminded the audience that the goals of user interface design are to be cognitively comprehensible and to be effectively acceptable, not to be adaptive, autonomous, or anthropomorphic. The scientific approach to designing user interfaces includes specifying users and tasks, accommodating individual differences, and predicting and measuring learning, performance, errors, and human retention.

Professor Schneiderman stressed the importance of usability in controlling security and privacy, as put forth by the Computing Research Association (CRA) and the 2005 President's Information Technology Advisory Committee (PITAC) Report. One of the grand challenges established by the CRA in 2003 was "to give endusers security they can understand and privacy they can control," and usability is increasingly important in areas such as patient health records, law

enforcement databases, and financial management. The 2005 PITAC report noted similar challenges for end users and operators. Professor Schneiderman listed five goals of security and privacy: availability, confidentiality, data integrity, control, and auditability.

Professor Schneiderman presented the security and privacy settings interface in Microsoft Internet Explorer, which, he noted is riddled with usability problems, from the tedious online help to the challenge of setting up a Virtual Private Network (VPN). He also mentioned the emerging research in the area of usability and security/privacy.

Professor Schneiderman offered several valuable strategies for improving the usability of security/privacy: use a multi-layer interface that ties complexity to control and that also permits evolutionary learning; use a cleaner cognitive model that has fewer objects and actions; show the consequences of decisions; and show activity dynamics with a viewable log. He urged improving commercial practices by putting more emphasis on usability engineering and testing, which will lead to improved product quality, reduced costs, improved organizational reputation, and higher morale. Using his suggestions and insights, he presented a sample design of File-sharing Onweb with Realistic Tailorable Security (FORTS), which uses the multi-layer interface approach.

Finally, Professor Schneiderman presented information visualization for security and repeated the mantra of information visualization: overview, zoom-and-filter, details-on-demand. Human perceptual skills are remarkable, and human storage is fast and vast. He suggested using information visualization as a valuable opportunity for security/privacy: for linking relationships, profiling users and traffic, and understanding hostile events. A number of commercial and academic visualization tools were demonstrated, including SpotFire, a rich and powerful commercial visualization package.

Panel

- ***National ID Cards***

  Niels Provos (moderator), Google; Drew Dean, SRI International; Carl Ellison, Microsoft; Daniel Weitzner, World Wide Web Consortium

  Summarized by Serge Egelman

  With the passing into law of the REAL ID Act (P.L. 109-13), many Americans have started to become aware of the concerns that come with a national identity system. It was only fitting that this year's USENIX Security Symposium featured a panel to discuss such concerns. In his opening remarks, moderator Niels Provos pointed out that most European countries already have had national identity cards for quite some time. He has had his card for his entire life and he uses it regularly for such activities as traversing borders and voting without any hassles. He quite likes his national identity card, in fact. But Germany has strong laws regulating the collection and sharing of personal data. The United States has no such laws, and that is why there is a legitimate concern regarding what a national identity system will do to personal privacy in this country.

  Carl Ellison, an expert on authentication and authorization systems who currently holds the title of Security Architect at Microsoft, laid out the arguments for and against national identity cards. He went on to say that both sides are wrong; the opponents are wrong, in that the defeat of such a system will not in fact end data privacy problems, and the proponents are wrong, because they do not understand that a national identity card will not achieve the security goals for which it was intended (i.e., the card will never be a "not a terrorist" card). To elucidate these arguments, Ellison went over the process of making a security decision: a channel is opened, an identifier is offered, and authentication occurs. Authentication involves proving that the client has a right to the given identifier and is authorized to access the requested resource. Thus, such a security decision cannot simply be based on a name or identifier; it must also involve determining whether the person has appropriate permission. This problem can clearly be seen with the proposed national identity system in this country: it is aiming to prevent terrorism, but only knowing a name says very little about whether someone is a terrorist and what their intentions may be.

  Ellison then brought up the example of Walton's Mountain. It is a fictional place where all of the residents are born and eventually die; everyone knows each other. Thus, when a security decision needs to be made, any resident just needs a name and can then recall memories about the person. National identity cards are trying to accomplish the same thing through what Ellison calls "faith-based security." Through the use of biometrics and identity documents, the government is trying to make assurances about names so that they can recall "memories" about a person from a nationwide database. Unfortunately, such a database does not exist, and even if it did, we would not know anything about a person we had never interacted with before. This is not a proper security decision; we are doing authentication but not authorization. Urbanization made this a very difficult task, and the Internet has made it impossible.

  Drew Dean's interest in the issue of national identity cards can be seen by his involvement in two separate National Research Council studies on authentication and national identity systems. He mentioned that in getting to the conference he

had to show two different forms of identification: a passport to get on the airplane and a state driver's license to rent a car. In this country, a state driver's license is recognized by every state (although there is no federal law mandating this, every state has passed its own law to recognize out-of-state licenses for the purpose of comity). However, outside of the U.S., it varies. One of the NRC studies that he referred to brought up the fact that a national identity system needs to cover more than just U.S. citizens. This and other problems are often failures of the system, not just the card. But before such a system can be fixed (or properly implemented), a few questions need to be answered: What will the purpose be? Who will be enrolled? What information is stored? Who has access to the information? What are the implications with regard to identity theft? While it is clear that existing credentials are very weak, it is even clearer that a single nationwide system would create a single point of failure.

Daniel Weitzner has also been involved with National Research Council studies on national identity systems. He started by mentioning that the Washington, D.C., sniper and the 9/11 hijackers have been the biggest motivators for creating a national identity system. It was largely the terrorist hijackings that motivated the passage of the REAL ID Act, which mandates states to create uniform identity cards within the next three years. The law defines what is to be included on the cards and what is to be stored in the national database, but it makes no mention of how the data can be accessed or used, and by whom. It is also unclear if it will solve the problems that it intends to.

Regarding the sniper case, the license plate number was recorded at least ten times near the sites of the crimes, but the car wasn't associated with the crime. As Weitzner put it, they were "looking for a white truck with white people instead of a blue car with black people." Had each license-spotting been stored in a database which was shared by all of the police forces, they could have correlated the fact that this car was spotted at the scene of many of the shootings. But at the same time, this challenges our current privacy model. Many intrusive practices occur from drawing inferences, rather than from data collection alone. Credit card transactions lead to profiling, Web logs lead to user patterns, and location-based systems lead to discovering travel patterns. What we need right now from a technical standpoint is enforcement of rules, as well as secure audit systems. From a policy standpoint we need to shift from limits on data collection to limits on data usage, where we can require accountability and auditing. The current threats to privacy are not coming from the information itself, but from the inferences. Thus, by increasing exposure to the personal information collected, we can actually advance personal privacy.

The question on everyone's mind for the panel was whether there would be a benefit to being a national identity cardholder. While they differed in their reasoning, all of the panel members agreed that the costs would greatly outweigh the benefits. Carl Ellison referred to Walton's Mountain again, reminding everyone that implementing authorization on the cheap is still an unsolved problem. Issuing cards in no way achieves authorization. Daniel Weitzner drove home that point, saying that when confronted with a new technology that they do not understand, government treats it as a panacea. Such systems are expensive to implement and do not provide the solution that their proponents claim. Drew Dean mentioned that one of the biggest privacy concerns is with regard to secondary uses of personal information. Originally, social security numbers were to be only used by the Social Security Agency, just as a driver's license was originally meant to be a license to drive. But since these systems exist, private industries have used them for other uses rather than spending money to create their own systems. All of these systems undergo function creep, and privacy concerns abound.

Invited Talk

- ***Homeland Security: Networking, Security, and Policy***

  Douglas Maughan, DHS, HSARPA
  Summarized by Ming Chow

  Douglas Maughan, program manager at the Department of Homeland Security Science and Technology Directorate, discussed some of the issues and tools the department is currently working on. Maughan provided an overview of the organization of the DHS, and discussed its research and deveopment priorities. He also explained the differences between research and development funding at DARPA and at the DHS: at the DHS, 85–90% of funds are tied to requirements, and 10–15% of funds are dedicated to research. The five priorities of cybersecurity in the department are testing and evaluating threats, critical infrastructure, customer service, coordinating research among agencies, and creating partnerships. Maughan engaged the audience in discussion about two policy issues: DNS, and securing protocols for the routing infrastructure. He acknowledged that people are unhappy with ICANN's model of managing DNS, which is a key part of the global Internet, and asked the audience several questions, including: What incentives should be put in place for industries to use DNSSec? Should the rootkey be managed using threshold cryptography or a single rootkey? Unlike DNS, there is no governance for the routing infrastructure. Maughan

acknowledged that ISPs are doing the bare minimum to protect networks, and he asked the audience what incentives should be provided to industries to encourage their adoption of a standard and development of solutions for deployment.

Next, Maughan presented two DHS projects, DETER and PREDICT. DETER is a shared testbed infrastructure for medium-scale security research, including repeatable experiments, especially for experiments that may involve "risky" code. The Protected Repository for Defense of Infrastructure against Cyber Threats (PREDICT) is a repository of defense infrastructure data, where the aim is to have private corporations donate real incident data for security researchers and academia to use. The goal of these projects is to provide an experimental infrastructure to aid development of a large-scale deployment security technology sufficient to protect our vital infrastructures. These projects are not without controversy. Maughan asked the audience to consider a number of other questions, including: What industries should be involved with DETER, and how? What is the level of anonymization of the data? What should be the level of institutional sponsorship of PREDICT, and what happens if one violates the terms of agreement?

Refereed Papers

DIAGNOSING THE NET

Summarized by Mohan Rajagopalan

- ***Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network***

  Ju Wang, Xin Liu, and Andrew A. Chien, University of California, San Diego

  Denial of service (DoS) attacks are a key problem as Internet service applications become an important part of the enterprise. This work focused on infrastructure-level DoS attacks and was based on two key ideas: enforced mediation, and the notion of distributed front ends. Since theoretical models cannot capture the dynamics of network and application behavior as observed in large networks, the authors' work addressed these challenges and performed a realistic study by using a large-scale packet-level online simulator, MicroGrid, that was better than NS2 and PlanetLab.

  The experiments produced three results: first, they showed that this approach performed better in terms of baseline performance. Second, the proxy network was effective against both "spread attacks" and "concentrated attacks." Finally, the results showed that their system was scalable.

  The first questioner asked Ju to compare their MicroGrid-based approach to a simpler one based on NS2. Ju replied that scale is important for realism and NS2 could not provide a realistic approximation. He referred to the paper for further details on what realism meant. When asked to comment on the switch over time he replied that while they did not consider it, it was something that would be seen in a real system.

- ***Robust TCP Stream Reassembly in the Presence of Adversaries***

  Sarang Dharmapurikar, Washington University; Vern Paxson, International Computer Science Institute, Berkeley

  Sarang Dharmapurikar described the growing interest in higher-level packet processing. The motivating question for this work was whether it's possible to reassemble packets at high speed. Previously, systems either did not have a buffer and so would drop packets (TCP instability) or would guess the amount of buffer required. The primary contribution of this work was to analyze TCP traces in order to measure buffer requirements that could then be used to improve the system. The objective was to optimize for the average case by introducing an inline hardware device that could kill connections and allow normalization while preserving TCP dynamics.

  This work presented three fundamental measurements: first, up to 15% of the connections may have had out-of-order packets; second, the maximum buffer required is small; and, finally, 60% of the holes lasted for less than 1ms. This indicated that reordering and not dropping was the right strategy. In order to deal with adversarial connections they proposed a policy-based defense; to prevent the attacker from filling the buffer with a single connection, they would restrict the policy of each connection to a preset threshold. Their policy would prevent multiple connections from a single host in order to prevent the adversary from creating multiple connections. The final policy evicted a page randomly and killed a connection in case of an overflow. The talk mentioned zombie equations that would be used to improve connection eviction packets. In conclusion, this work presented the facts that TCP reassembly would be important for security and that trace-driven analysis can be used to design and tune the system.

  The first question dealt with an adversary who would send a bunch of holes and then a bunch of small packets to fill the holes, thus flooding the analyzer. Sarang replied that this could be treated as an anomaly. The second question concerned the use of multi-path for group resiliency. The response was it would be difficult to handle.

- ***Countering Targeted File Attacks Using LocationGuard***

  Mudhakar Srivatsa and Ling Liu, Georgia Institute of Technology

  Mudhakar Srivatsa presented LocationGuard, which provides location hiding to protect against DoS and host-compromised attacks. There are two major problems this work tries to address: access control in a

wide area file-storage system, and defending against targeted attacks. The authors' approach tries to hide files, locate them for known users, and prevent inference attacks. A location key is used to hide the location of the file (A:(file,loc_key) -> location). The implementation was based on files stored in a distributed hash table.

Their approach uses a probabilistic look-up scheme which builds on a "safe obfuscation" algorithm for secure routing by never disclosing the file ID. In order to prevent inference attacks that are based on observing file accesses and frequency, files are divided into chunks. Periodically, the location key is changed, and this rekeying nullifies all past file inferences. The actual implementation is based on Chord using AspectJ. The authors found that their approach effectively defended against DoS, DDoS, and host compromise attacks and incurred minimal overheads.

Invited Talk

- ***Electronic Voting in the United States: An Update***

Avi Rubin, Johns Hopkins University

Summarized by Ming Chow and Jonathon Duerig

Avi Rubin began by discussing his recent experiences at an annual conference of state chief justices held in South Carolina, where he served on a panel about electronic voting. Surprisingly, most of the chief justices were not aware of the electronic voting problem, and most do not even buy into the idea of trojan horses. However, Rubin's talk pointed out some of the problems that result when voting technology loses transparency. It is important to educate the chief justices in this area, since they will increasingly be the arbiters of who wins elections, as was seen in a recent election in Washington state. Rubin noted that it was difficult to explain the technical issues of electronic voting to a mostly nontechnical group at the conference. Several chief justices (of Pennsylvania, Washington, Puerto Rico, and Florida) praised Rubin's talk for making them believers regarding the electronic voting problem and for stressing the importance of a paper trail.

Rubin reviewed the background of the electronic voting problem. Shortly after the debacle of the 2000 presidential election, Congress passed the Help America Vote Act (HAVA). The purpose of the act was to establish a program to provide funds to states to replace the punchcard voting program. In 2003, $1.4 billion was given to states to buy electronic voting systems. Members of Congress approved of the idea of electronic voting and didn't find any problems with systems, rebuking Rubin. However, before the 2004 presidential election, the controversy surrounding electronic voting escalated. Rubin noted numerous problems, including weak requirements from independent testing authorities (ITAs), no source code review of systems, controversies over the lack of a paper trail, lack of accommodation for blind people, and the fact that some people do not even look at their receipts.

Rubin noted that there is still a disconnect between Congress and the computer science community and that the HAVA money is almost gone: $4 billion has been spent. Maryland commissioned several studies to figure out how to retrofit new voting safeguards onto the old technology. The finding is that things are being done wrong, but there is no money to fix them. Rubin recalled a trip to the Carter Center in Atlanta, where he found that the people are very concerned about the fact that there is no way to observe electronic voting. In Oregon, everyone votes by mail; there, voter coercion and resale are problems. Except for Baltimore, Maryland is still using the highly controversial Diebold electronic voting machines. In New Jersey, legal battles over voting continue to rage. Politicians in Washington do not seem worried about these problems. People in positions of power are invested in voting-machine companies. Although progress is being made in confronting the problems in existing voting technology, the overall picture is mixed. And the difficulties in disseminating information on the problem of electronic voting means that many people in this country still do not believe there even is a problem.

Refereed Papers

MANAGING SECURE NETWORKS

Summarized by Stefan Kelm

- ***An Architecture for Generating Semantics-Aware Signatures***

Vinod Yegneswaran, Jonathon T. Giffin, Paul Barford, and Somesh Jha, University of Wisconsin, Madison

In this talk Jonathon described both the architecture and the implementation of Nemean, a system for automatic IDS signature generation. One of the objectives of Nemean is to take the human out of the signature-generation loop in order to reduce errors (both false positives and false negatives). He said that current solutions do not make use of application-level protocol semantics, whereas Nemean operates on the application layer, working with what he called semantics-aware signatures. In doing so, it is able to aggregate TCP flows, generate signatures for attacks where the exploit is only a small part of the payload, and produce generalized signatures. And it is easy to understand and, importantly, to validate.

Nemean's architecture consists of data collection, flow aggregation, service normalization, and clustering. The data collection component takes its input from a honeynet; the current implementation captures

HTTP and NetBIOS. The main part of the flow aggregation component is to manually assign weights to single data packets, which are subsequently used for automatic signature generation. Service normalizations take care of possible problems within the data flow. Finally, the clustering component is divided into session clustering and connection clustering.

Jonathon then presented some very impressive results of an experiment that ran over two days: they trained Nemean using captured honeynet data and achieved a detection effectiveness of about 99%, with 0 false alarms. Their research suggested that, depending on the attack, connection-level clustering makes sense at times and session-level clustering seems appropriate at others. For more information, see http://www.cs.wisc.edu/~giffin/.

- ***MulVAL: A Logic-Based Network Security Analyzer***

  Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel, Princeton University

  Xinming Ou presented MulVAL, a new approach to network security analysis. The motivation behind this approach is to find possible security weaknesses in software and/or network configurations before running a particular service. An administrator, Xinming argued, should be able to put questions to a so-called "reasoning engine"—for example, is there an attack path that could lead to exposure of confidential data?

  Input from sources such as CVE is converted into input which may subsequently be used through logic programming. The authors chose MulVAL, which is a subset of Prolog. Xinming gave two examples: network and machine configurations are being expressed as datalog tuples—"serviceRunning(webserver, httpd, tcp, 80, apache)"—whereas the reasoning logic is being specified as datalog rules—"networkAccess(Attacker, Host2, Protocol, Port . . .)". Standard prolog engines then conduct the analysis of configurations.

  The basic idea behind the architecture is to have a small scanner running on each host within a network and an analyzer which looks for new information sent by the scanners. Xinming described various reasoning rules such as possible exploitation of known vulnerabilities, OS semantics, and attack techniques. He then presented some real-world results of MulVAL and argued that their system scales pretty well, mainly because of Prolog's system optimization. They used MulVAL to check their department's network configuration and immediately found a potential two-stage attack path due to multiple vulnerabilities that existed on a single server.

  Xinming said that future work involves testing the system on more networks and that reasoning rules for Windows systems are needed, too. He concluded that logic programming is a good approach to network security analysis.

  For more information, go to http://www.cs.princeton.edu/~xou/.

- ***Detecting Targeted Attacks Using Shadow Honeypots***

  K.G. Anagnostakis, University of Pennsylvania; S. Sidiroglou, and A.D. Keromytis, Columbia University; P. Akritidis, K. Xinidis, and E. Markatos, Institute of Computer Science–FORTH

  Stelios Sidiroglou presented Shadow Honeypots, a security architecture combining rule-based intrusion detection systems (such as snort) which are good at detecting known attacks with honeypots and other anomaly detection systems which are good at detecting zero-day attacks. By taking "the best of both worlds" one should be able to minimize both false positives and false negatives.

  Unlike the traditional approach, shadow honeypots allow for two modes of operation: client-side and server-side. The basic idea is to have a filtering component as well as anomaly detection sensors. Sitting behind those sensors is the shadow honeypot, which is an instance of the system or software to be protected. It is basically a modified version of the software itself, with various hooks introduced throughout the source code.

  The prototype implementation presented by Stelios introduces a few new system calls such as transaction() and shadow_enable(): if the shadow honeypot classifies input as malicious, the corresponding packets are discarded; if the packets are regarded as okay, they will be handled correctly and transparently by the system.

  Stelios presented two widely used prototype implementations modified by those shadow honeypot system calls: the Apache Web server and the Firefox browser. In this implementation they focused on memory violations such as buffer overflows. And although benchmarking the modified versions showed an overhead of 20% and 35%, respectively, Stelios said that the ability to significantly reduce the rate of false positives is a good reason to improve shadow honeypots.

  For more information, see http://www1.cs.columbia.edu/~ss1759/.

Invited Talk

- ***Cybersecurity: Opportunity and Challenges***

  Pradeep K. Khosla, CyLab, Carnegie Mellon University

  Summarized by Boniface Hicks, OSB

  Pradeep Khosla discussed various elements of CMU's CyLab (http://www.cylab.cmu.edu/), of which he is the director. CyLab not only studies the technological aspects of computer security, but also integrates efforts with the Tepper School of Business and the Heinz School of Public Policy. It

extends internationally and includes the efforts of 150 security professionals and more than 50 industrial affiliate member companies. It is an ambitious and wide-reaching research center, embracing both short- and long-term projects.

Khosla himself is helping to build survivable storage systems. In hopes of making storage perpetually available, even in the face of failure or compromise of some disk arrays, the team, led by Greg Ganger, is using redundancy in a novel way. A naive approach would be merely to break up a file into a thousand pieces, like a jigsaw puzzle, and store the pieces on different disk arrays. In this way, if one piece were compromised, no information would be gained. An improvement is to duplicate the storage and break it up into four 1000-piece puzzles. In this way, even failure of a disk will cause minimal damage, and the degradation will be graceful over the failure of multiple disks. Furthermore, their system is self-healing, recognizing what has been lost and recovering it by using redundant information. In this way, they have been able to build a robust system using only non-robust components. As expected, however, increased safety is paid for with slower access rates.

Another CyLab project is the Grey System. Khosla showed a demo of this system, which is already being deployed in the computer science buildings at CMU. A person can get into his own office using a cell phone with Bluetooth. Furthermore, a person can remotely give authority for someone else to enter his office over the cell phone. The system allows for one cell phone to provide a certificate to another cell phone, which can then use the certificate to authenticate with the door. The logic for this delegation system is handled using automated theorem-proving software developed by Pfenning and Lee some 10 years ago. This novel application is one they never expected; it demonstrates how pure research produces unexpected results, even a decade after it has been developed. Khosla used this opportunity to petition for government agencies to be willing to provide funds for the sake of long-term results.

Using the Grey System, what prevents someone from stealing a cell phone and breaking into that person's office? Ideally, the cell phone would authenticate its user—using biometrics, for example. Khosla recognized that no biometric is perfect, but perhaps a combination of face and fingerprint, voice and iris recognition would make a robust system. One group in the CyLab has been making great progress in face recognition. Although there are an impossible number of variables (pose, illumination, expression, occlusion, time lapse, etc.), the lab has made significant progress in gaining excellent accuracy with the help of very few training images. Their software has produced far better results than current commercial software. There is still the challenge, however, of incorporating this resource-rich technology into resource-constrained devices such as cell phones or PDAs. Also, there is need for better user input for these devices, such as voice recognition. Furthermore, as this technology becomes more advanced and is more broadly trusted (biometrics will be required on passports by the year 2010), there are various business and policy issues which must be explored. It may be desirable to encrypt the biometrics on a passport, for example.

The last significant area covered by Khosla was education. Using examples from his own experiences with his son, he described the need for children to be made "cyberaware." Since it is so easy for a teenager to get a malicious script from the Internet and cause great damage, it is important to educate children in ethics and norms for Internet use. CyLab has taken on this social responsibility by forming a program that seeks to educate 20,000 young people in the Pittsburgh area, with the hopes of educating 10 million in the future. They're trying to reach kids aged 5 to 10 by incorporating ethics into an interactive game, which is available at http://mysecurecyberspace.com. In this game, the player interacts with characters such as Elvirus and MC Spammer. A study of the 20,000 children who will be required to play this game is being conducted scientifically, with a long-term evaluation of the effectiveness of this approach.

Throughout his presentation, Khosla made some observations about open areas and the waves of the future. He claimed that Human-Computer Interaction (HCI) is now the hot field in computer science. He identified the emerging field of resource-constrained devices such as mobile phones and even RFIDs, and believes they will be ubiquitous in the near future. Mobile access is the new wave, he said; it holds the promise of providing telephony in developing nations—77% of the world is already within range of a mobile network. At the same time, privacy, security, and capture resilience are needed for mobile technologies. Finally, there were comments about the reduction in funding for these projects. Khosla reiterated how important it is that there be ongoing funding for security—it is a problem that will never simply be solved. He also challenged DARPA not to require so many projects to be classified, since that leads to duplication of effort. Finally, there was an audience comment encouraging incentives to get kids involved in bug reporting as well as in reporting malicious activity. Khosla welcomed this idea.

Panel

- ***Sniffing Conference Networks: Is It Legal? Is It Right?***

  Abe Singer, San Diego Supercomputer Center; Bill Cheswick, Lumeta Corp.; Paul Ohm, U.S. Department of Justice; Michael Scher, Nexum, Inc.

  Summarized by Serge Egelman

  At many security conferences, intercepting wireless network traffic has become commonplace. DefCon is at the extreme—passwords are annually written down and taped to a "wall of shame." But USENIX Security has not been very different. Often the motivation calimed is that of educating users about poor security habits, but one thing is fairly certain: this behavior is illegal. This panel examined both the ethical and the legal impact of sniffing wireless conference networks.

  Bill Cheswick, currently the chief scientist at Lumeta Corporation, is well known for his 1991 paper "An Evening with Berferd," in which he lures a hacker to a machine that is being monitored. For months Cheswick watched as this cracker would attack other machines from the honeypot he had set up. At one point during this study Army Intelligence came to Cheswick and read him his Miranda rights; they saw attacks coming from his machine and assumed that he had something to do with it. He was let off the hook after explaining the project. While Cheswick learned many of this particular cracker's techniques, he had received little ethical guidance on how to proceed. Was what he was doing illegal? Was it ethical? After all, it was his own machine and this cracker had accessed it without authority (which certainly is illegal). While this example falls into a gray area, Cheswick mentioned how he used to sniff conference networks for plaintext passwords so that he could educate people about insecure protocols. Upon finding out that this activity was illegal, he has restricted his sniffing to networks that he owns.

  Paul Ohm, an attorney for the United States Department of Justice, gave an overview and history of the various federal computer crime laws. Starting in 1968, Congress passed regulations about eavesdropping after being outraged by the egregious activities of the FBI in monitoring citizens without any legal oversight. This law made it illegal both to tap phone lines without a warrant and to bug. This is commonly referred to as the Wiretap Act (Title III of the Omnibus Crime Control and Safe Street Act of 1968). In 1986 Congress passed the Electronic Communications Privacy Act (ECPA, 18 U.S.C. §§ 2510-2521). With a few exceptions, this law made it illegal to intercept electronic communications. Monitoring one's own network to protect rights and property is permissible, as is monitoring a network with consent from the users. Ohm pointed out that some might argue that by broadcasting passwords through the air in plaintext, the user is essentially "asking for it." Although it is entirely possible that this might eventually win in court, such a victory would be at the cost of thousands of dollars in legal fees for the defendant. At the same time, the likelihood of someone being arrested at a conference for sniffing traffic is very small. Ohm explained that when deciding to prosecute such a case, intent is crucial. But sniffing conference traffic also raises many ethical questions: even if it were legal, would this behavior be acceptable for someone not in attendance at the conference? What is the difference between a conference attendee sniffing traffic and an FBI agent sniffing traffic? The laws are in place to protect everyone equally.

  Abe Singer of the San Diego Supercomputer Center chose to concentrate on the ethical questions. Some of the common justifications range from "It's not a wiretap if there's no wire" to "I'm protecting the network." Of course this begs the question of what exactly is being protected by acquiring someone else's passwords. Another justification, "The user deserved it for using plaintext passwords," is similar to "She deserved it for walking down a dark alley alone." This sort of behavior embarrasses those who are subjected to it. These are often new users who do not know any better. Instead of alienating them, our time would be better spent educating them. Of course, one way around this would be to force all conference attendees to sign waivers of consent. Just imagine an ISP requiring this of all its customers.

  Mike Scher, general counsel and compliance architect for Nexum, Inc., chose to focus on enforcing normative behavior. Before a law is passed, there is always some consensus that the law serves to prohibit behavior in violation of ethical norms. We will often tell colleagues when they are behaving improperly. But in this community, sniffing is an ethical gray area, and it is therefore very difficult to become watchdogs. On the one hand, there are security luminaries who are using plaintext passwords, and on the other, there are other security luminaries who are sniffing. As a community, we need to reach a consensus as to whether this is unethical.

Invited Talk

- ***Treacherous or Trusted Computing: Black Helicopters, an Increase in Assurance, or Both?***

  William Arbaugh, University of Maryland

  Summarized by Kevin Butler

  The debate about trusted computing is passionate and pointed; as Arbaugh states, it is good when people debate issues, but bad when people make unsubstantiated

claims. Arbaugh presented an overview of trusted computing and spoke of the positive effects and possible negative ramifications. Much of the debate centers on who controls one's computing and one's information. There is a tension between owners and users of information. Owners want to control information (e.g., patient data) and while this seems laudable, there are scenarios where data leakage is important, such as with whistleblowers in a company (e.g., tobacco companies wanting to keep their documents secret). Trusted computing is inherently a "dual-use" technology, which can used for good purposes or ill. A user's expectations for what trusted computing might be will differ in many ways from what a larger company's expectations will be. An object is trusted and trustworthy if and only if it operates, and can be expected to operate, as expected. Therefore, one definition is that trusted computing is when your computer operates as expected. Note that the expectations themselves are not included in this definition.

What is a trusted computing base (TCB)? It's the totality of components responsible for enforcing security policy, including hardware, firmware, and software. A key component of a TCB, the reference monitor, mediates all access to objects from subjects. The implementation of a reference monitor is known as a reference validation mechanism (RVM); it should be tamper-proof and unable to be bypassed, but small enough to be well analyzed and tested. The reference monitor acts as a base case; i.e., if the base case fails, the proof falls apart. These concepts and others were codified in 1983 in the "Orange Book," which provided good definitions and theory but was unwieldy in practice. Trusted computing was not seriously considered again until 2002, when the Trusted Computing Group (TCPA/TCG) went public. There has been a flurry of recent activity: next year may bring virtualization software from Intel, secure execution mode from AMD, and other efforts.

The TCG features as its core element the Trusted Platform Module (TPM), a passive device that only does something if commanded over the system bus. This means it can't perform actions such as raining and interrupt to stop processing, can't take over a machine, and can't delete files. It's essentially a smartcard soldered to the computer, so it has lots of interesting crypto functions implemented in hardware, including random number generation and symmetric and asymmetric encryption. Storage is protected through on- and off-device shielded locations, and protected execution provides an environment for protected crypto functions to execute without modifications or exposure of key information. A key function of the TPM is attestation, in which the current status of both the TPM and the machine on which it resides is attested to by the TPM. Platform configuration registers (PCRs) are held in volatile storage in the TPM, and can be initialized to zero but not directly written to. The other operation permissible is extension, in which an extended value is hashed with the old value of the PCR to create a new value.

Arbaugh suggested that trusted computing can be broken into two phases: getting started (the pre-boot phase) and the operational, or post-boot, phase, where the system must remain trustworthy. Authenticated boot can be performed by the TPM; it ensures that at boot time the system is in a secure initial state, assuming that the measured software is trustworthy. This latter concept is problematic, as nobody to this point is capable of making such a guarantee. Authenticated boot is a passive method; if the bootstrap process detects malicious activity, it cannot stop the system from booting, and it might not even be able to detect if there is malice. Briefly, the operation breaks bootstrapping into several steps, where a hash is taken at each step and the PCR extended. Integrity measures are stored in a write-once register, so the hashes can be securely compared. While it can be proven to another authority, there is no way to prove to the user that they are in a trusted configuration, due to the lack of a trusted path between the hardware and the user (e.g., an OS can spoof values as displayed to the monitor). Secure boot, by contrast, is an active process that can prevent malice from executing. It proceeds similarly to authenticated boot, but proves that it is in the correct configuration existentially, as execution is halted if the hashes do not match. However, it cannot prove a trusted configuration to a third party. Arbaugh suggested that what is needed is a trusted boot, combining authenticated and secure boot. There are times when being able to provide the system configuration to a third party is helpful, though this is open to abuse. However, malice should never be executed if it can be detected, no matter how good the protection is. The addition of a trusted path to the user is the only way to implement this.

Post-boot methods include IBM's extension of the TCG into runtime operation and software to use the TCG past boot virtualization, such as Vanderpool and Pacifica. In IBM's work, presented at the 2004 USENIX Security Symposium, all objects are measured and a list is maintained in kernel data, with measured values going into a PCR. This only works if all software is trustworthy, meaning that much more software than just the BIOS and boot routine must be verified. Virtualization modifications are proposed by Intel and AMD; however, previous work showed that some instructions in the x86 instruction set cannot be virtualized without breaking the virtualization itself. Domain managers such as VMware and Xen act like reference monitors, where each OS

runs in a partition, firewalled from each other. Multi-level security could be implemented effectively through this scheme, but there is still a problem of moving information between partitions, and particularly of covert channels between the virtualized OSes. The Vanderpool specification includes the highly problematic virtualization of I/O. Lagrande includes processor and I/O modifications to increase security and has trusted I/O paths to the video and keyboard plus protected execution and additional memory protection.

The main thesis of the talk was that trusted computing can be used in good and bad ways, and Arbaugh considered examples of each. Electronic voting is a particularly good application, as attestations with a trusted boot are what one wants from a voting machine. However, digital rights management (DRM) restrictions can be brought into place, thanks to the configuration attestations. The ability to lock files and protect crypto keys with the TPM prevents key escrow, and the police cannot access your keys. However, files can be locked to applications to limit competition. Strong authentication can be provided to the platform, which can help parental controls, but could provide a loss of anonymity. The only way to get lawmakers to do the right thing is either through generous campaign donations or by explaining things without extremism in a way that they will understand. Arbaugh put forth the idea that, contrary to current claims, the TCG could be beneficial to GNU software: evaluation and certification on an approved platform might eliminate government resistance to its use.

Arbaugh made some predictions for trusted computing. Improvements will come from virtualization, but Lagrande will not survive, as the market will not understand the need for trusted paths, nor will it be willing to spend the money. The TCG will be hacked; looking at the XBox as an example shows that hardware hacking is just a different skill set from software, though some tools are more expensive. In conclusion, all technology is essentially dual use, and while laws and policies attempt to limit evil uses, they cannot be completely eliminated. One has to decide for oneself if the good provided by trusted computing outweighs the bad.

Refereed Papers

ATTACKS

Summarized by Mohan Rajagopalan

- ***Where's the FEEB?: The Effectiveness of Instruction Set Randomization***

  Ana Nora Sovarel, David Evans, and Nathanael Paul, University of Virginia

  The authors' objective in this paper, presented by Ana Nora Sovarel, was to evaluate whether an attacker could detect the randomization key remotely and then spread a worm on a network of instruction-set randomized machines. Their attack was based on exploiting incremental behavior by guessing instructions that corresponded to short control flow. They concentrated on a two-byte sequence that was used for a jump attack. A prime assumption in this work was that the same key would be used each time the application was randomized. Experiments were performed on Fedora without Address Space Layout Randomization. In particular the experiments evaluated whether it would be practical to spread a worm in such a deployment.

  Comments generally targeted the assumption that the same randomization key would be used each time. It was pointed out that ISR schemes re-randomize on each fork operation, and re-randomization is performed at load time, so the underlying assumption was incorrect.

- ***Automating Mimicry Attacks Using Static Binary Analysis***

  Christopher Kruegel and Engin Kirda, Technical University Vienna; Darren Mutz, William Robertson, and Giovanni Vigna, University of California, Santa Barbara

  This paper, presented by Chris Kruegel, discussed automating control flow attacks by analyzing applications to identify locations that an attacker could exploit.

  In particular, the authors hoped to defeat host-based intrusion detection systems through mimicry attacks, such as hijacking PLT entries. The goal was to set up an environment in which the attacker could regain control after executing the first system call. Symbolic execution was used to perform static analysis. They identified several instances where the attack would succeed on real programs.

  Someone asked whether this technique would work for non–buffer-overflow attacks. Chris replied that all that matters is the ability to inject code.

- ***Non-Control-Data Attacks Are Realistic Threats***

  Shuo Chen, Prachi Gauriar, and Ravishankar K. Iyer, University of Illinois at Urbana-Champaign; Jun Xu and Emre C. Sezer, North Carolina State University

  Shuo Chen from UIUC presented the last paper of this session, which explored how data flow can be exploited in order to compromise systems.

  The premise of this work was that several types of data, such as configuration inputs and user inputs, are security-critical and can be used to drive exploits. While it has been known that such attacks exist, the extent to which they are applicable has not yet been assessed. The authors show that many non-control vulnerabilities exist and the extent of damage is comparable to traditional attacks. Their experi-

ments indicated that several real-world programs, such as FTP, SSH, and Web servers, were vulnerable to such attacks. They were evaluated along two dimensions: the type of security-critical data, and the specific memory vulnerability that can be used to access the data.

Several defenses to protect against control data tampering were presented, ranging from the enforcement of non-executable pages to using low-level hardware infrastructure to protect control data. In general, memory corruption attacks remain a difficult problem.

Invited Talk

How to Find Serious Bugs in Real Code

Dawson Engler, Stanford University
Summarized by Francis Hsu

Dawson Engler shared his experiences using two dynamic techniques, implementation-level model checking and execution-generated testing, to find as many serious bugs as possible in real code. His earlier experiences with static techniques proved effective at checking surface visible properties like proper locking semantics. Since no code needed to be run or even compiled in static checking and it scaled well, it worked well in finding thousands of errors in code. Dawson successfully commercialized these two years ago by founding Coverity, a self-funded company with over 70 customers. However, this talk was not about his static analysis successes. While his dynamic techniques required all the code to run and took hours to diagnose a single bug when found, they did address a failing of static techniques: checking properties implied by code.

Implementation-level model checking is a mutation of formal method techniques, adapted for real code. Model checking is like testing on steroids, where every possible action is done to every possible system state. Since model checking makes low-probability events as common as high-probability events by exhausting the state space, corner-case errors could be found quickly. Dawson had several years of mixed results, but finally had a breakthrough success in checking three heavily used Linux file systems. He ran the entire Linux kernel with a virtual formatted disk in the model checker, applied each possible operation to the file system with failures at any point, and checked for proper crash recovery. Although the file systems would normally recover correctly after a crash, Dawson discovered that they usually broke when crashes occurred during the crash recovery process. In the end, he found 32 errors, including 10 places where a poorly timed crash would result in complete data loss.

An attendee wanted to get a handle on how much human and computational time was needed to apply the model checking for bug finding. Dawson said he wouldn't be surprised if it took a couple of weeks up front, since it's hard to figure out correct behavior of the code and understand any discovered bugs. The computational time could be infinite for a run and would also require lots of memory for searching the large state space, but in his experience Dawson usually found useful results in seconds or minutes of a run. Not finding any results in that time would likely be caused by a problem in the testing and not because the code was bug-free.

Another person asked if Dawson had seen cases in his testing where the access to the disk was not trusted to write the data it was given, and if he had seen any differences between brands. Dawson responded that he had tested the file system on RAM disks for performance reasons, but it could have been done on physical disks. A third attendee asked if Dawson had mode-checked fsck. Dawson confirmed that he did perform an end-to-end check of all the components of the file system, including fsck.

In the second half of the talk, Dawson described his more recent work with execution-generated testing, or "how to make code blow itself up." Creating good test cases for system code is hard work. Manual construction of test cases is laborious, and automated random "fuzz" testing may not hit corner cases or errors that require structured inputs. Execution-generated testing solves these problems by running the code to generate its own input test cases. Starting with an initial value of anything for the input, the program execution generates constraints for the values at fork points in the code. The collection of these constraints can then be used to generate inputs which, in turn, are used to test the code. With this technique Dawson generated format strings to test printf and network input to test an MP3 server, and discovered bugs in both.

Dawson has made the slides of his talk available at http://www.stanford.edu/~engler/usenix-security05.pdf.

Refereed Papers

PROTECTING THE NETWORK

Summarized by Kevin Butler

Mapping Internet Sensors with Probe Response Attacks

John Bethencourt, Jason Franklin, and Mary Vernon, University of Wisconsin, Madison

- ***Awarded Best Paper!***

Internet sensor networks are collections of systems monitoring the Internet, producing statistics related to traffic patterns and anomalies. Examples include collaborative intrusion detection systems and worm monitoring centers. Network integrity is based on the assumption that the IP addresses of the systems serving as sensors are secret; otherwise the

integrity of the produced data is reduced. Attempts to maintain anonymity include hashing or eliminating sensitive report fields (e.g., the IP address where an attack arrived), prefix-preserving permutations, and bloom filters. However, John Bethencourt presented a new class of attacks discovered by his group, called probe response attacks, which are capable of compromising the anonymity and privacy of Internet sensors.

Using the SANS Internet Storm Center (ISC) as an example, Bethencourt showed that given an IP address, if a probe is sent to the address then one can wait for the sensor network to report activity; if it doesn't, the address is monitored. With the ISC, only one TCP packet is necessary to initiate a probe connection, as incomplete SYNs are monitored. It is possible to send packets to every potential address, though this is not possible in a serial manner, given that most participants make only hourly reports and there are 2.1 billion routable addresses. Checking in parallel, however, is feasible. Starting with the full list of addresses, the search space is divided into intervals. After sending a series of probes and waiting two hours, the reports can be checked for activity, and those reporting none are discarded. For the others, a divide-and-conquer strategy can be used to further subdivide the intervals and make probes until, ultimately, all monitored IP addresses are found. Simulation results show that an attacker using a T3 can complete the attack in five days. With this information, an attacker can avoid monitored addresses in malicious activities such as port scanning or propagating worms, avoiding detection. Sensors can also be flooded with errant data. While the ISC was primarily considered, similar attacks are possible against other sensor networks, such as Symantec's DeepSight site.

While hashing, encryption, and omitting certain report fields can make attacks more difficult, they are still possible. Private reports would be effective but would severely limit utility. Top lists could publish only the most significant events, providing some useful information but not a complete picture, allowing attackers to avoid detection by keeping activity below threshold levels. Puzzles, captchas, and random log sampling are other techniques to prevent information attacks. One question posed was whether sensing in the core would be more useful than at the edge. This is more difficult to implement, as was mentioned in other papers. Another questioner asked about biasing data, as clever attackers can attack sensors from a variety of locations. More investigation into these forms of attack is needed.

- ***Vulnerabilities of Passive Internet Threat Monitors***

  Yoichi Shinoda, Japan Advanced Institute of Science and Technology; Ko Ikai, National Police Agency of Japan; Motomu Itoh, Japan Computer Emergency Response Team Coordination Center (JPCERT/CC)

  Yoichi Shinoda described still other methods of finding vulnerabilities in threat-monitoring networks. Passive threat monitors were inspired by the successes of Internet telescopes; results have been published in graph and table form. Determining where sensors are can compromise the monitoring network's integrity and can be performed by looking for feedback to induced input. By propagating a number of UDP packets at four /24 address blocks, they graphed the monitoring system, showing a spike four hours afterward. By targeting a particular system and looking at information such as company white papers and handouts, the basic system properties can be determined. Combined with packet-marking algorithms, which can be customized to the type of feedback from the network, sensors can be found efficiently. This was backed up by case studies.

  Protecting the monitors is not easy. Methods include throttling information flow, providing less information, and, in particular, detecting marking activity, looking for statistical anomalies where flurries of similar messages are sent. While system protection methods have been proposed, their effectiveness and completeness have not yet been verified, and unknown attacks may yet exist. Information leaks can still occur even with protection, and continuous assessment is necessary to study attacks and protection methods.

  A question about correlating sensor information was posed during the Q&A session. If sensor output is normalized as a countermeasure based on sensors looking at different networks, could similar patterns still be observed? Shinoda responded that while this was explored in the paper, the problem is that different monitors have different sets of sensors providing different results, and knowing why different results are provided is still a work in progress.

- ***On the Effectiveness of Distributed Worm Monitoring***

  Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis, Johns Hopkins University

  To protect against threats, monitoring active networks, and the routable unused IP address space in particular, is attractive, since no legitimate traffic should occur in these areas. With a single monitor, backscatter patterns can be found if a DoS attack is initiated; it is also useful for worm detection. However, a single monitor view is too limited, as worm scans that hit other parts of the network will be missed. Moheeb Abu Rajab presented methods of monitoring for worms using multiple, distributed models, concentrating on the fact

that non-uniform distributions more accurately model the real world. For an extended worm propagation model, the model must incorporate population density distribution, especially non-uniform worm propagation.

Equations were derived for the number of infected hosts in a /16 subnet, with the total infection being the sum of infected hosts. Abu Rajab presented simulations that showed that while non-uniform scanning worms propagated slightly more slowly than uniform scanning worms over uniformly distributed hosts, they spread much faster when a real data set was used. Based on this, better worm detection can be implemented by concentrating on different evaluation metrics. System detection time—the time for the monitoring system to detect a new scanner with a particular level of confidence—is important. Deploying distributed monitors with smaller address blocks, giving a finer level of granularity, produced optimal response times. Even partial knowledge of population distribution was found to improve detection times by a factor of 30.

In the Q&A, an audience member asked whether the worm will take longer to propagate if it starts in very sparse populations under a skewed population distribution. Abu Rajab responded that because worms have a random component to their dissemination, even if some start in sparse areas, at some point they will target heavily populated subnets and propagate much faster from there onward. Another question concerned the speed of detection as a metric; has the communication overhead between probes been considered as a factor reducing the speed at which the worm can be detected? This is a good question, agreed Abu Rajab. The research to this point concentrated on evaluating space requirements and assumed that an infrastructure was in place; for distributed systems, an adaptive routing system that minimized overhead would have to be implemented.

Invited Talk

Open Problems with Certifying Compilation

Greg Morrisett, Harvard University
Summarized by Mohan Rajagopalan

Greg began the talk by stating that mobile code is not the basic security problem. The real difficulty lies in understanding the semantic properties of code rather than its syntactic properties. For example, even simple policies are undecidable. Proof-carrying code (PCC) is an approach where each program is accompanied by a proof. The advantage here is that functionality is moved from the trusted computing base to the proof checker. Certifying compilers are programs that systematically transform proofs along with source. The question now is how to derive initial proofs.

One approach is to use type systems in such a way that they map to policies. Citing Microsoft Research's Singularity project as an example, he mentioned that some high-level language-based approaches have suggested eliminating C altogether. Software fault isolation is another approach; it checks that all memory accesses are to valid locations within a program's address space. The idea here is to track mapping from source to target address. Control flow isolation was mentioned as an implementation for the x86 platform. This approach meant that policies were relatively simple and easy to enforce—for example, by rewriting the binary.

The remainder of the talk dealt with C and type safety, focusing on two approaches: CCured (Necula et al.) and Cyclone (Morisset et al.). The first idea proposed was to insert code to box all values and tag them at runtime to check the right types. This approach was rejected due to the excessive overhead it imposed. A better idea would be to enforce soft typing—do type inference at compile time. Any statically inferred code need not be checked. CCured is based on this principle and introduces three types: T_safe, which corresponds to a single value that need not be checked at runtime; T_seq, which evaluates to a sequence of values that may be traced using fat pointers (perform bounds checks); and T_wild, which indicates a pointer to a tagged value. Security constraints are generated based on how pointers should work. A disadvantage of this approach is that the compiler may insert undesirable checks—within inner loops, for example.

Cyclone, on the other hand, aims to be the type-safe language that CCured maps to. Programmers control where and when to tag values, allocate memory, etc. The downside is that much more information is required from the programmer. For example, there are two ways to do bounds checks, either through the fat keyword or by placing an assertion. Floyt-Hoare Logic is used for verification, and the key challenges that need to be addressed are scalability and soundness. For example, when translating diamonds there is an exponential blowup. Loop invariants pose another problem, and the solution here is to rely on iterative fixed-point computations.

A challenge they have to cope with is that of unsound assumptions. Current work is targeted at increasing the trustworthiness (mismatch in assertions), extensibility, and completeness. Extensibility deals with the problem of using a variety of techniques to check the VCs that are generated. There are three key domain-specific problems that Greg mentioned in terms of completeness: first-order logic does not work; concurrency; and, finally, substructural languages.

PCC is a powerful principle: It minimizes the TCB and places the burden on the code producer. Certifying compilers are a good step in that direction, but they are weak and their theorems are loose. In response to questions, Greg mentioned that Cyclone is currently available and that software maintenance is an interesting direction to explore with VCs.

Refereed Papers

DEFENSES

Summarized by Francis Hsu

- ***Protecting Against Unexpected System Calls***

  C.M. Linn, M. Rajagopalan, S. Baker, C. Collberg, S.K. Debray, and J.H. Hartman, University of Arizona

  Mohan Rajagopalan presented work on a collection of host-based techniques to limit the scope of remote code injection attacks, by denying a remote attacker use of the system calls.

  By recording in an Interrupt Address Table all the addresses of all the legal system calls of an executable before it is run, the technique prevents the use of any newly inserted system calls from injected code. To deter mimicry attacks of injected code using the legitimate system calls in the program, the actual syscall instruction is disguised as other instructions that trap into the kernel. Additional binary obfuscation techniques, such as dead code insertion and layout randomization, make it more difficult to scan for the system calls. To thwart scanning attacks against the code, a pocketing technique splits the code section into noncontinuous segments and unmaps the unused regions of process address space.

  The authors implemented these techniques with a binary rewriting tool that analyzed executables and embedded a new ELF section and a modified OS kernel that made the checks. The techniques worked to protect an executable subjected to synthetic attacks written by the authors, while imposing less than 15% overhead in performance and an increased memory cost of 25%.

- ***Efficient Techniques for Comprehensive Protection from Memory Error Exploits***

  Sandeep Bhatkar, R. Sekar, and Daniel C. DuVarney, Stony Brook University

  Exploitation of memory errors has been responsible for 80% of CERT advisories over the last two years. Although prior work in address space randomization removes the predictability of memory locations, it still allows attacks using existing pointers to calculate relative addresses and does not prevent data overwriting or leakage. Sandeep Bhatkar presented a way to address this problem with a set of transformations on the stack, static data, code, and heap to randomize the absolute location and relative distances of all objects.

  The authors produced a modified compiler and loader to rewrite the C source of existing programs to support the randomization. The actual randomization of the program's objects then only occurs at runtime, enabling the same binary produced by the compiler to be distributed to all users. Experiments have shown that the transformations add an average overhead of 11%, which is comparable to previous address space randomization techniques that did not address all the other attacks mentioned above.

- ***Finding Security Vulnerabilities in Java Application with Static Analysis***

  V. Benjamin Livshits and Monica S. Lam, Stanford University

  While Java has addressed the problem of buffer overruns from unchecked input, Java Web applications are still vulnerable when data in the input buffer is not properly validated. Ben Livshits listed the many sources of injected data to such a Web application, such as parameter manipulation, hidden field manipulation, header manipulation, and cookie poisoning. Once the injected data is in the program, it can be used to exploit the application through SQL command injections, cross-site scripting, and arbitrary command injections. To address the multitude of injection and exploit techniques, Livshits presented a framework for formalizing the vulnerabilities and a static analysis tool to discover vulnerabilities in these applications.

  Vulnerabilities such as SQL injection caused by parameter manipulation can be described at a high level in a Program Query Language (PQL), and these specifications are automatically transformed into a static analysis. The static analysis is both sound and precise, guaranteed to find all the vulnerabilities described in such a specification while limiting the number of false positives. More precision is gained through use of both a context-sensitive analysis and an improved object-naming scheme to help with pointer analysis.

  The authors have collected a set of open source Web applications to form Stanford SecuriBench, a benchmark on which their and others' security tools could be evaluated. Livshits reported that static analysis of this code found a total of 29 security vulnerabilities with only 12 false positives with their most precise analysis.

- ***OPUS: Online Patches and Updates for Security***

  Gautam Altekar, Ilya Bagrak, Paul Burstein, and Andrew Schultz, University of California, Berkeley

  While software vendors may race to provide patches after a discovered security vulnerability, users frequently do not respond with the same urgency. Gautam Altekar suggested that the current patching mechanism is responsible, since patches are unreliable, irreversible, and disruptive. Altekar introduced

OPUS as a practical dynamic patching system to address the problem of patches, making the patch safer and removing the need for a user to restart the patched application.

OPUS consists of three components: a static analysis tool to address the safety of dynamic patches, a dynamic patch generation tool integrated with the GNU build environments, and a runtime patch installation tool. The static analysis identifies a patch's unsafe side effects (e.g., writes to nonlocal data such as the heap or return values). To install the patch, the new, modified function is copied to memory and a forwarding jump is added to the start of the old function. To ensure that the old and new code are not mixed, the redirection is done only after the old function is no longer on the call stack.

To date, the authors have generated dynamic patches for 30 vulnerabilities from vendor-supplied patches without modification. Altekar reported that they could not generate dynamic patches in some instances. These were for cases such as modifications to global values, input configuration files, functions at the top of the call stack, and inline functions.

An attendee asked if restarting applications was such a large problem that online patching would be necessary. Altekar responded that they address a usability issue, where patching has gotten to be so annoying that users are ignoring them. Another attendee suggested that online patching is useful in situations where an administrator patching the system isn't the one sitting at the computer. Such an administrator would not want to disrupt the users and might need to wait for the users to restart the applications on their own.

More information about OPUS is available at http://patch.cs.berkeley.edu.

Invited Talk

What Are We Trying to Prove? Confessions About Certified Code

Peter Lee, Carnegie Mellon University
Summarized by Boniface Hicks, OSB

Peter Lee gave an excellent overview of the work that has been done in proof-carrying code (PCC) and outlined the challenges that remain. PCC developed as a way to say something concrete about a software artifact (e.g., mobile code) without the use of a third party or the heavy overheads of execution monitoring, while still maintaining a small Trusted Computing Base (TCB). Peter Lee and George Necula accomplished this by providing proofs of safety properties, which can be small even for large programs. A proof for the theorem "There are no buffer overflows" would be an example. These proofs are tied into the program text in such a way that they are tamperproof (one can't change the proof without changing the program). Furthermore, because the burden of proof is placed on the software producer, they are lightweight to check. Lee gave the example of a maze. For an infinite-width maze, it might be impossible automatically to find a path from start to finish, but given a path, it is trivial to verify it. For real programs, the "path" can be expressed as an ML program which can be verified merely by ensuring that it typechecks. At this point Lee rhapsodized on the sheer beauty of this simple, yet powerful solution.

Unfortunately, the proofs get oppressively large. As an optimization, the proofs can be turned into "oracle strings." To return to the maze analogy, an oracle string would provide only the answers to queries about which way to go at each intersection. Thus, the oracle string, which would express only "Left," "Right," "Right," for example, could be encoded as a binary string. This gives the proof a very compact form, requiring only slightly more work on the part of the automatic verifier. In a real program, the oracle strings are tied to the program itself. The verifier iterates through the program text, and when it finds a dangerous command (STORE, for instance), it queries the oracle string about whether this command is safe. The oracle string provides the needed evidence. This turns out to be very effective. The checker is less than 52KB and the proofs are generally 0–10% of the program size. In some tests the oracle strings were much smaller than the checksum for the programs. The SpecialJ compiler, which compiles Java class files with oracle strings into x86 binaries, using heavy optimizations justified by proofs, outperformed Java, JavaML, and the JIT compiler. The TCB for PCC is only approximately 100KB.

Unfortunately, the picture is not all so rosy. Lee made his confessions during the second part of the talk. The first major obstacle is that the module that checks the code (VCgen+) is rather beastly. The core of VCgen is 20,000 lines of C code, designed specifically for x86 code output from a Java compiler with a specific policy. To change the policy, one must change the VCgen code. Andrew Appel et al. came up with another solution to alleviate this problem. By finding the right global invariant (a long, complicated thing) and proving that the start state and each future state obeys it, one can use PCC to prove safety properties about programs. They call this Foundational PCC. Other variants of this approach, including TALT and TL-PCC, have been developed as well. Unfortunately, none of the foundational systems are practical yet, because of large proof size or slow proof-checking times.

Another confession Lee made concerned the safety policy. What is the "right" safety policy, and how can it be specified? Currently, the

two key properties that have been used are type safety and memory safety. This is certainly valuable; it eliminates one of the most often exploited security vulnerabilities, buffer overflows. On the other hand, as one member of the audience pointed out, this kind of bug accounts for only 50% of security failures. PCC is fundamentally limited to safety properties. Although safety properties can be used to approximate liveness and information flow properties, this approximation leaves something to be desired. When specifying policy, one really wants to say something direct: that no program should write to the kernel, for example. In PCC such a property can only be expressed in an indirect way, by specifying programs' structural rules that imply this condition. Some promising directions for developing solutions to this problem are use of first-order temporal logic, and model checking.

In conclusion, Lee asserted that certified code is a great way to ensure safe code. Proof-carrying code is able to eliminate the most basic program flaws exploited in security attacks. Engineering PCC into a practical system, however, is challenging. Furthermore, some attacks are not (yet!) able to be addressed by PCC. For example, one would like to guard against trojan horses. It is usually the case, however, that trojan horses are safe and live. In this case, PCC may not be very useful, because it may only verify that the trojans won't crash. Vergil Gligor asked a question about the limitations of approximating information flow policies with safety policies. He noted, for example, that Bell-LaPadula and Biba are both approximations of information flow policies. Each eliminates a different covert channel. Their composition, however, introduces a new covert channel. This goes to show that one of the hard problems in certifying code is getting the security policy right—hopefully, PCC can make some headway in this.

Refereed Papers

BUILDING SECURE SYSTEMS

Summarized by Francis Hsu

- ***Fixing Races for Fun and Profit: How to Abuse atime***

  Nikita Borisov, Rob Johnson, Naveen Sastry, and David Wagner, University of California, Berkeley

  In "Fixing Races for Fun and Profit: How to Use access(2)" at last year's USENIX Security Symposium, Dean and Hu presented a countermeasure to a race condition attack, where an adversary is required to win k-races instead of just one for an attack to succeed. They accomplish that by making the access and open calls in a loop, so that an attacker would need to change the symbolic links to point to the correct files many times. This year Naveen Sastry presented an attack on such a defense by constructing a filesystem maze to win the races against the loop and synchronizing with the access and open system calls.

  Filesystem mazes ensnare the victim process making the access and open checks, forcing the process to block for I/O and allowing the attacker to win the race. The attack was constructed by creating chains of deep directory trees and placing the target at the end of it. If one of the directories was not in the buffer cache, the victim process would need to block and incur disk I/O. To reliably detect when each access or open call began, the authors monitored the atime of a symbolic link in the path given to the victim process. Even against a k-race algorithm where k=100, the author's attack succeeded 100 out of 100 trials on one of the platforms tested.

  An attendee observed that the order of the access and open calls was built into the assumptions of the attack and asked what would happen if the order was randomized. Sastry deftly advanced to a backup slide that described the attack on a randomized k-race using system call distinguishers. He explained that the information on the system call being made can be gathered from the process ID under the /proc file system. Another attendee noted that having deep directories of hundreds or thousands of directories for the attack might be detected as unusual behavior. Sastry reported that while mazes of size 800 were used in the attacks, he speculated that much smaller mazes of 10 or 20 might work if an effective strategy for flushing the buffer cache at the same time was used.

- ***Building an Application-Aware IPSec Policy System***

  Heng Yin and Haining Wang, College of William and Mary

  Heng Yin began his presentation by describing the security benefits of IPSec, but noted the failing that the transport mode of IPSec is not widely used because of the lack of PKI deployment and poor application support. The IPSec policy support lacked knowledge about application context, disallowing fine-grained policy that might be needed by applications such as peer-to-peer systems that deal with unpredictable remote hosts and dynamic port usage. Additionally, the application API support of IPSec is inferior compared to the more popular SSL/TLS.

  The authors addressed these weaknesses of IPSec by creating an application-aware IPSec policy system, and they implemented it on a Linux 2.6 system. Evaluation of the system revealed that IPSec could counter network-level attacks such as SYN flooding using fewer CPU cycles than other mechanisms such as SYN cookies. The authors also secured the FTP protocol with an IPSec policy to provide privacy for

the communications, and they observed that files could be transferred faster under the secured FTP than with sftp, a protocol secured at the application level.

- ***Shredding Your Garbage: Reducing Data Lifetime Through Secure Deallocation***

  Jim Chow, Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum, Stanford University

  Jim Chow began his presentation by posing the following rhetorical question: How good are computers at keeping secrets? To gauge the lifetime of data in a computer's memory, the authors ran a small program that filled several megabytes of memory with markers, then freed it. They then continued to use their machines normally. At the end of each day, they would observe the memory contents. Surprisingly, they were able to recover kilobytes to megabytes of data, weeks afterward, even after the machines were rebooted.

  Chow noted that good application programmers may remember to properly overwrite memory that may have contained sensitive data. However, he argued that protecting sensitive data is a whole-system property, since data in memory may be copied by the system to many different buffers or flushed to a swapfile on disk. To remedy this, the authors propose secure deallocation of the memory by explicitly clearing the contents of any memory whenever it is freed by the system. Chow reported that such a system incurred 0–7% performance overhead even in the worst cases. He explained this minimal overhead was because while data would be free in kilobytes or megabytes per second, the system could zero out memory in gigabytes per second.

  The talk was followed by a lively Q&A session. An attendee asked if the authors had looked at the latency overhead of their system. Chow replied that the paper did not specifically address latency, but noted that applications didn't normally batch free operations, so the overhead was spread out. Another attendee noticed that some benchmarks reportedly ran faster with the secure deallocation system, which Chow attributed to noise in the benchmarks since overheads were very small. When asked about the half-life of data in their marking experiment, Chow said that it was very short, within seconds, but the time to live for some bits were very long. An attendee wondered if the experiments demonstrated that the buffer caches on the tested operating systems were inefficient, since the unified virtual memory should have allowed the regular memory to be used for buffering I/O. Chow explained that holes in the pages used were responsible for allowing data to survive usage by the buffer cache. While pages were reused and reclaimed, they were not completely overwritten in the process.

Invited Talk

- ***Six Lightning Talks (and a long one)***

  Ben Laurie, The Bunker

  Summarized by Stefan Kelm

  Ben opened his remarks by admitting that he thought he'd have to give a one-hour presentation until he was told that his session would cover 90 minutes. He therefore added some topics and changed the title of his talk from "Four Lightning Talks." (Ben impressed the audience with some extremely fancy animations throughout his talk, most of which he apparently hadn't seen before himself.)

  Ben delved into the first of his six (plus one) topics: "Why open source vendors are bad for security." He argued that vendors of open source software often cause security problems because they change default installation directories, split software packages, fail to change version numbers correctly, and sometimes even introduce security flaws during packaging. This in turn makes it hard for the user or administrator to apply security patches. Ben stated that vendors create the myth that they are needed for reliability, which, in his opinion, is not true. Ben then talked about the much-discussed issue of full disclosure and argued that the role of coordinating bodies such as CERT or NISCC in practice is reduced to protecting their stakeholders. With respect to the open source vendors the solution he proposed was that "packagers should make themselves redundant."

  His next topic was on an almost ancient rule of thumb, first defined in RFC 760: "An implementation should be conservative in its sending behavior and liberal in its receiving behavior." He gave some examples of servers which, in his opinion, are way too liberal in what they accept as an incoming connection. He cited HTTP Request Smuggling, a real-life attack scenario that has not garnered much public discussion. He concluded that being liberal in what a server accepts is bad for security.

  DNSSec, which Ben covered next, has been in the IETF standards for quite some time but is not being used by anyone, due to several (mostly organizational) problems. Ben described some of those problems: the size of DNSSec packets, islands of trust, the key-rollover problem, and issuing DNSSec-secured negative responses without allowing what is called "zone walking" DNS servers. He pointed out solutions to those problems, even though some of them remain in the standards.

  Next, Ben discussed privacy-enhanced identity management (PEIM) and a library he and a colleague are currently writing to implement a bit-commitment scheme which is related to zero-knowledge (ZK) proofs. As an example, he mentioned the infamous "Where's Waldo?" question in which I want to prove I know

where Waldo is without revealing Waldo's location. The library they're writing implements several ZK proofs and provides low-level functions to do the necessary crypto operations, but no protocols.

Ben moved to his next sub-talk, the focus of which was an OpenPGP SDK he is currently writing. The SDK will be a BSD-licensed free C implementation of OpenPGP which aims to be complete, flexible, storage agnostic, protocol agnostic, and correct (in contrast to being too liberal, as proposed in RFC 760). Since an end-user application already exists with gpg, all they'll be providing is a library, not an application.

Before starting with his "real" talk, Ben briefly discussed anonymous presence, another solution to secretly communicating with others. In this example a so-called "rendezvous server" allows Alice (who else?) to rendezvous with (guess who) Bob. The two main objectives are that Alice doesn't want anyone to know she's talking to Bob, and Alice and Bob don't want their conversations to be linked, even in the presence of a global passive adversary. Even though the rendezvous server is not regarded as trusted, the protocol allows for these goals to be achieved. Apres, an anonymous-presence implementation, is a Perl library written by Ben and implemented for plain TCP and IRC.

After these short talks Ben tried to squeeze his remaining "long talk" into the final minutes but failed to do so. His final talk was on another implementation of his called CaPerl, which implements capabilities in the Perl programming language. If one wants to run possibly hostile code safely, traditional approaches such as sandboxes and jails often fail for several reasons: they often are either too restrictive or too lax; moreover, there's no easy way to specify access to a file by a certain program while disallowing access by any other program.

A solution to this problem is capabilities (not to be confused with POSIX capabilities), nicely described by Ben as "an opaque thing that represents the ability to do something." Using capabilities, an environment can choose exactly what the visiting code can do. He went on to talk about how to implement capabilities in different programming languages and, finally, presented CaPerl, his "surprisingly small" implementation: CaPerl is able to convert standard Perl into a capabilities language, and it compiles into standard Perl, the main modification being the introduction of trusted vs. untrusted code within CaPerl. (Ben's explanation of trusted vs. untrusted code was way too short, so the interested user should check both his slides and his Web site for further information.) On using CaPerl the output is Perl, which one runs the normal way, with the CaPerl libraries in the path.

For more information, have a look at Ben's home page at http://www.apache-ssl.org/ben.html.

Work-in-Progress Reports

Summarized by Jonathon Duerig

- ***The Program Counter Security Model: Automatic Detection and Removal of Control-Flow Side Channel Attacks***

  David Molnar, Matt Piotrowski, David Schultz, and David Wagner

  In a regular cryptographic attack model, the adversary has access to a box with a key and an arbitrary mechanism. The adversary sees output given known inputs. In the real world, other characteristics can be used, such as time or power usage. This WiP is about preventing attacks based on side channels that leak control flow information. Suppose that the adversary can track the program counter as a given algorithm is executed. Given this model, a system is secure if the adversary learns nothing in spite of this extra information. The authors are developing a system to automatically detect and fix algorithms (in C) that are insecure in the face of a leaked program counter. The cost of modifying an algorithm to resist an attack using the program counter is a fivefold increase in time and a twofold increase in space. They are also developing a static analyzer for assembler code based on taint. This can detect insecurities introduced by an optimizing compiler.

- ***Implementing N-Variant Systems***

  Benjamin Cox, University of Virginia

  Benjamin Cox is developing a system to protect vulnerable Web services. An input replicator splits input from the user to several variants of a Web server. These variants are artificially diverse, running in disjoint address spaces and with potentially different instruction sets. A monitor detects when system call parameters disagree and shuts all Web servers down if they do. A simultaneous attack is required to compromise the system as a whole, and the artificial diversity makes simultaneity more difficult. He has thwarted an attack on a vulnerable Web server (a format string attack). Open questions remain: What kinds of variations work well? What kinds of classes of attacks can we prevent? Can the system perform acceptably? There are two current problems with the system. First, some input and output can be done without resorting to system calls. The monitor may therefore be bypassed by such methods. Second, while the server is harder to compromise, it is easier to kill. The long-term goal is to get some provable security that doesn't rely on secrets: for instance, a system where even if the variations were known, the system would still be secure.

- ***Effortless Secure Enrollment for Wireless Networks Using Location-Limited Channels***

  Jeff Shirley, University of Virginia

  How do you enroll temporary users into wireless networks? Such a system must be easy and provide mutual authentication, ensuring that the enrollee is an authorized user and that the wireless network is trusted. The solution is location-limited channels. The author proposes audio tones as such a channel. It is human-evident, the range is limited, and it is available on all systems. Previously authorized users act as intermediaries. They verify through the audio property that the authorized new users are at the same place. This leverages the relationship between the current user and the prospective user. The author has a working implementation. There are several open issues: How should the client software be distributed? How can interoperability be ensured? Can the reliability and transmission speed of the channel be improved?

- ***Revamping Security Patching with Virtual Patches***

  Gautam Altekar, University of California, Berkeley

  Patching is ineffective because it is unreliable, disruptive, and irreversible. There is no extant work that addresses all of these issues. Many kinds of patches have two basic parts: a check and a fix. The check is a test added to the original code to determine if the vulnerability will be triggered. The fix is the code to handle the anomalous situation. The author presents the notion of a virtual patch, where the developer denotes which part of the patch is the check and which part is the fix. The check is sandboxed to prevent a side effect from affecting the rest of the program unless the vulnerability is triggered. Each check and fix can be represented as a nested C function. Much of the overhead can be optimized away. Virtual patches are nondisruptive, because they are simple additions to the program and can be inserted dynamically. The limitation is that the programmer must explicitly annotate the code to indicate which part of the patch is the check and which part is the fix. Is there a virtual patch that is equivalent to any conventional one? If so, conversion is possible. Given a patch for some bug, is there some way to change the behavior of the program to allow a single check and fix?

- ***Automatically Hardening Web Applications Using Precise Tainting***

  Salvatore Guarnieri, University of Virginia

  The goal of the system is to prevent PHP and SQL injection attacks. An example of the relevance of this problem is the recent attack on phpBB which was based on PHP injection. The problem was that the programmer called "http-decode" one too many times. This allowed code to be inserted. The solution is to insert a dynamic fine-grained taint analysis. All user-supplied data is marked as dangerous. Taint is determined on a character granularity rather than the coarser-grained string granularity. The system is implemented in PHP. It modifies taint info in the same way that the string is modified. It prevents tainted data from being used for system state. The system detects what the tainted information will be interpreted as. Dangerous tokens, such as unexpected delimiters, can be detected. Server administrators can install this system merely by switching the version. Application developers need do nothing.

- ***Automatic IP Address Assignment for Efficient, Correct Firewalls***

  Jonathon Duerig, Robert Ricci, John Byers, and Jay Lepreau

  Having worked on optimizing the assignment of IP addresses to nodes in a network so as to minimize the size of routing tables, the authors are now looking at extending this work into minimizing firewall rule sets. Firewalls typically match IP addresses using subnets, but this approach scales poorly if the sets of hosts that are protected by a particular firewall rule have discontinuous subnets. In addition to efficiency concerns, this produces correctness problems. The more firewall rules there are, the more likely it is that one of them is incorrect (i.e., does not express the desired policy). Given a complex topology with a large number of hosts and policies, an organization can end up with a huge number of rules. The authors' work on routing-table minimization uses a metric called Routing Equivalent Sets (RES), which quantifies the extent to which routes to sets of destinations can be aggregated. Using this metric, they achieve a two- or threefold decrease in the number of routes. There are two basic approaches to adapting RES to firewall rule sets, depending on how much information is supplied. If the only information is the firewall locations as annotations, then when evaluating RES, count only the firewalls. If the firewall rule sets are also provided, then the algorithm can assign addresses using sets of nodes covered by a common policy. Both of these approaches look promising, but need to be evaluated.

- ***Turtle: Safe and Private Data Sharing***

- ***Bogdan C. Popescu, Bruno Crispo, and Andrew S. Tanenbaum, Vrije Universiteit, Amsterdam, The Netherlands; Petr Matejka, Charles University, Prague, Czech Republic***

  The goal of Turtle is to use a peer-to-peer network for safe sharing of sensitive data which cannot be censored by an adversary. The best current example of this kind of system is Freenet, but even it fails to provide complete protection. The connectivity model is open and good nodes can interact with censored nodes when exchanging data. When a good node is so exposed, the owner of the good node is open

to legal harassment. Turtle creates a peer-to-peer overlay network based on social links. Communication between links is encrypted. The key distribution must be completely decentralized, and messages must go hop by hop across the overlay network. To start a virtual connection, flood query is used to find the endpoint. Only parties that trust each other communicate. There is no direct link between the source and the destination of the virtual circuit. This means that even if the destination is compromised, there is no way to find out which node the source is and vice versa. Though node compromises cause only local damage and this system is immune to Sibyl attacks, the system is still susceptible to a subpoena attack.

- ***Towards an Online Flow-Level Anomaly/Intrusion Detection System for High-Speed Networks***

  Yan Chen, Northwestern University

  Most intrusion detection systems are end-host-based. Rapidly and accurately identifying attacks is critical for large network operators. Therefore the author proposes a system which detects network anomalies at the routers. The system stores data-streaming computation in reversible sketches. This allows millions of flows to be recorded. So far, the author has focused on TCP SYN scanning. Existing schemes for detection have high false-positive rates. The system infers key characteristics of malicious flows for mitigation. This is the first flow-level intrusion detection system that can sustain tens of gigabytes per second. The input streams are summarized and values are forecast for the next intervals. If the incoming value is different from the forecast, then an anomaly has been detected. This was evaluated on 239 million hosts with worst-case traffic.

- ***Mitigating DoS Through Basic TPM Operations***

  William Enck, SIIS Lab, Penn State University

  Denial of service (DoS) attacks are an ever-increasing problem. One way of avoiding DoS attacks is by requiring clients to solve computational puzzles, which slows down the rate at which a client can make requests. There is an inherent unfairness about this system because some computers are orders of magnitude more efficient than others. One way to level the playing field is by requiring the puzzles to be calculated by the Trusted Platform Module (TPM), the hardware processor behind trusted computing. There are fundamental characteristics of the TPM: accessing it is slow and it cannot execute arbitrary code. This slow access can be used as a rate limiter. The puzzle that the client must solve can involve accessing the TPM a certain number of times. This would provide a constant delay. TPMs will be ubiquitous; therefore they can be used as an efficient and effective resource limit.

- ***PorKI: Making PKI Portable in Enterprise Environments***

  Sara Sinclair and Sean Smith, Dartmouth College PKI/Trust Lab

  The goal of PorKI is to attack the problem of usability in public key infrastructures. Users need their keys to be portable. Whether they actually move from one computer to another or whether they are running a number of virtual machines on the same physical workstation, they want to use their standard key pairs everywhere. One solution is to have a key dongle, but these require special software. PorKI puts the key pairs on a Palm Pilot and transfers them via Bluetooth (though they do not rely on the Bluetooth security model). The Palm Pilots can generate short-lived keys and these can interact with keys on the workstations themselves. The information can be used to customize the user experience, for instance by not authenticating sensitive data on a public computer (notifying the user appropriately). Some trust information can be stored in the machine without requiring user effort. There are many other applications. Open issues include protecting the key repository, finding a good way to establish trust between the workstation and the PDA, and extending the key-transfer protocol beyond Bluetooth.

- ***DETER***

  Terry V. Benzel, University of Southern California

  In the past, most network security research has been done in small or isolated labs. DETER aims to provide more objective, scientific, and reproducible measurements. DETER provides a secure infrastructure with networks, tools, methodologies, and supporting processes, plus reusable libraries for conducting realistic experiments. It takes concepts from science and math where results are reproducible. DETER, which is accessible over the wide area network, also allows canned topologies and attacks, and quick runs of different experiments. Based on Emulab, DETER has 201 nodes of four different types. It contains a control plane and various types of PCs and switches. Each node can run virtualized. Clients can run FreeBSD and Linux, and soon will be able to run Windows. DETER is hosting an upcoming workshop. More information about DETER and the workshop can be found at http://www.isi.edu/deter/.

- ***Minimizing the TCB***

  David Lie, University of Toronto

  The Trusted Computing Base (TCB) is the group of components of a system that a segment of code must trust to function correctly and securely. The operating system,

libraries, and other applications are all part of the TCB. For most systems the TCB is millions of lines of code. The author shows how to minimize the TCB for a particular security-critical section of code. He does this by running that piece of code in its own virtual machine with a custom operating system. Since the operating system is single-threaded and need not optimize heavily, it can be much simpler than a general-purpose operating system. This can reduce the size of the TCB from millions of lines of code to around ten thousand. At that scale, it becomes feasible to run static analysis tools and gain even more confidence in the correctness of the code. The security-critical section can even be implemented in a safer language. The only remaining issue is that the developer has to select the portion of the program that is security-critical, which may be nontrivial.

- ***Strider HoneyMonkeys: Active Client-Side Honeypots for Finding Web Sites That Exploit Browser Vulnerabilities***

  Yi-Ming Wang, Microsoft Research (Strider Research Group)

  A user visits a URL with a Web browser. Since Web sites can transparently redirect the browser, a malicious URL can send the browser to many different intermediate URLs. Each intermediate URL can try a different exploit on the browser. HoneyMonkeys are programs that emulate a human using a browser. They seek out Web sites with various versions of the browser software, trying to get infected. A HoneyMonkey is inside a virtual machine for quick reset after an infection. Infections are detected because their payload compromises the host by modifying the registry or the file system. HoneyMonkeys use previously developed software (Strider Gatekeeper and Strider Ghostbuster) to determine whether the payload has been delivered. HoneyMonkeys detect the payload rather than the vulnerability. This means that they can detect an exploit even if the vulnerability is unknown (zero day). Several versions of the browser are used: an unpatched version to detect all malicious URLs, partially patched versions to detect how effective patching is, and fully patched versions to detect zero-day exploits. The HoneyMonkey crawls when it detects a site with many exploits. Malicious sites tend to be well connected with each other. The sites that host the original URLs redirect to the spyware sites who pay them. Information is frequently stored in the redirected URLs, including vulnerability names and account names. Many malicious sites are among the top click-through links from a search engine. They are most likely to occur on sites about celebrities, game cheats, song lyrics, and wallpaper. Because HoneyMonkeys detect zero-day exploits, they can be used to discourage such exploits.

- ***Making Intrusion Detection Systems Interactive and Collaborative***

  Scott Campbell and Steve Chan, Lawrence Berkeley National Laboratory, NERSC

  Most open source applications are controlled by text configuration files. They are often non-interactive. This applies to security monitoring response software as well. The lack of interactivity makes adaptive changes more difficult and makes it much harder to teach or train new operators to use them. The presented work improves upon Bro, a stateful network intrusion detection system, in two ways. First, the authors added an interactive command line interface to it. This allowed state, such as memory or CPU usage, or host characteristics to be queried. It also enabled, among other things, additional monitoring of particular connections. Second, they turned the command line interface into a Jabber bot. The system can be monitored and controlled through an instant messenger conference. This allows many interactive sessions to be run simultaneously. Each bot can join the same conference and be controlled and monitored in tandem. Logs can be saved easily in any chat program. New operators can observe firsthand the interactions of more experienced administrators. This also allows the network intrusion detection system to be run easily from anywhere using any Jabber client.