conference reports

THANKS TO OUR SUMMARIZERS

SECURITY '06

Madhukar Anand
Tanya Bragin
Kevin Butler
Lionel Litty
Michael Locasto
Bryan D. Payne
Manigandan Radhakrishnan
Micah Sherr
Patrick Traynor
Wei Xu

METRICON 1.0

Dan Geer

NSPW '06

Matt Bishop
Michael Collins
Carrie Gates
Abe Singer

CONTENTS

Security '06: 15th USENIX Security Symposium

Vancouver, B.C., Canada
July 31–August 4, 2006

KEYNOTE ADDRESS

The Current State of the War on Terrorism and What It Means for Homeland Security and Technology

Richard A. Clarke, Chairman, Good Harbor Consulting LLC

Summarized by Kevin Butler

Richard Clarke gave an impassioned, scathing indictment of the U.S. government's policies and practices that electrified the crowd in Vancouver, bringing them to their feet at the conclusion of his keynote speech. Clarke was the former counterterrorism advisor on the U.S. National Security Council for the Bush administration and has advised every administration since Reagan. He brought his 30 years of consulting work to bear on his speech, where he laid out many of the issues facing both the security research community and the populace at large, given the current global situation.

Clarke began by commenting that, nearing the fifth anniversary of the September 11 attacks, we should examine what measures to increase security have been addressed since that time. As Clarke pointed out, five years is a long time: It took less than five years for the Allies to destroy Nazi Germany and Imperial Japan. The day after the 9/11 attacks, President Bush asked Clarke to outline a series of plans to maintain security. Clarke responded with a series of blueprints for going after Al Qaeda while simultaneously addressing vulnerabilities at home. Part of these plans concerned the state of IT security, given the growing dependence of the nation on cyber-based systems. Unfortunately, since that time, little of what had been discussed was implemented, while other, more draconian measures that impinge on civil liberties have instead taken their place.

Much of Clarke's criticism was aimed at the administration's handling of the Al Qaeda threat. After 9/11 there was talk about finding the cells and taking out the leadership, when the focus should have also included attacking the root causes of problems: a battle of ideas, where the West displays a better ideology that is more appealing to the Islamic world than that promulgated by Al Qaeda. The results have not been great to this point, as Al Qaeda still exists and has transformed from a hierarchical, pyramid structure to a decentralized organization with dozens of individual organizations. In the 36 months after 9/11, Clarke asserted, the number of attacks by groups related to Al Qaeda doubled around the world. Clarke satirically referred to an Arabic satellite TV station established by the U.S. government "that nobody watches" as our entry into the battle of ideas.

He also showed many similarities between the American occupation of Iraq and the French occupation of Algeria, a conflict that the French ultimately lost. By alienating the Iraqi population thanks to the stories from Abu Grahaib, the assaults on Fallujah, and the perception of killing innocent civilians in Iraq (despite what the truth may be, Al Jazeera and Al Arabia show daily images of Americans killing women and children through aerial and other assaults), we are losing the battle of ideas and convincing the Iraqis that Al Qaeda was right: For its oil, we are taking over a country that did nothing to us. We are almost fulfilling bin Laden's prophecies. What are the metrics of success in Iraq? Clarke suggested these are the economy and stability of the country. When asked how many innocent civilians had been killed in Iraq, President Bush replied, "Around 30,000"; however, a team from Johns Hopkins did a field survey using internationally accepted metrics and found the number was close to 100,000, not counting the 2,500 dead and 10,000 wounded Americans. The Pentagon admits to having already spent $400 billion on the war, and some estimates put the total price tag at over a trillion dollars. Part of the rationale of the high economic and human cost is that fighting the terrorists "over there" means we won't be fighting them "here." This argument is logically fallacious, however; nothing we are doing there prevents them from fighting us here or anywhere else, as evidenced by the attacks on the Madrid train system, attacks on the London subway, and the planned attacks in Toronto. None of the fighting overseas prevented these situations, nor will it prevent future attacks in Canada and the United States.

Clarke argued that more effort should have been spent protecting critical infrastructure and minimizing the possibilities of damage here. An ABC report showed how easy it was to take a backpack onto a train and leave it unattended, the modus operandi of the Madrid attacks. Similarly, 120 chemical plants in the United States store lethal gas, with over a million people in the plume radius. If one such plant were attacked and a plume unleashed, over 17,000 casualties would be expected; however, Congress has debated plant protection for over three years but nothing has been done. This has occurred with issue after issue. Instead, measures that reduce civil liberties—such as the ineffective color-coded threat system—have been implemented by abusing the term "security." The government engaged in mass wiretapping without any judicial oversight and, combined with other abuses, this has led to the association of security with Big Brother in the minds of the public.

The current situation has implications for security researchers, as currently many systems were not designed with security in mind, and Clarke asserted that research into securing these systems is necessary. However, DARPA no longer funds many of these activities, and research money is being handled by the Department of Homeland Security through the HSARPRA program; however, this program is severely underfunded, with this year's budget cut from $16 million to $12 million—Clarke noted that such was the administration's commitment to cybersecurity research. This view is shared by the president's own committee, which identified the program as requiring significant new funding, but this has not yet come to pass.

Clarke made some controversial statements describing what regulation should be required. He suggested that identifying best practices for ISPs is possible to regulate the service provider industry. He also suggested governmental regulation in critical systems such as electrical power systems and banking, but he considers the current government ideologically opposed to such measures. The upshot is that many opportunities to focus on real security have been wasted. The best thing for us in the community to do is not to rely on the government, but to unite and come up with ways of solving problems ourselves, as well as staying vigilant for governmental attempts to further erode civil liberties. He ended his address by noting that when people erode civil liberties, privacy, and constitutional guarantees, they need to be reminded of their oath to defend the Constitution against all enemies, lest they become the enemies themselves.

A spirited question period followed. To the question "Is it that the administration is apathetic, malicious, or doesn't have a clue?" Clarke responded, "Yes." The administration, in his view, cannot be educated and needs to be replaced. He also reiterated the need to win the battle of ideas by supporting moderate and progressive voices in the Middle East, and he noted the amount of "security theatre" being practiced, where there is a show of security without substance. A particularly resonant point was the need to find and disclose vulnerabilities; as Clarke noted, our real enemies already know our vulnerabilities and security processes, and pretending we won't talk about them because these enemies will learn about them is very wrong. One of the best defenses against future attacks is an environment of openness and disclosure. Another important point was that we need to accept that risk is present and casualties will occur; the question of an attack is not "if" but, rather, "when." We should focus on mitigating the effects of the next attacks whenever they may happen, while focusing on ideals such as due process and guaranteed Constitutional rights that are the hallmarks of our civilization: If we cannot preserve those, then what are we fighting for?

AUTHENTICATION

Summarized by Micah Sherr

■ *A Usability Study and Critique of Two Password Managers*

Sonia Chiasson, P.C. van Oorschot, and Robert Biddle, Carleton University

Sonia Chiasson presented a usability study of two password managers, PwdHash (USENIX Security '05) and Password Multiplier (WWW2005). Although both password managers attempt to increase security by shifting the burden of creating and maintaining strong passwords away from the user, the study demonstrated that usability issues in the applications led to incorrect usage of the managers and, in some cases, to the leakage of password information.

Sonia presented the results of a follow-up questionnaire given to members of the focus group. Participants indicated that they did not perceive an improved sense of security. Users were uncomfortable with not knowing the passwords to the sites they visited, and many had misconceptions as to the operation of the managers (e.g., the belief that their passwords were kept in a large and remote database). At the same time, the transparency of the applications often led to false senses of security. For example, users who installed a password manager but failed to activate it falsely believed their passwords were being protected. Sonia concluded by noting that the usability of a system directly impacts its security. Security systems must support users' conceptions as to how software should operate; they should imbue a sound (and not necessarily transparent) sense of security.

■ *On the Release of CRLs in Public Key Infrastructure*

Chengyu Ma, Beijing University; Nan Hu and Yingjiu Li, Singapore Management University

The talk was given by Yingjiu Li, who presented a method of determining a more nearly optimal schedule for distributing certificate revocation lists (CRLs)—time-stamped lists of certificates that have expired or become compromised or otherwise invalidated. In the first part of his talk, Yingjiu described an empirical study of CRL distributions. By examining CRL release data from VeriSign, he and his coauthors derived a probability distribution function for the distribution of CRLs. They determined that most revocation requests occur within the first few days after a certificate is issued and are less common as the certificate ages.

In the second part of his talk, Yingjiu introduced a new economic model for optimizing the distribution of CRLs. The objective of the model is to minimize the total operational cost of distributing the certificates while maintaining a reasonable degree of security. Yingjiu presented evidence that different types of certificate authorities (CAs)—i.e., start-up CAs and well-established CAs—should adopt different strategies for CRL distribution and that the number of CRL distributions will stabilize after a period of time.

■ *Biometric Authentication Revisited: Understanding the Impact of Wolves in Sheep's Clothing*

Lucas Ballard and Fabian Monrose, Johns Hopkins University; Daniel Lopresti, Lehigh University

Lucas Ballard presented a study of biometric authentication, a method of using biological and physiological traits to authenticate humans. In particular, Lucas's talk focused on exposing weaknesses in both the implementations and evaluation methods for biometric authentication schemes.

The presentation examined the security of biometrics based on the writing of passphrases (i.e., a human is authenticated by comparing his or her passphrase against a corpus of previously supplied handwriting samples). Lucas presented results that show that even novice forgers (students who showed some talent in producing forgeries) can produce forgeries far better than what the biometric systems predicted could be produced.

In the last part of the talk, Lucas described a system for automating forgeries based on a generative model. A synthesis algorithm selects n-grams from a corpus of the target's handwriting samples. The n-grams are then used to forge a passphrase in the target's handwriting style. Lucas presented results that show that the generated and forged passphrases perform better than those generated by skilled forgers.

INVITED TALK

■ *Selling Security to Software Developers: Lessons Learned While Building a Commercial Static Analysis Tool*

Brian Chess, Fortify Software

Summarized by Tanya Bragin

Brian Chess said that finding security vulnerabilities in source code is like trying to find the proverbial "needle in a haystack." However, he stressed that well-built, highly configurable tools not only help organizations uncover potential problems but also shape long-term software development policy and its enforcement.

There are several reasons static analysis works well. First, existing software development procedures can easily be modified to employ source code analysis and in doing so allow bugs to be fixed before deployment. Second, there are a limited number of common vulnerability types, so patterns can be used to detect them via static analysis methods. Third, static analysis tools can be built to give uniform coverage over large code bases in reasonable time, as opposed to, for example, those using dynamic analysis, and thus they are practical.

Although there have been many academic efforts to develop static analysis techniques, they are not appropriate for use in commercial environments, for a number of reasons. First, the tools developed in academic settings were not explicitly designed for the scale and variety of environments in which they could be used. Consequently, they generally lack the customizability needed in a complex organization. Second, reporting in such tools tends to be insufficient for enterprise use. Finally, the management interfaces required for over-time tracking of defects and activities is lacking.

It turns out that these shortcomings are key to the eventual success of a practical static analysis tool. Aside from simply working well, finding important vulnerabilities, not crashing, not hanging, and not having any vulnerabilities of its own, the tool has to be scalable, well-documented, and intuitive to a person who is not an expert in all nuances of software security. The main users of static analysis tools in industry tend to not just be developers themselves, but security auditing departments that outline, monitor, and enforce organization-wide security policies, and Brian stressed the importance of working closely with such teams.

Adopting new technology that fundamentally changes how software should be written is hard and can cause resistance from developers. "Developers are optimizers," stated Brian, himself having come from that background. "If they think that security is optional, they will optimize it out." He continued with a couple of amusing examples of excuses he has heard out in the field about why obvious vulnerabilities do not actually make code insecure, such as "I trust the system administrator" or "That code will never be run." But in reality these vulnerabilities can open unexpected backdoors into critical software, which significantly damage the company brand, open it up to liability, and cause economic losses.

Teaching developers to think about security is critical, concluded Brian, but we need to enable them to do their job efficiently with the necessary tools. A good analogy is spelling, he offered. We teach everybody to spell, but we still have spell checkers in our word processors. Similarly to how we already have compilers to check for proper programming language syntax, static analysis tools can go a step further and help enforce good programming practices that result in fewer software vulnerabilities and incrementally help developers write good code on their own.

INVITED TALK

- *Security Vulnerabilities, Exploits, and Attack Patterns: 15 Years of Art, Pseudo-Science, Fun, and Profit*

Ivan Arce, Core Security Technologies

Summarized by Madhukar Anand

In conferences such as the USENIX Security Symposium, it is typical to learn about the work and experience of academicians and researchers in computer science. The invited talk by Ivan Arce (CTO of Core Security Technologies) was very unlike this. It offered an insightful, alternate perspective on the evolution of information security. Arce presented his views based on his forays and experiments in designing systems for fun and profit.

Arce, introduced to the world of computers through the Commodore VIC 20, recounted how he saw computers as a toy to experiment with. Consequently, he grew up with a notion of computers as a game rather than a tool. Programming the VICs taught him that computers could be tailored and have many hidden features, and he learned the key difference between an enemy and an adversary. Some of these experiences have shaped his views on the evolution of attacks and security paradigms in the past 15 years.

Evolution of Attacks: In the early 1990s (post–RT Morris worm) there was no Linux, TCP/IP stack in Windows, or the Web. Security information flowed from technical journals, bulletin boards, and underground publications. For most people outside of the academic world, these underground publications were the main source. This was the time when many home-computer users started turning professional—kids who picked up programming, started exploring their home PCs, and turned to security for jobs. By the mid-1990s, exploits in shell code become common and stack smashing was done for fun and profit. From then on, there has been increasing complexity in software and, subsequently, an increasing sophistication in attacker skills.

Today, the attacker has the ability to hack networks, write viruses, and reverse-engineer software. The trend now, post-2001, is to go directly at a workstation and own the whole network. This works because there are myriad vulnerable applications, it is difficult to implement inventory, it is difficult to deploy and manage countermeasures, desktops are operated by careless and unaware users, and, above all, it represents the law of "minimal effort for maximum profit."

Arce observed that in our "quest for completeness," we have too strongly emphasized understanding attacks, rather than vulnerabilities. To illustrate this point, he gave the example of the ssh v1 CRC insertion attack discovered in 1998. The patch distributed to fix this vulnerability was itself found to contain a bug. Therefore, it is important that we focus on getting the basics right rather than going after the obscure and complicated.

Another problem with current systems is that underlying models of systems have not kept up with changing technology. The management, deployment (policies, ACLs, RBAC, authentication tokens, etc.), and generation of security value (AV/IDS signatures, patches, certificates, vulnerability checks, etc.) in an information system are centralized. In contrast, outside of the information world, there has been an evolution of open source software, P2P, mobile code, and technology based on social networking and reputation- and collaboration-based systems. So the current information security technology and business models should not ignore them.

In conclusion, Arce felt that the generation that entered the information security field in the early 1990s has successfully managed to create a market for security. This generation has embraced and promoted open and unmediated discussion about security. He felt that we are better off now than we were 15 years ago (at least, we know where we are wrong), and so he urged everyone to learn from past mistakes, stop reinventing the wheel, and brace ourselves for a new generation in information security.

ATTACKS

Summarized by Patrick Traynor

■ *How to Build a Low-Cost, Extended-Range RFID Skimmer*

Ilan Kirschenbaum and Avishai Wool, Tel Aviv University

Ilan Kirschenbaum explained that as radio-frequency identifier (RFID) technology begins to permeate our lives (credit cards, passports, etc.), many in the community have voiced their concerns about the security of such devices. An attacker can, with little difficulty, skim the data entrusted to these devices from a distance of tens of meters with little effort. Unlike with other networking technologies, however, no one has examined the challenges of building a device capable of such a feat. To address these issues, Ilan discussed the process of developing a portable, extended-range ISO 14443-A compliant RFID skimmer. Using a Texas Instruments RFID reader and one of two antennas (a 10x15 cm printed PCB antenna and a 39 cm copper-tube antenna), the researchers were able to develop a leaching device capable of capturing RFID data from a distance of 35 cm, or approximately 3.5 times the advertised operational range. At approximately US$100, such a mechanism is easily within the budget of any dedicated adversary.

Many of the questions addressed the use of a loop antenna, which inherently has poor range. Additionally, because of the size of the copper-tubing antenna, many in the audience wondered about the practicality of using such a device without it being noticed. In its current form, Ilan argued, the device would in fact be extremely effective if the attacker could hide it well before the time of attack (e.g., behind drywall or around a potted plant). Ilan then concluded by mentioning that although the current work was simply a proof of concept, better antenna technology could easily be applied for a modest increase in cost.

■ *Keyboards and Covert Channels*

Gaurav Shah, Andres Molina, and Matt Blaze, University of Pennsylvania

Awarded Best Student Paper

Gaurav Shah pointed out that there are uncountable ways to extract sensitive data from a targeted machine. Most of these techniques, from "shoulder-surfing" to the installation of spyware, are easily detectable given current techniques. Shah discussed the JitterBug, a PS/2 keyboard attachment designed to capture such information. Unlike previous hardware schemes, which suffer from problems in recovery, the JitterBug delivers compromised data through an interpacket covert channel. Based on the time between two packets, an adversary located somewhere between the legitimate sender and the receiver can decode single bits of information without arousing suspicion on either end of the communication. Guarav stressed that such attacks are not limited to password stealing. By placing such hardware into machines prior to their distribution, adversaries from rival corporations to watchful governments can easily establish surveillance over a number of targets.

When asked how the JitterBug determined which data to transmit, Guarav discussed the use of trigger sequences. For example, attacks targeting user passwords can be activated after "ssh hostname.domain" is typed into the keyboard. Others asked about the use of more robust TCP covert channel encoding methods that have been implemented in software. Guarav agreed that such encoding mechanisms could be incorporated into the JitterBug and that the use of interpacket spacing was sufficient for a proof of concept. Guarav concluded with a discussion on the need for error correction and repeated messages in order to account for network jitter and uncertainty. By publishing this work, the authors hoped to draw attention to the need for all devices to be part of the trusted computing base in order for real security to be attained.

■ ***Lessons from the Sony CD DRM Episode***

J. Alex Halderman and Edward W. Felten, Princeton University

When Sony, the world's second largest music company, deployed digital rights management (DRM) protection for its music, it also released a number of critical vulnerabilities into the digital landscape. J. Alex Halderman discussed how he and Edward Felten of Princeton University logged the events as the first major rollout of mandatory DRM software occurred. The software, made by First4Internet and SunnComm, works by automatically installing itself on the first insertion of a CD-ROM. Both products installed themselves as rootkits to each client's machine, in order to prevent their subsequent uninstallation. Unfortunately, because of vulnerabilities in both products, other malicious programs could then exploit the DRM software to gain root control. Sony initially refused to offer uninstallation software, even when presented with the weaknesses discovered by numerous researchers. However, after numerous class action lawsuits and much public duress, the music distributor eventually provided the tools necessary to rid infected machines of these rootkits.

Questioning by the audience was limited, focusing largely on the response from other members of industry. For example, a number of attendees were interested in whether or not the software by First4Internet and SunnComm has been classified as spyware by antivirus companies. The speaker said that he believed that all major antivirus manufacturers in fact now classify this software as such. Halderman then discussed how the elimination of enabling mechanisms such as AutoRun would go a long way to prevent a repeat of such an incident. In the end, this work highlights the conflict between entities attempting to protect their intellectual property and legitimate users of that content. In an attempt to prevent "unapproved" use of their music, Sony in fact broke into and endangered its clients' PCs. Because the use of DRM is effectively an attempt to undermine a user's control of his or her own computing apparatus, its use is fundamentally a security issue.

SOFTWARE

Summarized by Lionel Litty

■ ***Milk or Wine: Does Software Security Improve with Age?***

Andy Ozment and Stuart E. Schechter, MIT Lincoln Laboratory

Andy Ozment presented the results of a study that examined whether the reporting rate of security vulnerabilities decreases over time. An earlier study by Eric Rescorla had found no evidence that this was the case for the four operating systems he studied. The authors of the study found otherwise by scrutinizing OpenBSD over a period of 7.5 years, starting with version 2.3, the "foundational" version for this study. They carefully examined the 140 security advisories released over that period of time, as well as the OpenBSD source code repository, to determine exactly when a vulnerability was introduced in the source code and when it was fixed. In addition, they estimated the amount of new code introduced by each release of OpenBSD to find out whether there was a correlation with the number of vulnerabilities that were introduced in a release.

Andy reported that a majority of the vulnerabilities found during the past 7.5 years were already present in the foundational version and that the median lifetime of those vulnerabilities was at least 2.6 years. In addition, he showed that the rate at which vulnerabilities were discovered in the foundational version decreased over time, with only half as many vulnerabilities reported during the second half of the time period. Using a reliability growth model, the authors also estimated that 42 vulnerabilities remained in the foundational version. Andy concluded by saying that they found no clear correlation between number of lines of code added between versions and number of new vulnerabilities, and that the OpenBSD source code was indeed like wine.

When asked whether he thought the findings would extend to other operating systems that may have less of an emphasis on security, Andy replied that his stab in the dark was that it may, but that the decrease rate was probably slower. Theo de Raadt asked (via IRC and Dug Song) whether the study took into account the mitigation mechanisms, such as having a nonexecutable stack, added by Open-

BSD since the foundational version. Andy said that they had not and that taking them into account would make the picture look even better for OpenBSD.

■ *N-Variant Systems: A Secretless Framework for Security Through Diversity*

Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser, University of Virginia

Artificial diversity consists of introducing differences in the way software is executed on a machine, for example by randomizing the memory layout. This may foil an attacker who is trying to take control of the machine, as long as the attacker does not know the key used to randomize the memory layout. Benjamin Cox described a new technique that would eliminate this reliance on a secret by running several variants of the same process in parallel, resulting in provable security. For all benign inputs, the variants would behave identically, but for some attacks, the variants would diverge. A monitor would detect the divergence and raise an alarm.

Benjamin discussed two ways to introduce diversity between the variants: partitioning the address space and tagging instructions. When partitioning the address space, the two variants use distinct memory addresses. If an attack relies on the absolute memory address of either code or data, it will cause a memory fault in at least one of the variants. Instruction set tagging adds a tag to each instruction. This tag is checked and removed by a dynamic binary translator before the code is run. Any code injected by the attacker will have an incorrect tag for at least one of the variants, once again raising an alarm. Depending on the workload and the diversification technique used, overhead for the prototype ranges from 17% to more than double the execution time for Apache.

Ron Jackson questioned whether security was provable for the address space partitioning scheme. He described an attack consisting of only overwriting some of the bits of an address, allowing the attack to be successful in both variants. Benjamin answered that this attack simply fell outside the class of attacks prevented by address space partitioning. Other limitations of the current prototype, left for future work, included handling signals and shared memory.

■ *Taint-Enhanced Policy Enforcement: A Practical Approach to Defeat a Wide Range of Attacks*

Wei Xu, Sandeep Bhatkar, and R. Sekar, Stony Brook University

Wei Xu observed that before performing a security-sensitive operation, an application should check that this operation is not under the control of an attacker. This can be done by tracking what parts of an operation are tainted, i.e., originating from untrusted interfaces. Data in memory originating from the network is marked as tainted and taint is then propagated throughout the execution of the application. To this end, the source code of the application needs to be automatically instrumented to maintain a bitmap of tainted memory locations.

When a security-sensitive operation is performed, a policy-defined check is conducted to make sure the operation is safe. For instance, an application may construct a SQL query based on user input. To prevent SQL injections, a check ensures that only data fields in the query are under user control. Wei suggested that a policy ensuring that no two consecutive tokens are tainted would achieve that goal. Similar policies can be used to defeat other types of attacks, such as cross-site scripting, path-traversal attacks, and command injections. These policies are enabled through the availability of fine-grained taint information. Various low-level optimizations allow the performance overhead of the approach to be below 10% for server applications.

Asked whether one could apply the same approach directly to binaries, thus suppressing the requirement that source code be available, Wei explained that it may be possible but would almost certainly preclude the optimizations they used to keep the performance overhead acceptable. Anil Somayaji asked whether hardware support could help improve performance. Wei answered that it was a possibility that had not been fully explored yet.

INVITED TALK

■ *Signaling Vulnerabilities in Wiretapping Systems*

Matt Blaze, University of Pennsylvania

Summarized by Patrick Traynor

With a recent push for the compliance of VoIP systems with the Communications Assistance for Law Enforcement Act (CALEA), many in the community have focused on the technical hurdles of this proposal. In an attempt to understand the security and reliability of these new eavesdropping systems, Matt Blaze and a team of University of Pennsylvania researchers have examined the inner workings of traditional telecommunications surveillance technology. As part of the Trustworthy Network Eavesdropping and Countermeasures (TNEC) project, this group explores the challenges behind building both surveillance-friendly and resistant networks. Most important, this work asks the question, "Assuming properly functioning tools, is wiretap evidence trustworthy?"

There are a number of ways in which telecommunications traffic can be intercepted. If direct access to the targeted phone is possible, the eavesdropping party can gain control of the two wires connecting the phone to the network (known as the local loop). Such methods are prone to discovery by the monitored party and are therefore not typically used in law enforcement. This work instead focuses on the use of the loop extender, a wiretapping device placed between the targeted phone and a network switch that allows for monitoring in both traffic analysis ("pen register") and full content modes. The loop extender works by redirecting a copy of the content from the "target line" to recording devices on a "friendly line." Because both traffic analysis and content are always sent to the friendly line, the actual information recorded is set via configuration at the recording device.

Blaze and his team began by examining vulnerabilities in the pen register mode. In order to determine which number a target is calling, the recording equipment decodes the series of unique audio tones sent across the line. In order to accommodate a wide range and quality level of products, both the network switching equipment and eavesdropping tools accept a range of analog tones. The ranges accepted by the devices, however, are not the same. Accordingly, Blaze was able to show that sending tones just outside the range of the telecommunications switch (either above or below) but within the range of the loop extender allowed the party being eavesdropped upon to forge the list of dialed numbers recorded in pen register mode.

Blaze then discussed subtle vulnerabilities in the audio recording mechanism used for eavesdropping. The taping of audio content automatically begins at the cessation of the "C-tone," an audio tone transmitted across a phone line when it is not in use. Accordingly, when the C-tone is again detected, the recording equipment automatically shuts itself off. To prevent sensitive material from being recorded, a targeted party can simply play the C-tone continuously into the receiver. This technique is successful even when the C-tone is played at 1/1000 of its normal volume.

The new CALEA standards fix many of the problems associated with loop extender wiretaps. Eavesdropping now occurs at the switch itself and signaling has been separated onto a separate channel. Unfortunately, many vendors continue to offer systems that are compliant to the C-tone shutoff mechanism. Depending on the configuration of these new CALEA-compliant devices, modern eavesdropping systems may also be susceptible to the vulnerabilities discovered for loop extender systems.

The attendees of the talk asked Matt a wide variety of questions. Because of the illegality of wiretapping equipment, many were curious as to how this group was able to perform their work without fear of legal recourse. Matt mentioned that because this research was part of an NSF grant devoted to wiretapping technology, they had implied consent to carry it out. Others were curious about testing the local loop by the phone companies. Matt said that it was indeed possible for the phone company to detect the deviation of frequencies from the accepted band but that the wide range of quality across user devices means that such divergence from the standard may only be the result of poor construction. Matt's talk concluded with a discussion of the practical difficulties facing IP wiretapping efforts.

NETWORK SECURITY

Summarized by Wei Xu

■ *SANE: A Protection Architecture for Enterprise Networks*

Martin Casado and Tal Garfinkel, Stanford University; Aditya Akella, Carnegie Mellon University; Michael J. Freedman, Dan Boneh, and Nick McKeown, Stanford University

Martin Casado started his talk by pointing out that traditional techniques for putting access control onto enterprise networks are associated with many problems, such as inflexibility, loss of redundancy, and difficulty in management.

To address these limitations, Martin proposed SANE, a Secure Architecture for the Networked Enterprise. SANE introduces an isolation layer between the network layer and the datalink layer to govern all connectivity within the enterprise. All network entities (such as hosts, switches, and users) in SANE are authenticated. Access to network services is granted in the form of capabilities (encrypted source routes) by a logically centralized server (Domain Controller) according to high-level declarative access control policies. Each capability is checked at every step in the network.

Martin also presented several important details in SANE (such as connectivity to the DC, establishing shared keys, and establishing topology), as well as the countermeasures that SANE offers for attack resistance and containment. He also mentioned that their prototype implementation showed that SANE could be deployed in current networks with only a few modifications.

■ ***PHAS: A Prefix Hijack Alert System***

Mohit Lad, University of California, Los Angeles; Dan Massey, Colorado State University; Dan Pei, AT&T Labs—Research; Yiguo Wu, University of California, Los Angeles; Beichuan Zhang, University of Arizona; Lixia Zhang, University of California, Los Angeles

In this talk, Mohit Lad first briefly introduced the BGP (Border Gateway Protocol) and then illustrated the widely reported BGP prefix hijack attack, in which a router originates a route to a prefix but does not provide data delivery to the actual prefix.

After that, Mohit presented a Prefix Hijack Alert System (PHAS). The objective of PHAS is to provide reliable and timely notification to prefix owners when their BGP origin changes, so that prefix owners can quickly and easily detect prefix hijacking events and take prompt action to address the problem. PHAS uses BGP data collectors (especially RouteViews and RIPE) to observe the BGP updates. The updates are then provided to the origin monitor, which detects the origin changes and delivers email notifications to the affected prefix owners through a multipath delivery. According to Mohit, PHAS is lightweight and readily deployable.

■ ***Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting***

Jason Franklin, Carnegie Mellon University; Damon McCoy, University of Colorado, Boulder; Parisa Tabriz, University of Illinois, Urbana-Champaign; Vicentiu Neagoe, University of California, Davis; Jamie Van Randwyk, Sandia National Laboratories; Douglas Sicker, University of Colorado, Boulder; Scott Shenker, University of California, Berkeley

Parisa Tabriz gave the first part of this talk, in which she explained the motivation for their work by arguing that the emerging driver-specific exploits were particularly serious for the 802.11 wireless devices because of their wide deployment and external accessibility. Device driver fingerprinting can be of help for an attacker to launch a driver-specific attack.

Damon McCoy then presented the details of their passive fingerprinting technique, which identifies the wireless device driver running on an IEEE 802.11 compliant device. Their technique is based on the observation that many of the details of the active scanning process are not fully specified in the IEEE 802.11 standard, and hence they are determined by wireless driver authors. As a result, drivers can be distinguished by their unique active scanning patterns. Damon said that they had generated signatures for 17 different wireless drivers, and their evaluation showed that this fingerprinting technique could quickly and accurately fingerprint wireless device drivers in real-world wireless network conditions. Finally, Damon discussed several possible ways to prevent the 802.11 wireless driver fingerprinting.

When asked whether the fingerprint of a wireless driver would change in different operating systems or when the driver interacted with different access points, Damon and Parisa answered that the active scanning behavior should depend only on the driver implementation, but they had not looked into these particular problems.

INVITED TALK

■ ***Turing Around the Security Problem: Why Does Security Still Suck?***

Crispin Cowan, SUSE Linux

Summarized by Bryan D. Payne

Crispin Cowan began this invited talk saying that "security sucks" more than any other aspect of computing. He backed up this statement by showing how Turing's theorem can be used to show that security is an undecidable problem. In addition, security is harder than correctness. Although correctness is important, security increases programmer burden, since attackers can produce arbitrary input. History has shown that neither money nor even diligent corporate practices are sufficient to solve the problem.

Crispin explained that one approach to security is to use heuristics. This is seen, for example, in static analyzers. The problem with heuristics is that they are not perfect. Heuristics cannot analyze all programs, and they provide imperfect results in other applications.

As an alternative, Crispin suggests that security professionals use the OODA Loop. Colonel John Boyd (USAF) invented the idea of an OODA Loop. OODA stands for Observation, Orientation, Decision, and Action. The OODA Loop provides a pattern for fighter pilots to use in decision-making. In general, the person with the fastest OODA Loop will win a battle. Crispin spent a large portion of the talk mapping the process of security to the OODA Loop.

The OODA Loop can be mapped to the three critical questions: when, where, what. Crispin walked through many security tools and concepts, mapping each to one of these three questions. He covered topics such as Saltzer and Schroeder's principles of secure design, syntax checkers, semantic checkers, type-safe languages, kernel enhancements, intrusion detection, intrusion prevention, network access controls, host access controls, capabilities, and more.

In closing, Crispin changed gears and spoke about AppArmor, SUSE's approach to improving application security. Crispin described the fundamental difference between AppArmor and SELinux: AppArmor uses path names and SELinux uses labels. He also acknowledged that AppArmor is a work in progress; some types of protection are not currently possible.

In the Q&A session, Pete Loscocco, a leader of the SELinux project, suggested that the differences between SELinux and AppArmor are more fundamental than Crispin stated because SELinux tries to be comprehensive whereas AppArmor has less control. Crispin acknowledged that there are trade-offs and suggested that SELinux achieves higher security but also has a higher cost. Other people asked questions about practical security issues. Why aren't companies doing more about security? Crispin believes that they are secure enough for their purposes. What's being done to address the security problems today? Crispin pointed to better programming languages and various forms of isolation. How does one properly extend security from the kernel into the applications? Crispin suggested using discrete programs that cooperate (e.g., Postfix).

PANEL

Major Security Blunders of the Past 30 Years

Matt Blaze, University of Pennsylvania; Virgil Gligor, University of Maryland; Peter Neumann, SRI International Computer Science Laboratory; Richard Kemmerer, University of California, Santa Barbara

Summarized by Madhukar Anand

Matt Blaze: We are going to have more damage because of excess attention to security. For instance, consider an alarm system. It is designed such that the alarm is triggered as quickly as possible. Consequently, any attack that aims at not setting off the alarm system is entering into a arms race with the alarm sensor design. Although the alarm system could win such an arms race, it must be noted that the main objective of the attacker is not to defeat the sensor system but to break the system. A simple strategy of setting off the alarm repeatedly would do the trick. The police might be tricked into losing faith in the alarm system.

Another example is the signaling in telephone systems. The objective of the attacker here is to defeat the billing system and make free long distance calls or call unauthorized phone numbers. The protocols are designed by people who did not know they were designing a security function. The only people who are motivated to break into such a system are the bad guys.

The Telnet protocol was designed with an option for encryption (DES 56-bit key in a 64-bit package). The newer library, to be more compliant with the DES standard, had 8 bits designated as the checksum. If the checksum would not tally, then the key would be all 0s. Because of this, if there were bit errors, there was a 1/256 chance of using encryption. In 255/256 cases, the key would be all 0s. So, this is an counterexample to the principle "Good code is secure code." Code that checks secure values in this case indeed made it less secure.

Virgil Gligor: Blunder 1: Morris worm—Buffer overflow in fingerd. This was the first tangible exploit of a buffer overflow, the first large-scale infection engine, and the first large-scale DDoS attack. Although there were no lessons learned for a long time, it did result in the creation of CERT. It also led to the realization that any new technology introduces new vulnerability and vulnerability removal takes 6 to 12 years, creating a security gap. Also, we learned that "security is a fundamental concern of secondary importance."

Blunder 2: Multilevel Secure Operating System (MLS-OS). This was pushed by the defense establishments in the United States and Europe. However, there was no market for MLS-OSes as they break off-the-shelf applications and are often hard to use. In fact, the stronger the MLS-OS, the worse the system (e.g., B2 secure Xenix broke all our games). This blunder largely drained most security funding and was a major R&D distraction.

Blunder 3: Failed attempts at fast authenticated encryption (in 1-pass -1 cryptographic primitive.) This blunder was largely academic. Starting with CBC+CRC-32 systems in 1977 to NSA dual counter mode + XOR, each successive system was built only to be broken later. Together, they constitute the largest sequence of cryptographic blunders. The upshot of this is the realization that some of the simplest cryptographic problems are hard. The lesson to be learned is to stick to basic system security research.

Richard Kemmerer: Blunder 1: U.S. export controls of cryptography. These controls were based on international traffic in arms regulations. However, they were very difficult to enforce. The lessons learned from the export control episode are that they make sense but change constantly and that the enforcers have no sense of humor—a case in point being the Verification Assessment study (Kemmerer, 1986), which was published with the warning that its export is restricted by the arms export control act. This hampered the publication of the study and resulted in many copies being stocked and not reaching their readership.

Blunder 2: Kryptonite Evolution 2000 U-Lock. Although the Kryptonite was advertised as the "toughest bike lock," all it took to break it was a Bic pen. After cutting small slits in the end of the pen's barrel to ease it in, the lock opened with a single twist.

Blunder 3: Australian Raw Sewage Dump in March 2000. Vitek Boden from Brisbane, Australia, hacked into a local waste management computer system and directed raw sewage into local rivers, into parks, and even onto the landscaping of a Hyatt Regency hotel. Boden had previously been fired from the company that installed the waste management system, and he vandalized the public waterways as an act of revenge. The lessons learned from this were to pay attention to security in SCADA systems and to insider threats. In conclusion, "Beware of gorillas invading your system while you are counting basketball passes," which was depicted by showing a video where a person in a gorilla suit walks through the scene but is unnoticed by most of the attendees until the second showing of the video.

Peter Neumann: The big picture is that security is complex, it is a end-to-end system emergent property, and there is a weakness in depth. We do not really learn from experience, as some of the blunders keep recurring. Security is holistic. So, we should not consider it in isolation. For instance, application security is easily undermined by OS security.

Propagation Blunders: 1980 ARPANET collapse. The collapse, resulting from a single point failure in one node, ended up bringing down the entire network. In 1990, there was a half-day when AT&T long lines collapsed, again as a result of a single point of failure. Yet another example of propagation blunders are the repeated power outages from 1967 to August 2003. In some sense, we don't design our infrastructures and our systems to withstand things as simple as a single point failure. The standard answer is, "This can never happen."

Backup and Recovery Blunders (e.g., air traffic backup and recovery failures): In July 2006, there was a power outage in Palmdale, CA, including the air traffic control center. The center automatically switched to backup diesel generators. They worked for about an hour but then the system that switches the center between commercial and backup power failed and the building went dark. In 1991, three New York airports were closed as backup batteries were accidentally drained. Other examples of such blunders include the Swedish train reservation system failure and the Japanese stock exchange system failure in November 2005. There have been blunders where there was no backup, such as when the New York Public Library lost all its references.

Software Blunders: There have been many types of buffer overflows. Programming language research could help reduce these, but they could also prove to be a hindrance. In fact, Multics was mostly free of buffer overflow vulnerabilities, owing to the use of PL/I as the implementation language. PL/I required an explicit declaration of the length of all strings.

Election Systems Blunder: The requirements of an electronic voting system include end-to-end reliability, integrity, and accountability. Therefore, Help America Vote (HAVA) was a huge blunder. Because none of the electronic paperless systems were auditable, they lack integrity.

The lessons learned are that individual cases may be less important than the fact that we see the same types of problems. We need a massive cultural change in how we develop software.

More resources on risks and trustworthy architectures can be found at http://www.csl.sri.com/users/neumann/chats4.html.

INVITED TALK

■ *Aspect-Oriented Programming: Radical Research in Modularity*

Gregor Kiczales, University of British Columbia

Summarized by Kevin Butler

Gregor Kiczales gave a practical, code-oriented talk that provided an introduction to aspect-oriented programming (AOP). Kiczales and his team originated the idea of AOP while he was at PARC. The talk focused on AOP and what it is, what makes it interesting, and how software will be different with AOP compared to what has come before. The focus was on the Aspect/J programming language, a seamless extension to Java that is fully supported by the Eclipse open source project and is a model for other AOP tools.

One of the biggest benefits of AOP is its ability to aid code expressiveness: The code looks like the design, and what is going on in terms of operations is clear. As an example, Kiczales pointed to a Java program called jHotDraw which allows joining of points and lines, setting colors, and other graphical tasks. This can be designed and implemented with object-oriented programming, where shapes become classes. There are also objects relating to the display of the shapes and to perform update signaling (e.g., when shapes change). The goals for any programming paradigm are expressiveness, modularity, and abstraction. For jHotDraw, object-oriented programming allows good support of these goals for shapes and their display, but it is not good for update signaling. The structure of signaling is not localized, clear, or declarative, and the resulting solution is neither modular nor abstract. Signaling is clearly not localized as it cuts across multiple objects. With AOP, it is possible to provide the three goals set out.

Some of the terminology specific to AOP includes the idea of *join points*, which are defined points in a program flow (Aspect/J allows for dynamic join points) and *pointcuts*, a set of join points on which a predicate is based that may or may not match. Pointcuts are composed like predicates, and a set of primitive pointcuts are defined in Aspect/J. When a pointcut is reached, a specified piece of code, called *advice*, is run. To go back to the drawing example, there is an observer pattern aspect of the system, and some points in the system's execution are changed; after returning for a change, the display is updated. Updated calls would be scattered and tangled with object-oriented programming; by contrast, with AOP, values at the join points are defined, a pointcut can explicitly expose certain values, and advice can use these explicitly exposed values. The key to AOP is its ability to deal with *crosscutting* across multiple concerns. For example, a pointcut in a model can specify a slice of a different part of the world to provide an interface. This could be particularly pertinent to industry: One can walk up to a big system, drop pointcuts in it, and program against the system even if it doesn't have the structure the programmer was expecting. In response to questions about this, Kiczales explained that one of the fundamental differences with AOP is that there is no need to explicitly signal events as is necessary with OOP. For example, one could take a million lines of already written code and do pointcuts without even having the source code for the original system. When asked whether the goal of AOP was to be able to retrofit features into legacy code, Kiczales was circumspect and careful in his answer. The biggest goal of AOP is to modularize crosscutting concerns, which sometimes can be done for legacy code and sometimes not. Oftentimes, new features are crosscutting concerns.

Kiczales expressed some thoughts as to whether going further than Aspect/J could be feasible. Referencing the book *On the Origin of Objects* by Ryan Smith (where objects refer to actual real-world things, not class instances), he talked about perception and how we see the world in different ways (e.g., a glass half empty vs. half full), and how do we both grab the same thing. Registration is the process of parsing objects out of the fog of undifferentiated "stuff," and we are constantly registering and reregistering the world. We also possess the ability to process critically: We have multiple routes of reference, such as the ability to exceed causal reach (e.g., describing someone as closest to average height in Gorbachev's office) or indexical reference (e.g., the one in front of him); allowing this sort of expressiveness into programming could be similarly useful. Join point models can decompose software into different ways and atomize into the fog of undifferentiated points, with connections and effects made through registration. All could have causal access to the same point but access it in different ways, given the multiple routes to reference.

Much of the Q&A session focused on the security aspects of AOP. For example, if AOP or Aspect/J can touch a JAR file, then all bets are off. Audit logging is a clear example of where AOP is useful, but what other security uses are there? Is ACL checking a potentially good use? Kiczales responded that his group intentionally did not touch on the uses for security, leaving that to security professionals, but intimated that there were certainly security problems that could best be attacked with AOP. Gary McGraw commented that this logic could be taken only so far: If something is a security feature, it can probably be done as an aspect, but security is not just a bunch of features. Other questions included dealing with aspect clashes, to which the best defense is good software engineering and a quality codebase; how to debug code, which has improved greatly as Aspect/J has matured; and the problems of retrofitting legacy code with access control hooks, which is nontrivial. Kiczales responded to this by commenting that no sane person claims that one should be able to take a system and add modular implementations of crosscutting without having to jigger the source code a little bit, particularly when dealing with the domain functionality of a system. However, refactoring and some mild editing of the code could make all the difference.

STATIC ANALYSIS FOR SECURITY

Summarized by Lionel Litty

- ***Static Detection of Security Vulnerabilities in Scripting Languages***

Yichen Xie and Alex Aiken, Stanford University

Alex Aiken observed that Web applications are increasingly popular and that they present application-level vulnerabilities that are hard to prevent using low-level security mechanisms. Instead, static analysis may prove valuable when hunting for security bugs in such applications. He reported that his group was successful in performing static analysis of a scripting language (PHP) to find SQL injection vulnerabilities, despite the difficulties inherent in analyzing scripting languages: dynamic typing, implicit casts, weak scoping, and extensive use of string manipulation functions and hash tables.

To scale to large applications without making the analysis too imprecise to detect bugs reliably, three levels of abstraction were used: intrablock level, intraprocedural level, and interprocedural level. The result of the symbolic simulation of each code block is summarized before performing the analysis of each procedure. The result of this analysis is in turn summarized to perform interprocedural analysis. Each summary captures exactly the information necessary to perform the next step of the analysis, which is the key to scaling. Applying this technique to six large applications resulted in the discovery of 105 confirmed vulnerabilities with only a small number of spurious warnings.

Christopher Kruegel asked how the tool dealt with variable aliasing. Alex answered that the tool did not perform sound tracking of aliasing outside of the block level and that this was one way in which the tool was not a verification tool, meaning that it will not guarantee the absence of vulnerabilities. He also highlighted that programmers often do not use the full power of a programming language. This enables the tool to perform well despite making a number of simplifications.

- ***Rule-Based Static Analysis of Network Protocol Implementations***

Octavian Udrea, Cristian Lumezanu, and Jeffrey S. Foster, University of Maryland

Octavian Udrea argued that whereas formal methods have been used extensively to check network protocol designs, the actual implementations of these protocols often do not get the same level of attention. He described a tool called Pistachio that addresses this situation by focusing on static verification of a protocol implementation against a high-level specification of the protocol. This manually constructed specification, derived from documentation describing the protocol, consists of a set of rules that the protocol must adhere to.

With the help of a theorem prover, execution of the implementation is then simulated. Developing rules for the SSH and RCP protocols took the authors only seven hours, although these rules did not cover all aspects of the protocols. Implementations of these protocols were then analyzed, and rule violations were found in the LSH implementation of SSH and the Cygwin version of RCP. These violations were checked against a database of known bugs. Of the warnings produced by Pistachio, 38% were spurious. Pistachio failed to find 5% of known bugs. More information is available at http://www.cs.umd.edu/~udrea/Pistachio.html.

David Wagner asked how the authors were able to estimate the false-negative rate. Octavian explained that this number corresponded to known implementation bugs that Pistachio did not find. This number might be inflated by still unknown bugs in the implementation.

- ***Evaluating SFI for a CISC Architecture***

Stephen McCamant, Massachusetts Institute of Technology; Greg Morrisett, Harvard University

Awarded Best Paper

Software-based Fault Isolation (SFI) is a sandboxing technique used to confine the effects of running a piece of code from the rest of the system by inserting checks before every memory write and every jump. While the technique has been shown to work for RISC architectures, Stephen McCamant showed that it could be applied to a CISC architecture, the IA-32 architecture, as well. The key challenges in building a prototype, PittSFIeld, were the variable length of the IA-32 instructions and their lack of alignment constraints, potentially allowing complex code obfuscation.

These challenges were addressed by forcing instruction alignment via assembly code rewriting and by adding padding. In addition, the authors implemented new optimization techniques and carefully analyzed the performance overhead introduced by SFI, which is a 21% slowdown on the SPEC benchmark. One reason for the slowdown is increased cache pressure caused by the 75% increase in size of the binary code as a result of padding. Finally, Stephen emphasized that the security of PittSFIeld relies on a small verifier. The authors formally proved, using the ACL2 theorem proving system, that the verifier checks ensure confinement. More details can be found at http://pag.csail.mit.edu/~smcc/projects/pittsfield.

The audience asked whether the same technique could be performed directly on the assembly code. Stephen answered that it might be possible with a very good disassembler or if additional symbol information is available. Stephen was also asked the difference between SFI and Control Flow Integrity (CFI). He said that the two approaches were very similar. SFI provides memory isolation, but this can be added to CFI as well. Timothy Fraser mentioned that he knew of failed efforts to implement SFI for IA-32 and commended the authors.

INVITED TALK

Surviving Moore's Law: Security, AI, and Last Mover Advantage

Paul Kocher, Cryptography Research

Summarized by Micah Sherr

Paul Kocher's talk was divided into two parts. In the first half, he explored how complexity can be used to secure systems. He began by examining the effect of Moore's Law on cryptography. On the one hand, increased CPU performance seems to favor the cryptographer. For example, moving from DES to 3DES requires only three times the CPU cost, at the same time requiring an exponential increase in work by the cryptanalyst. However, as Paul points out, systems are becoming more complex, and added complexity (and lines of source code) raises the number of potential vulnerabilities dramatically. The growth of human intelligence does not necessarily match the growth of system complexity. There is therefore a need to harness complexity to improve security rather than hinder it.

Paul proposes increasing complexity by adding components to systems and connecting those components in a stream. The output of the system is the combination (e.g., XOR) of all of the components' outputs. If one system fails or is compromised, the overall security of the system does not necessarily falter. Generally, he advocates adding "micro CPUs"—independent and isolated execution areas that are specialized for different operations. The micro CPUs communicate with each other to produce the final result.

In the second half of the talk, Paul described the current state of security as a tug-of-war between those who protect systems and those who attack them. Paul believes that companies no longer believe that security is binary and that they have adopted the strategy of minimizing (rather than preventing) the amount of harm done to their systems. The major defense mechanism is the patch, which is applied after an attack, but which hopefully reduces the possibility of further attack. On the flip side, the attacker attempts to exploit new and unpatched vulnerabilities. This leads to a stalemate in which attacks are closely followed by fixes. According to Paul, this stalemate is the best-case scenario for the defender: Playing for a stalemate requires less devotion than comprehensive security fixes, stalemates allow defenders to place the blame on others (those who haven't applied the patches), and developing patches is easier than in-house security testing. However, playing for a stalemate leaves the defender with nonoptimal security.

Paul concludes by considering the future of this tug-of-war between defender and attacker. To some degree, generation of attacks (exploits) and defenses (patches) can be automated. At the same time, the undecidability of certain problems means that there are limits to what automation can accomplish. To some degree, the human element will always be involved in the automation process.

INTRUSION DETECTION

Summarized by Michael Locasto

SigFree: A Signature-Free Buffer Overflow Attack Blocker

Xinran Wang, Chi-Chun Pan, Peng Liu, and Sencun Zhu, The Pennsylvania State University

Xinran Wang began by illustrating the problems with current approaches to blocking buffer-overflow based attacks. Such attacks can be previously unseen or potentially delivered by polymorphic malware. Remote buffer overflows are typically driven by network input containing the exploit code, and Xinran and his coauthor's key insight is that detecting the presence of legal binary code in network flows was one way to recognize such attacks without having to resort to a signature-based scheme.

Xinran described two techniques their Instruction Sequence Analyzer (ISA) uses to build an extended instruction flow graph (EIFG), a construct similar to a control flow graph. Since x86 is a very dense instruction set, instruction sequences can be derived from most binary strings, and the authors propose building an EIFG from the network traffic. Intuitively, the larger the EIFG, the more likely it is that a certain binary string is a working x86 program. The first technique for building an EIFG employs pattern detection to provide a frame of reference for the ISA—it detects the push-call instruction sequences that represent a system call.

Xinran outlined a more sophisticated method (Data Flow Anomaly), which is based on detecting abnormal operations on variables. The motivation for this technique is that a random instruction sequence is full of these types of data flow anomalies, but a useful x86 program (such as one provided by an attacker) is not. This latter technique is also more resistant to polymorphic attacks, since the malware decoder does not necessarily need to make system calls, but still must include useful instructions. The authors report on fairly good results: No false positives were observed in a "normal" test set, and their techniques detected about 250 attacks and variations. They are looking forward to using a weighted scheme to frustrate attackers who may know the threshold of useful instructions for the DFA scheme. A follow-up question on how well this technique may apply beyond the x86 instruction set was tabled for offline discussion. An attendee from Cisco asked whether the code for the system would be publicly available, since this type of capability was of widespread interest, but Xinran indicated that the techniques were currently under patent review. Finally, another attendee from MIT asked about the nature of the "normal" requests that actually contained fairly long sequences of code. That discussion was taken offline.

■ ***Polymorphic Blending Attacks***

Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee, Georgia Institute of Technology

Prahlad discussed the design and implementation of a class of malware capable of dynamically adapting its payload to the characteristics of the surrounding network traffic in order to bypass content-based anomaly sensors such as PayL. The key idea is that the malware is able to sneak past the anomaly detector because purely statistical content anomaly detectors discard both the syntax and the semantics of the normal traffic content.

The malware has two capabilities that allow it to create a variant that will not trip the content anomaly detector. First, it is able to observe a portion of the target system's network traffic. Second, it knows the algorithm that the anomaly detector uses to construct a model. After creating an initial and approximate model of the traffic that the target system is receiving, the malware creates a variant of itself using shellcode encryption and padding to match the generated profile within some error bound. In their experiments using PayL as the content anomaly detector, the authors found that all they needed was 20 to 30 packets to learn the target profile. The authors closed by suggesting some possible countermeasures to the polymorphic blending attack, including the higher-cost techniques of actually parsing the packet data rather than using solely statistical methods. It may also be possible to use multiple independent models or to somehow randomize the implementation of the anomaly detector algorithm. At the end of the presentation, the attendee from Cisco reprised his question on the availability of the system, and Prahlad answered that they shared code with some partners of the lab. A second questioner asked for clarification as to the practicality of the first countermeasure (doing deep packet inspection). Prahlad answered that with appropriate support or in some types of low-traffic environments, such inspection would not be prohibitively expensive.

■ ***Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection***

Holger Dreger, Anja Feldmann, and Michael Mai, TU Munchen; Vern Paxson, ICSI/LBNL; Robin Sommer, ICSI

Holger raised the question of how network intrusion detection systems actually know which protocol decoder should be applied to the traffic that these systems observe. Most current systems simply rely on the transport layer port numbers in order to make this decision. This assumption does not hold for standard services that are run on nonstandard ports nor for standard or nonstandard services that are run on the "incorrect" port.

Holger then used some summary statistics of a large body of data to further explicate the problem. The data was a full packet capture over a 24-hour period, totaling some 3.2 TB, 137 million TCP connections, and 6.3 billion packets. The authors ran the Linux netfilter i7 protocol signatures on the traffic trace and focused on HTTP, IRC, FTP, and SMTP. They found that i7's matching was good for the normal case, that is, when a particular protocol appeared on a port it normally does. But a significant portion of the traffic defied classification or was incorrectly classified.

The key question the authors seek to address is the problem of distinguishing among network services without taking a hint from the port number. Holger presented the design and implementation of a modification to the Bro NIDS that performs dynamic analysis of the observed traffic: The system matches multiple possible protocols in parallel. The system provides the mechanism for doing so; whether or not to commit to a particular parsing decision is left as a matter of thresholding and policy.

Holger described a number of interesting results, including that most connections on nonstandard ports in their traffic trace were HTTP and that the protocols tunneled therein were mostly P2P services, some raw HTTP, and some FTP traffic. They were able to identify and close some open SMTP relays and HTTP servers that violated the site's security policy and have been able to detect some botnets at one of the author's sites. Holger closed by indicating that the system will be incorporated into Bro.

The first question focused on how difficult it would be to add a new protocol definition to the system. Holger indicated that it was a moderate amount of work, but Bro provides a great deal of infrastructure for doing this already. The remaining questions focused on the system's ability to handle tunneled traffic in various forms, in particular, how easy it would be for an attacker to craft input that prevented the parallel logic from making a choice between two protocols. Holger replied that the system provides the mechanism, and the policy to control it is up to the user or site administrator. Even so, he indicated that the combination of protocol signature matching and dynamic analysis somewhat alleviated this problem.

■ ***Behavior-Based Spyware Detection***

Egin Kirda and Christopher Kruegel, Technical University Vienna; Greg Banks, Giovanni Vigna, and Richard A. Kemmerer, University of California, Santa Barbara

Egin gave an engaging talk about the techniques used in real spyware systems and described the author's system for detecting this spyware. He began by pointing out that spyware is a burgeoning problem, and that it is difficult to identify this type of malware because it often comes in many different guises and is potentially installed by a trusted but unwary user. Signature-based spyware solutions are often on the losing end of an arms race.

However, spyware often cannot escape its main goal, and that goal is to gather information about a user's behavior and then transmit that information somewhere. This type of anomaly or behavior-based detection is a promising technique for detecting spyware that doesn't yet have a signature.

The authors focus on spyware for Internet Explorer, and Egin said that most spyware registers as a Browser Helper Object (BHO) or toolbar (for all practical purposes, a toolbar is no different from a BHO). One major reason for BHO's popularity as a spyware vehicle is that high-level user behavior and data are readily available to a BHO, thus greatly simplifying the implementation of the spyware component: It can simply use the events, data objects, and services that IE provides to all BHOs. Egin indicated that their techniques are of limited applicability for spyware that does not use COM services. For example, spyware may utilize other forms of communication such as issuing GET requests to URLs that are under the attacker's control.

Egin described their implementation of a "stub" IE instance that intercepts communication between the browser and BHOs. The system combines static and dynamic analysis to drive the training phase. The dynamic analysis records all "interesting" COM calls and provides the starting point for static analysis to construct a control flow graph to see whether information could have been leaked. For all of their 51 test samples (33 malicious/spyware, 18 benign), the false-negative rate was zero, but the false-positive rate only improves as the system moves toward the combination of static and dynamic analysis. One interesting result is that the privacy/P3P plugin acts like spyware, according to the authors' behavior characteristic: It reads a URL and then contacts the Web site. After the talk, an attendee asked whether or not it would be possible for a malicious BHO to detect that it was being run by the detector version of the browser and adjust its behavior accordingly. Egin said that this situation was similar to a program detecting whether it was running in a VM, and that the countermeasures taken by the spyware may be detectable, but it was certainly an avenue for future research.

INVITED TALK

■ ***DRM Wars: The Next Generation***

Ed Felten, Princeton University

Summarized by Tanya Bragin

For a number of years digital rights management (DRM) has been a hot topic in the press and a focus of many research and development efforts. But has there been any significant progress since the Digital Millennium Copyright Act (DMCA) of 1998 bolstered DRM by criminalizing any attempts to circumvent it? And after the public outrage over the Sony rootkit fiasco, what future awaits DRM? Ed Felten is uniquely positioned to answer these questions. He spent many years as a computer scientist in both industry and academia, but in recent years he has been increasingly focusing on legal and policy issues related to technology.

"'Rights Management' is somewhat of an Orwellian term," stated Ed Felten. "It implies that DRM advocates want to manage your rights, not control what you do." But in reality DRM software by the very nature of what it tries to accomplish must restrict users' ability to control their personal computers in order to prevent them from copying restricted content. When DMCA came out, it was considered reasonable for users to surrender some control in order to protect the artists' right to collect revenue for their work. However, it is unclear that technology can deliver on the DRM promise, so some people are starting to question why users have to hand over control of their machines to DRM software companies.

According to Ed, the main issue with the assumption that technology can actually solve the problem of protecting digital content is the "rip once, infringe everywhere" phenomenon, published in what became known as the "Darknet Paper" by Microsoft Research in 2002 (www.freedom-to-tinker.com/?p=206). Given the ease with which people can participate in peer-to-peer file sharing networks, it is only necessary for one person to defeat DRM in order for everyone else to have ready access to the unprotected content. This type of threat model is really hard to defeat and DRM advocates have gradually become convinced that this problem is intractable in practice.

With this realization, incentives for deploying DRM software have also changed. Originally DRM was supposed to provide antipiracy features that benefited artists and ensured compliance of digital content with the copyright law. However, given the apparent infeasibility of these goals, DRM software has come to serve other purposes. First, it allowed digital content distributors to perform *price discrimination*, which is enabled by DRM's ability to hinder transfer of content from user to user. Second, it allowed some organizations, such as Apple, to gain significant market share through *platform locking*. Apple's successful combination of iPod, iTunes, and the Apple Store has created an unrivaled digital music sale franchise.

It is unclear that these new rationales for using DRM continue to benefit artists. For example, Apple, publicly published an "interoperability workaround" that allows users to burn protected content to a CD and then rip that content from the CD in an unprotected format. Similarly, price discrimination primarily benefits digital content distributors, because it allows them to make a profit on products they otherwise could not afford to produce. Although price discrimination can also benefit artists and consumers, since it allows certain types of products to come to market, arguably it can be achieved using techniques other than DRM.

Ed expects to see more efforts by companies to use DRM for self-serving purposes, while continuing to claim that antipiracy is the ultimate goal. Ed warned audience members to look out for these troubling developments, since they are likely to hinder interoperability, complicate products, and frustrate consumers. In Ed's opinion, DRM policy should be neutral, neither discouraging nor bolstering its use. Given the "inherent clumsiness of DRM," he believes that market forces would ultimately decide against it. For more information about DRM and other legal and policy issues related to technology, visit Ed Felten's blog, "Freedom to Tinker," at www.freedom-to-tinker.com.

SYSTEM ASSURANCE

Summarized by Bryan D. Payne

■ ***An Architecture for Specification-Based Detection of Semantic Integrity Violations in Kernel Dynamic Data***

Nick L. Petroni, Jr., and Timothy Fraser, University of Maryland; AAron Walters, Purdue University; William A. Arbaugh, University of Maryland

Current systems design makes the kernel a high-value target for attackers. Defenses such as load-time attestation and run-time attestation are valuable, but these are only capable of detecting changes to static data. Nick Petroni described how attacks can exploit this limitation to avoid detection, using an example of manipulating dynamic data structures to hide a process in Linux.

The proposed solution is to create a high-level model of the dynamic data and to use this model to validate the data. This approach, which was inspired by work from Demsky and Rinard, uses a data structure specification language to model constraints on the data. Nick showed a simple example that would catch the previously mentioned attack. The technique has some limitations, including the possibility of missing short-lived changes and the need for a kernel expert to model the constraints.

In the Q&A session, the session chair asked whether people would actually use a parallel language such as this. Nick believes that they would and that companies could provide the specification for binary operating systems such as Windows or Red Hat Linux.

■ ***vTPM: Virtualizing the Trusted Platform Module***

Stefan Berger, Ramón Cáceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn, IBM T.J. Watson Research Center

Stefan Berger presented an architecture for creating virtualized trusted platform modules (TPMs). Multiple virtual machines cannot share a single hardware TPM, because of limited resources in the TPM and because the TPM has a single owner. In addition, virtual machines can migrate, whereas a TPM is physically tied to a single machine. The problem is that hardware-rooted security is still desirable.

As a solution to this problem, Stefan presented a virtualized TPM (vTPM) that is linked to the TPM through signed certificates. The vTPM is implemented as a software component, but one could also build it into a secure co-processor for improved security properties. Although there are still some challenges related to virtual machine migration, Stefan explained how the vTPM architecture allows a TPM to be extended to a virtual environment today.

In the Q&A session, the first question involved hardware requirements. Stefan said that TPM is a standard and is available from a variety of vendors. To a question about the size of the vTPM implementation, Stefan answered that it is about 250 kbytes per vTPM.

■ ***Designing Voting Machines for Verification***

Naveen Sastry, University of California, Berkeley; Tadayoshi Kohno, University of California, San Diego; David Wagner, University of California, Berkeley

Naveen Sastry presented techniques for building direct recording electronic (DRE) voting machines for easier verification. Current-generation DRE voting machines are built as one large, monolithic system. However, by building these systems using distinct components with minimal data passing, each component becomes easier to verify. Naveen proposed building separate components for each of the security-critical functions. In addition, Naveen recommended rebooting between voters to ensure that each voter starts with a known good system state.

In the presentation, Naveen discussed the need to select specific security properties and design the system to satisfy these properties. Specifically, he looked at two security properties: (1) none of the voter's interactions with a DRE may affect a later voter's sessions, and (2) a ballot may be stored only with the voter's consent to cast. Compartmentalized implementation techniques were used to demonstrate a system that upholds these properties.

In the Q&A session, one questioner asked about the problem of malicious developers. Naveen answered that this can be viewed as another security property and a system designed to handle the problem. Naveen also addressed concerns that a reboot would not clear the system memory by suggesting that one cut power to the system.

WORK-IN-PROGRESS REPORTS

Summarized by Manigandan Radhakrishnan

■ ***Exploiting MMS Vulnerabilities to Stealthily Exhaust a Mobile Phone's Battery***

Radmilo Racic, Denys Ma, and Hao Chen, University of California at Davis

This attack is aimed at draining the battery of the victim's cell phone without his or her knowledge. The cell phone discharges the most when it is in the *Ready* state as opposed to *Idle* and *Standby*. The attack works by keeping the cell phone at the Ready state as long as possible even when no useful function is performed. The authors achieve this feat by exploiting vulnerabilities in the Multimedia Messaging Service (MMS), specifically the Packet Data Protocol (PDP) context retention and paging channels. The attack works in two phases. The first phase involves the creation of a *hit list*, which is a list of mobile devices that includes their cellular number, IP address, and model number. The second phase is the battery-draining phase, where repeated UDP/TCP ACK packets are sent to the mobile phone either just before the timer expires or just after the timer expires (so that the network has to page the mobile device). The timer here is the duration the mobile device waits before going to Standby state from the Ready state. The experimental results show that the authors have been able to discharge cell phone batteries at a rate that is up to 22 times faster than that in the Standby state.

■ ***Applying Machine-Model-Based Countermeasure Design to Improve Protection Against Code Injection Attacks***

Yves Younan, Frank Piessens, and Wouter Joosen, Katholieke Universiteit Leuven, Belgium

Code injection attacks are more prevalent today, and the authors point out that the countermeasures are developed in an ad hoc manner. The authors suggest a more methodical approach to countermeasure development. They present two approaches, machine-model and meta-model. The machine-model is the model of the execution environment of the program including the memory locations (stack, global tables, etc.) along with the operations performed on them. The relations between the different components and their interactions are captured using UML (although the authors do note that they are working on a better alternative). Such a model not only allows the designer of countermeasures to function at an abstract level but also provides a means to compare and evaluate different countermeasures. The shortcoming of this approach is the fact that the machine model is closely tied to a particular architecture. To overcome this the authors suggest the meta-model, which is an abstraction of several machine models, and which, according to them, provides uniformity when constructing machine models and allows a designer to work out the global principles of a countermeasure independent of a specific platform.

■ ***Building a Trusted Network Connect Evaluation Testbed***

Jesus Molina, Fujitsu Laboratories of America

The Trusted Network Connect (TNC), a part of the Trusted Computing Group (TCG) protocols, comprises a set of standards that ensure multivendor interoperability across a wide variety of endpoints, network technologies, and policies. The particular issue that this presentation tries to address is the composition of secure applications developed by different vendors. The author is building a TNC testbed to test various vendors' products under a variety of policies. The idea is to expose any possible vulnerabilities.

■ ***The SAAM Project at UBC***

Konstantin Beznosov, Jason Crampton, and Wing Leung, University of British Columbia

Every authorization system has a policy-based decision maker and an enforcement engine. In the model presented in this WiP, they both communicate through message exchange to decide whether a particular authorization request is allowed. If this decision maker fails or is unresponsive, the system either becomes unavailable as no operations are allowed or gets breached because every operation is allowed. To prevent such a situation, the authors propose a Secondary and Approximate Authorization Model (SAAM). SAAM derives approximate authorization decisions based on cached primary authorization decisions. The efficiency of a SAAM system depends on the authorization policy of the system. The authors tested their scheme on a system implementing the Bell-LaPadula model and found a 30% increase in authorization requests that can be responded to without consulting the primary policy decision maker.

■ ***ID-SAVE: Incrementally Deployable Source Address Validity Enforcement***

Toby Ehrenkranz, University of Oregon

Routers deployed on the Internet today do not track the source of an IP packet. They base their routing decisions only on the destination address of each packet. The authors contend that this has been the root cause of IP spoofing attacks over the Internet and of the unreliability of the RPF protocol. ID-SAVE works by building "Incoming Tables" for valid IP addresses that can route packets through this router. A packet not matching a valid entry in the Incoming Table is dropped. The authors do reference Li et al.'s work ["Source Address Validity Enforcement (SAVE) Protocol," 2002] and mention that their approach has similar ideas to SAVE. The novelty of ID-SAVE lies in its approach toward deployment of the protocol in the Internet, which has many legacy routers that may not support Incoming Tables. They use a variety of techniques such as packet marking, neighbor discovery, on-demand updates, blacklists, and packet-driven pushback to derive benefits even with partial deployment.

■ ***Automatic Repair Validation***

Michael E. Locasto, Matthew Burnside, and Angelos D. Keromytis, Columbia University

Automated intrusion prevention and self-healing software leave the system administrator a little perplexed, since it is just not possible for him or her to manually ensure that the fix actually fixes the vulnerability. To address the issue and motivate further research in this direction, the authors propose bloodhound, a system that facilitates automatic verification of fixes by subjecting them to the original input that caused the intrusion. This can be achieved by automatically logging suspicious network traffic and reusing the data to test the fixed software. As was pointed out by the authors, the challenges to this approach lie in sifting through enormous amounts of network traffic to identify malicious inputs, in indexing identified input and storing them, and in ensuring that the replay is an effective guarantee to show that the problem is fixed. Also, this technique has time and space limits and has to be optimized for both if this technique is to be efficient.

■ ***Secure Software Updates: Not Really***

Kevin Fu, Anthony Bellissimo, and John Burgess, University of Massachusetts at Amherst

Software updates are used by a wide variety of applications to fix bugs and patch security vulnerabilities. This WiP took a comprehensive look at the update mechanisms found in some of the most common applications (Mozilla Web browsers, Adobe Acrobat, etc.). They found these mechanisms to be heavily dependent on the presence of secure networks for correct functioning. And there were instances of them being vulnerable to man-in-the-middle attacks. The software update mechanisms were also found to be prevalent in embedded devices (typically with limited computing power and battery life) that either operate over insecure networks or cannot support secure computations. The specific examples presented were those in use in automobiles, electronic voting machines, and some implanted medical devices. The statement "Help! My heart is infected and is launching a DDoS on my pancreas" to emphasize the need for secure content distribution was not only hilarious but also thought-provoking.

■ ***Integrated Phishing Defenses***

Jeff Shirley and David Evans, University of Virginia

This WiP presented an integrated approach to identifying and thwarting phishing attacks. The proposed system has two parts: (1) to identify suspicious emails and Web sites and (2) to prevent users from accessing these phishing sites. This integrated approach to identification combines existing identification heuristics (link obfuscation and email text contexts) with the authors' own idea of gathering URL popularity measures (derived using such common search engines as MSN and Google). The novelty of their system lies in their use of an HTTP proxy that diverts a user to a warning page when he or she tries to access a URL classified as a phishing site. The authors present experimental evidence that this greatly reduces the false-positive and false-negative rates. They also show that this scheme is successful in preventing a user from reaching a phishing site 94% of the time (with an error rate of about 3%).

■ ***The Utility vs. Strength Tradeoff: Anonymization for Log Sharing***

Kiran Lakkaraju, National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign

Logs contain a variety of information and some of these may be sensitive information about an organization. To facilitate sharing of logs among organizations, these logs need to be sufficiently anonymized to prevent any leakages. This WiP talked about the FLAIM framework for log anonymization. The authors described a "strength vs. utility" trade-off to decide on the extent of anonymization. Here "utility" refers to the amount of usable information that is still present in the anonymized log, and "strength" refers to the extent of anonymization. The noteworthy fact here is that the stronger the anonymization, the less useful the log is, and vice versa. The FLAIM anonymization framework uses multilevel policies coupled with a library of anonymization algorithms to anonymize logs. The anonymization policy is represented in a XML-based policy language.

■ ***Malware Prevalence in the KaZaA File-Sharing Network***

Jaeyeon Jung, CSAIL, Massachusetts Institute of Technology

More and more viruses are reported to use peer-to-peer (P2P) file-sharing networks such as KaZaA to propagate. These malware programs disguise themselves as files (e.g., Winzip.exe or ICQ.exe) which are frequently exchanged over P2P networks; they infect the user's host when downloaded and opened. They leave copies of themselves in the user's sharing folder for further propagation. The authors present a crawler-based malware detector, called "Krawler," built specific for the KaZaA network. The crawler has a signature database and looks for files with different file names but matching signatures known to the crawler. The crawler uses longest common substring to minimize false positives. Based on their experiments, the author reports that they found 15% of crawled files were infected by 52 different viruses, 12% of KaZaA hosts were infected, and 70% of infected hosts were in the DNS blacklists.

■ ***Election Audits***

Arel Cordero and David Wagner, University of California, Berkeley; David Dill, Stanford University

Random election audits are a way of ensuring that the election process is fair. When done correctly these audits can provide objective and measurable confidence about the election process. The difficulty lies in the selection of the samples because (1) the process for selection of the samples should be transparent and yet (2) the samples should be as random as possible. For example, the use of software for random selection may produce a random sample but is not a transparent process. To make the selection process fully transparent and to allow public oversight the authors suggest the use of a die (a 10-faced one, to increase the bandwidth). They are wary that the choice of dice may and probably will encounter resistance in adoption because of the public perception that dice are associated with gambling. And they also note that such public perception has led to superior selection techniques having been rejected in the past. They advocate education as the means to shift perception toward acceptance of physical means such as dice and a lottery in use of random selection.

■ ***The Joe-E Subset of Java***

Adrian Mettler and David Wagner, University of California, Berkeley

Every process executes with the same set of privileges inside the Java Virtual Machine (JVM). This means that the principle of least authority is not enforced to ensure that each process has just the necessary set of privileges. Joe-E is a subset of Java designed to build secure systems. Joe-E uses capabilities for the enforcement of protection; hence the program can perform only those operations for which it has capabilities. Joe-E has the advantage of permitting safely extensible applications, where an application's privileges can be limited to the capabilities granted, thereby simplifying analysis.

■ ***Prerendered User Interfaces for Higher-Assurance Electronic Voting***

Ka-Ping Yee, David Wagner, and Marti Hearst, University of California, Berkeley; Steven M. Bellovin, Columbia University

It is critical that the software present in voting machines be validated. The authors contend that the codebase in commercially existing voting machines is not only huge (37,000 lines) but also contains a lot of code that is related to the user interface (14,000 lines). Prerendering, to generate the user interface before the election, dramatically reduces the amount of security-critical code in the machine, thus reducing the amount and difficulty of software verification required to ensure the correctness of the election result. Moreover, publishing of the prerendered user interface as a separate artifact enables public participation in the review, verification, usability testing, and accessibility testing of the ballot.

■ ***Fine-Grained Secure Localization for 802.11 Networks***

Patrick Traynor, Patrick McDaniel, and Thomas La Porta, The Pennsylvania State University; Farooq Anjum and Byungsuk Kim, Telcordia Technologies

This WiP presented a novel approach to ascertaining a user's location in an unforgeable manner. This is essential when the user's location is a necessary part of user authentication. The authors use a network of access points to transmit cryptographic tokens at various power levels. This location information is further refined by using a low-cost radio in the desktop machines present in the vicinity of the location ascertained by the access points. The use of hash functions (as part of the tokens) ensures that this method is resilient against spoofing attacks.

■ ***KernelSecNet***

Manigandan Radhakrishnan and Jon A. Solworth, University of Illinois at Chicago

In most of today's operating systems, networking authorizations are practically nonexistent; that is, common networking operations are unprivileged and end-to-end user authentication is done (if at all) on a per-application basis and is never tied to the authorization system of the operating system. The KernelSecNet project is aimed at providing a unified networking and distributed system abstraction that is implemented at the operating-system level, allows networking policy specification, and provides automatic encryption of all traffic between hosts. The authors intend to develop a model that is specific to the KernelSec project and another independent UNIX-based model.

■ ***Taking Malware Detection to the Next Level (Down)***

Adrienne Felt, Nathanael Paul, David Evans, and Sudhanva Gurumurthi, University of Virginia

The WiP presented a novel technique for malware detection using the disk processors. The detection works by having the disk processor monitor sequences of I/O requests and identify sequences that correspond to known malicious actions (essentially, a signature). Implementing the detection at this low a layer makes the mechanism immune to most subversion mechanisms that are present in the layers above and makes it extremely hard to circumvent. Also, this mechanism works in isolation from the operating system and can operate even when the host is compromised. The authors mention that this technique is more effective when the processors can store more meta information about a request, for example, whether it is a create-file or a remove-file request.

■ ***Data Sandboxing for Confidentiality***

Tejas Khatiwala, Raj Swaminathan, and V. N. Venkatakrishnan, University of Illinois at Chicago

The authors present a technique they call "Data Sandboxing," which allows information flow confidentiality policies to be enforced on different parts of a legacy application to prevent leakage of confidential information. This is especially relevant when the application is a monolithic one that accesses (via reading or writing) ordinary as well as confidential media. The technique partitions the application into two parts (each run as a separate program) with each part having its own confidentiality policy. The first program performs operations on public output channels, and the confidentiality policy does not allow it to read confidential information. The second program is allowed to read confidential information, but is not allowed to write to public channels. The authors contend that this partitioning enables them to enforce a confidentiality policy that in totality prevents leakage of confidential information from the original program on publicly observable channels.