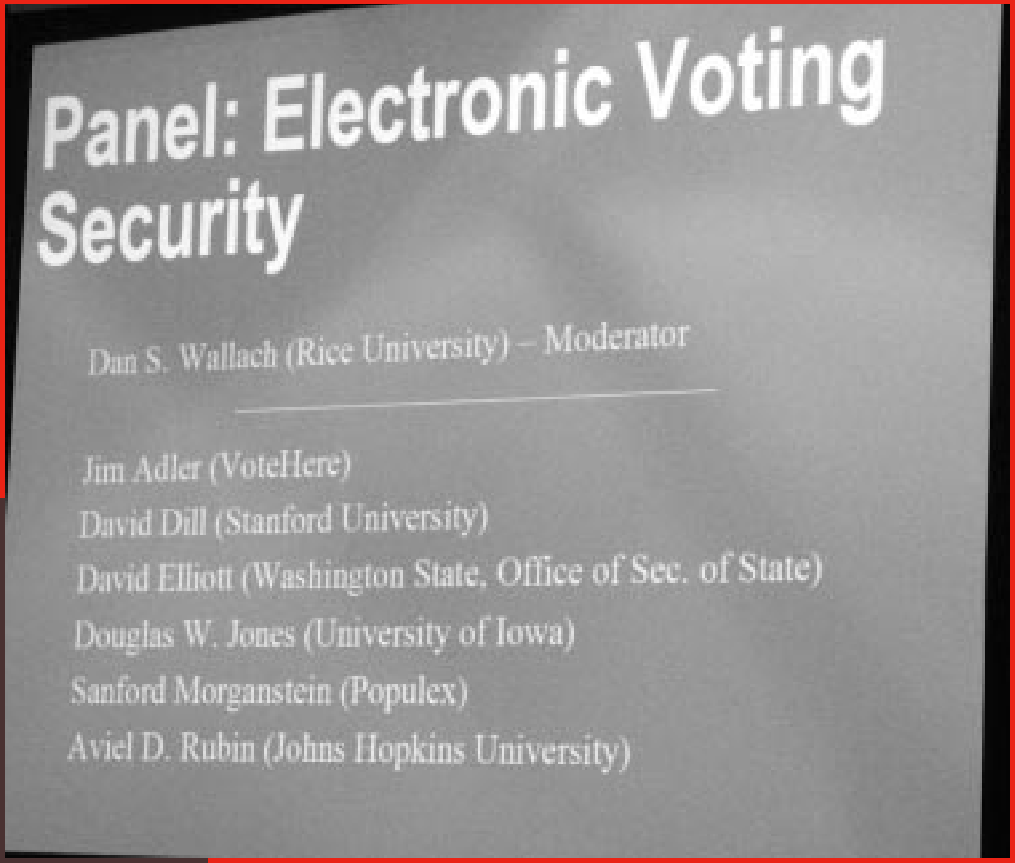
;login:

THE USENIX MAGAZINE

December 2003 • volume 28 • number 6



Panel: Electronic Voting Security

Dan S. Wallach (Rice University) – Moderator

Jim Adler (VoteHere)
David Dill (Stanford University)
David Elliott (Washington State, Office of Sec. of State)
Douglas W. Jones (University of Iowa)
Sanford Morganstein (Populex)
Aviel D. Rubin (Johns Hopkins University)

inside:

SECURITY

Perrine: The End of crypt() Passwords . . . Please?
Wysopal: Learning Security QA from the Vulnerability Researchers
Damron: Identifiable Fingerprints in Network Applications
Balas: Sebek: Covert Glass-Box Host Analysis
Jacobsson & Menczer: Untraceable Email Cluster Bombs
Mudge: Insider Threat
Singer: Life Without Firewalls
Deraison & Gula: Nessus
Forte: Coordinated Incident Response Procedures
Russell: How Are We Going to Patch All These Boxes?
Kenneally: Evidence Enhancing Technology

BOOK REVIEWS AND HISTORY

USENIX NEWS

CONFERENCE REPORTS
12th USENIX Security Symposium

Focus Issue: Security

Guest Editor: Rik Farrow

USENIX

The Advanced Computing Systems Association

conference reports

Our thanks to the Summarizers:

Scott A. Crosby
Catherine Dodge
Clif Flynt
Seung Won Jun
David Molnar
Chris Ries
Gelareh Taban
Tara Whalen

NOTE

Reports for BSDCon '03 arrived too late to be included in this issue. They are available at *http://www.usenix.org/publications/library/proceedings/bsdcon03/confrpts.*

12th USENIX Security Symposium

August 4–8, 2003 Washington, D.C.

KEYNOTE ADDRESS: REFLECTIONS ON A DECADE OF PSEUDONYMITY

Black Unicorn

Summarized by Tara Whalen

Black Unicorn, aka A.S.L. von Bernhardi, kicked off the conference with his talk on the relationships between identity, reputation, and trust. Anonymity has negative connotations, or "sunny climes attract shady characters." He chose his pseudonym when the cypherpunks list was being archived, and suggested that anybody who doesn't want to attract undue attention should probably pick a benign pseudonym, such as "Bill Smith."

Black Unicorn and Vern Paxson

He then launched into a detailed discussion of identity: its definition, its value, and some common fallacies. The definition he prefers is "the distinct personality of an individual regarded as a persisting entity; individuality."

The value of identity encompasses both its use as a unique identifier and in establishing the uniqueness of a reputational assertion.

The problems with identity are a lack of a standard unique identifier, reliance on third-party attestation, and the absence of static physical characteristics for identification. In addition, the link between identity and reputation is not addressed in conventional systems.

This led to the next phase of the talk: reputation. Black Unicorn's most appropriate definition is a specific characteristic or trait ascribed to a person or thing: a reputation for courtesy. The value of reputation is as a predictor of behavior, as a means of valuation, and as a means for third-party assessment. Black Unicorn took issue with the notion of reputation as a behavior-predictor.

Black Unicorn then discussed trust; he presented several definitions, choosing "reliance on something in the future; hope" as the most appropriate for this talk. The value of trust has several components: as a means of inexpensive due diligence; as a delegation enabler; as a means to reduce robustness of environmental security; and as an indication of risk tolerance.

Black Unicorn then discussed the relationships between trust, identity, and reputation, showing how factors such as credentials, third-party attestation, and certification affect trust. Identity information can be augmented by due diligence, consequence augmentation (e.g., penalties for crimes), and third-party attestation.

Q: Given the reticence of Americans to use ID cards, how will we be able to identify people?

A: This problem is likely to remain. There is no good answer; identity is always going to be nebulous. We have different roles that we play, and we tend not to like third-party assertions about our identities.

Q: How is the use of third-party attestations different from due diligence?

A: We often look at collections of attestations, which is better than looking at only one attestation. The problem with third-party attestations is that these parties have limited motivation to do proper due diligence (to save money), and their own liability is limited by insurance, which makes them less likely to be appropriately diligent. Direct experience and/or lore may be better guides.

Q: Do you want different ID documents for different purposes?

A: Definitely. You have to look at the agency making the attestation—what are they qualified to say? Probably not a lot. When looking at ID documents, you have to look at what the agency is good at attesting.

Q: Considering this issue from a cultural perspective: Identity is partially defined by race, religion, etc., and this affects one's reputation. Comment?

A: I agree. A lot of reputation is tied up in name, for example, and differs from culture to culture (e.g., Europe vs. US). Look at what's used for credibility: in Europe, it's family history, unlike in the US.

REFEREED PAPERS: ATTACKS

Summarized by Clif Flynt

Remote Timing Attacks Are Practical

David Brumley and Dan Boneh, Stanford University

The authors received the Best Paper award for their report on the viability of timing attacks on the RSA algorithm as implemented in OpenSSL. They proved that they could extract RSA private keys, not only when running on the same hardware as the secure application, but also across a network including several routers.

Dan Boneh described how a timing attack works and how to defend against it. In a timing attack, the secure server is sent many different queries that are expected to fail, and the time it takes to reject each query is measured. Rejecting a query requires that the server perform several math operations. Depending on the organization of the 1s and 0s in the query, different speed optimizations may be performed. A program can deduce the positions of 1s and 0s from the time required to perform the math operations.

The timing attacks have been done many times in the past, but previous work was done against "simple" devices like smartcards. Boneh and Brumley's work shows that a surprising level of noise can be introduced into the timing data without degrading it beyond usability. Complex systems with multiple applications running simultaneously, or even the variance of network latency, do not produce enough noise to disguise the time difference required to perform the analysis.

OpenSSL was chosen for this work because the package is used in many other applications, including mod_SSL, stunnel, sNFS, and more. Of the applications examined, only the Mozilla NSS packages defaulted to secure behavior.

Because the RSA algorithm requires many multiplications of very large numbers, most implementations use arithmetic optimization techniques to speed up encrypting and decrypting. These techniques are sensitive to certain bit patterns in the test and real key, making a message rejection faster or slower depending on how closely the bit patterns of the test key match the real key. A defense against this attack is to use RSA blinding, in which the client and server should decrypt a random string as well as the actual encrypted text to provide a random decryption time. This produces a 2–10% hit in performance. Brumley found that this was sufficient to prevent a successful timing analysis. This is implemented by default in versions of OpenSSL after version 0.9.7b.

802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions

John Bellardo and Stefan Savage, University of California, San Diego

While many members of the audience were connected to the outside world via their laptops and 802.11b wireless links, Stefan Savage demonstrated just how insubstantial that link can be. He opened his talk by showing a graph of current network activity and then attacked John Bellardo's link, halting his download. When this attack was started, the graph showed that traffic to Bellardo's laptop was reduced to nearly 0 packets, and Bellardo concurred that he was no longer downloading any data. Savage then extended the attack to the rest of the audience, and I can confirm that my connection to the outside world was gone. This lasted a few seconds after which Savage discontinued the attack, and service was restored.

He then described the two denial-of-service attacks he used, both of which use legitimate features of the 802.11b protocol. The first attack, which disabled only a single 802.11b user, used a deauthentication attack. In a normal conversation between an 802.11b client and access point, the client requests and receives an authentication. At any point in the conversation, the client can request that the session be deauthenticated. The deauthentication request is not a secure message; thus one client can send a message to deauthenticate another client. Once a session has been deauthenticated, the original client must return to the beginning of the conversation with the AP before transmitting any more information.

He described a simple technique to protect against this attack. When the AP receives a deauthenticate request, it should hold the request for a period of time before implementing it. If a fresh data packet from the deauthenticating client is received, the AP will not process the deauthenticate request.

Savage described other potential attacks based on fields designed to enable power conservation and reduce packet collisions. However, when they implemented some of these attacks they discovered that current implementations of most 802.11b interface cards don't implement these features, and data in those fields is ignored.

As the use of 802.11b becomes more widespread and quality-of-service demands become greater, we can expect new cards to implement these features, so Savage simulated a network of cards that implement the specifications correctly, and simulated attacks on that network. One of these specifications is the Network Allocation Vector (NAV) field. The NAV field allows an AP-client pair to reserve a period of time for their exclusive use. This allows a large packet to use up the bandwidth without having another unit collide with it. This time period can be as long as 31.5 milliseconds. The NAV bit is maintained by the firmware and, in theory, can't be modified by a user program. Savage described how an attacking application can scan the SRAM to find the value being placed into the packets and modify it. The defense against this attack is simply to reject unreasonably large values in the NAV field. In actual use, legitimate values are always under 3 ms.

A book-signing party

Denial of Service via Algorithmic Complexity Attacks

Scott A. Crosby and Dan S. Wallach, Rice University

Scott Crosby pointed out that algorithms have best-case, normal-case, and worst-case behaviors. When we implement algorithms, we rely on normal-case behavior. A system with malicious users may be forced to demonstrate worst-case behavior.

A hash algorithm displays a constant time to access an element (O(1)), unless there are many collisions. As the number of collisions increases, the behavior of the hash access approaches linear (O(*n*)). By discovering a few sets of letters that hash to "zero," an attacker can easily generate a large number of collisions.

Crosby demonstrated that thousands of collisions are required before the linear nature of the hashtable search becomes
a problem, but that this can be achieved in six minutes with a typical dialup modem. In an attack on a Perl application, he generated 90,000 collisions. After these collisions, retrieving a value for a key with no collision took about two seconds, while retrieving a value for a key that had collisions took about two hours.

This vulnerability was found in Perl, Squid, the Bro Intrusion Detection System, the Linux routing cache and directory entry cache, and others. Since the paper was written, the Bro IDS and the Linux vulnerabilities have been addressed.

Crosby described several alternative hash implementations and discussed their strengths and weaknesses. The universal hashing algorithm described by Carter & Wegman in 1979 is about as fast as the Perl 5 hash algorithm. The UMAC hash generator is optimized for modern compilers. A new hashing library dubbed UHASH was developed based on the Carter & Wegman code and UMAC code with further optimizations and generalizations to make it useful for applications with unknown length strings (CGI applications that may hash on user input), as well as applications where a key length can be determined (compilers where the keyword size is fixed).

The UHASH library is slower than the Perl hash algorithm for strings of fewer than 60 characters, but faster for longer strings.

INVITED TALK: DISTRIBUTING SECURITY: DEFENDING WEB SITES WITH 13,000 SERVERS

Andy Ellis, Akamai

Summarized by Seung Won Jun

Andy Ellis showed the audience some numbers to give an idea of the scale of the Akamai network: 14,000 servers that run 100 applications and are distributed in 2,400 different locations. They collectively serve 25 billion connections and 200 terabytes per day. Securing such a system is a challenge.

A traditional Web service model may provide confidentiality and integrity via the separation of application/database servers guarded by firewalls and IDSes, but it does not provide availability very well. Availability can be compromised when flash crowds or denial-of-service attacks occur. Akamai addresses it by delivering contents from the edge. While some servers on the edge may be overwhelmed, plenty of others can still serve the requests. The principle of delivering from the edge applies to almost all protocols Akamai supports, including DNS traffic as well as Web traffic.

Several management techniques were presented. Given so many servers and so few administrators (about 30), installing software could be a headache without an automated process, which is called "net deploy." Automated installation gives flexibility for managing servers. If a suspicious event occurs, the relevant servers can be wiped out and freshly reinstalled rather than examined and patched. Key management is done cooperatively by several components: key generation center, key distribution center, access control database, and audit server. An AuthGate is a gateway for access management. Administrators ssh to the AuthGate, which is replicated, to access edge servers rather than having direct access to them. This way, access control policy can be more flexible. For example, the policy can say that a particular user is allowed to access a certain number of machines rather than having the list of machines that are allowed to a user. Edge Diagnostics is the facility to test edge servers. The primary purpose is, hopefully, to show customers that the problem is not in Akamai's servers but somewhere else (maybe in the network). With so many servers, it is important to know what happens to which servers. Event management also requires a scalable solution. Events from several edge servers are merged into a log file, which is collected by Query Aggregator. The Monocle application automatically checks the Query Aggregator and reports any alerts to clients.

While Akamai runs many servers, it does not own, or have control over, any network backbone. Ellis mentioned that BGP is not particularly good about performance or reliability; it is all about "screwing the neighbors" or "not having your bits on my network." Though Akamai is across the street from MIT, Internet traffic requires 15 hops and is routed through New York. On September 11, this became 54 hops, with traffic routed through Israel. To mitigate BGP's routing, Akamai uses some servers to relay the traffic, which is especially effective for transcontinental traffic, producing up to 40% improvement in round-trip time.

Ellis concluded the talk by giving three lessons he had learned: plan for failure, use heavy automation, and make a decision in advance.

REFEREED PAPERS: COPING WITH THE REAL WORLD

Summarized by Clif Flynt

Plug-and-Play PKI: A PKI Your Mother Can Use

Peter Gutmann, Auckland University

In contrast to the many very technical papers regarding timing attacks, encryption, file systems, and packet protocols, Peter Gutmann examined the user experience of public key infrastructure (PKI) to obtain a certificate, and proposed some solutions.

He noted that using the current public key infrastructure is more difficult than it should be. While obtaining a connection to an ISP can be done in a few minutes with three pieces of information (login name, password, and credit card number), obtaining a certificate from a public certificate authority (CA) can take a skilled user between 30 minutes and a month, and may or may not have any verification.

In practice, most sites use the sample (clown suit) certificate that can be generated for testing purposes with OpenSSL, cryptlib, and others. What is needed is a system with the following prime directives:

- Don't scare the user. The certificate request forms should only require information they will actually use (username, password), not unnecessary information such as passport number.
- It must be possible to bootstrap the procedure with no previous certificate. Username/password should be all that's required to get a basic certificate.
- The user can submit the certificate to the CA to get it signed. The CA will return a signed certificate to the user.

The system Gutmann developed relies on a few assumptions: (1) The user has some existing relationship with the certificate-issuing agency. He proposes that banks offer CA services, since a user can be assumed to have a relationship with a bank that has already been verified. (2) There is a centralized server that can act as a CA locator. This is much like a DHCP server. In practice the HTTP 3xx Redirect message was used to redirect a user from a central site to the CA. (3) The resulting procedure need only be as secure as the applications warrant. This is for online shopping, not exchanging nuclear missile launch codes.

The first problem in obtaining a certificate is finding a certificate authority. This can be implemented by having ISPs provide a redirect from a common named location—for example, *http://pkiboot.example.com* or *http://www.example.com/pkiboot*, and using the HTTP redirect to redirect a browser (or automated certificate acquisition program) to the proper page. Current PKI systems use a baby duck security model. The user imprints on the first available system, and believes it to be secure. When started, an automated certificate-acquisition system will be initialized to remove existing state. It will trust the first certificate authority it discovers.

This is obviously insecure, since it is based on physical location. While there are many PKI RFCs in existence, none of them met Gutmann's requirements, so he came up with a partially home-grown solution. A user simply enters his or her username and password and obtains a certificate. A developer can create a Plug-and-Play PKI session that performs PKIBoot using username + password, generates signing key, requests signing certificate using username_password, and generates encryption key. This can be used with files or smartcards for key storage. Gutmann commented that SCEP (Simple Cert Enrollment Protocol), which is used in IPSec routers, is somewhat quirky. The certificate messaging is done with secured messages, but certificate fetch is done via (insecure) HTTP GET. The initial bootstrapping procedure is difficult, and there is no provision of rMAC'd messages. The SCEP system also uses all-in-one certificates, with no separation of signing and encryption.

Analyzing Integrity Protection in the SELinux Example Policy

Trent Jaeger, Reiner Sailer, and Xiaolan Zhang, IBM Research

The SELinux project is an attempt to provide mandatory access controls to Linux. The project includes a set of example rules that are intended to provide a secure base for developing local rule sets. Jaeger described a technique for analyzing those rules to determine whether or not the example policy is actually secure. This technique relies upon the Gokyo tool, which compares a set of SELinux policy rules and the desired integrity goals and reports how well the rules implement these goals. This work is done as part of the Linux Analysis Tools project, housed at *http://www.research.ibm.com/vali.*

The Linux Security Modules project provides a framework for implementing a set of Mandatory Access Control (MAC) rules within the Linux kernel. The SELinux project is developing a set of rules to be implemented by the LSM framework to implement a comprehensive integrity policy. For each application there are policy statements which define a particular threat model and the reaction to these threats. This can lead to many statements for each application on a Linux system. The SELinux policy base is composed of over 50,000 policy statements, making manual coverage analysis difficult.

At a system level, the SELinux defines an aggregate of the application policies. There is no coherent threat model, and the application policy interactions are not examined. This leads analysts to express concerns regarding the complexity and size of the system being analyzed.

Jaeger and his associates claim that the complexity is necessary because it flows from flexibility in the system. They also point out that while the policy base is large, they can reduce size. Only a smaller policy subset is needed to express an adequate Trusted Computing Base (TCB) rule set. Their technique is to assess the threats and evaluate the policy against TCB security requirements in the threat model. The goal of a Trusted Computing Base is to protect higher integrity data and applications from modification or misbehavior by lower integrity applications.

Security Holes . . . Who Cares?

Eric Rescorla, RTFM, Inc.

Eric Rescorla reported on research into user compliance with security upgrade notifications. They probed many machines to see what versions of OpenSSL were installed. They discovered that most sites that will install an upgrade (about 40%) do so immediately when an update is released. The next surge in updates (about 20% of sites) happens when an exploit is released. The number of patched sites asymptotically approaches 35%. They found that large installations are more responsive than small sites. This can be explained by large installations having a full-time administration staff responsible for installing upgrades.

During the question period, one person involved in writing up descriptions of security flaws wasn't aware that disabling SSLv2 would provide a workaround. A reward/punishment system for installing or not installing security upgrades might work, but nobody knows where the money for rewards would come from. It was observed that the OpenSSL upgrade was not trivial; if upgrades are not easy, they won't be installed. Rescorla was asked what percentage of sites surveyed were home-user systems. He replied that this was not known, and the number of poorly administered home systems is a serious problem.

INVITED TALK: PROTECTING THE INTERNET INFRASTRUCTURE

John Ioannidis, AT&T Labs–Research

Summarized by Seung Won Jun

The Internet has infrastructure, something upon which applications depend: links, routers, supporting services such as DNS servers and perhaps Google, and buildings where the equipment is located. Before getting into details, John Ioannidis mentioned several security mantras. There is no global security solution, so we must remain vigilant. Since security is always a cost-benefit trade-off, it is important to understand the threat model (who is out to get us and how they might get us), trust model (who are friends and from whom can we get help), and available tools (or, in the end, money, which is not sufficient but is necessary for security).

Infrastructure consists of several layers. Physical infrastructure includes fiber, wires, routers, buildings, and electric power. To protect the physical infrastructure against intruders, natural and manmade disasters, and other accidents, we can use hardening and replicating in conjunction with traditional measures such as alarms, locks, traps, and armed guards.

Bit transport is the infrastructure in which bits travel. Links are characterized by capacity, delay, and bit-error rate. Attacks are typically on capacity, that is, denial of service. According to a four-year-old survey of attacks, the DoS attacks are fairly crude and anisotropic, which means that attacks are directional rather than pervasive, probably because attackers can take control of only a limited number of hosts. What can we do about the DoS attack? We can detect the attack by monitoring traffic carefully or even marking some traffic. Upon detection, although we may not completely stop the attack traffic, we can reduce its collateral damage. A "blackhole" router, which is configured manually or semi-automatically, swallows the attack traffic rather than forwarding it to the victim. As a result, other parts of the network are saved from the attack. Going even further, we can "push back" the attack traffic by making a router tell its upstream routers not to forward the attack packets.

A major component of control infrastructure is DNS. There are many points where DNS can go wrong. While the root and TLD DNS servers are critical, there are so few of them that they can be, and are, DoS-attacked. The DNS response, which is carried on UDP, is not authenticated and, hence, can be spoofed. Cache poisoning (corruption of local DNS servers) can misdirect the traffic, and the servers are prone to be misconfigured. Although DNSSec tries to address some security aspects in DNS, it still leaves many problems unsolved. A semantic problem is fundamental: Does microsoft.com represent "the" Microsoft that you intend? Availability is still not addressed, and the key management for DNSSec will be a nightmare. The configuration of DNSSec is more difficult than DNS, and we have little operational experience.

Another component of control infrastructure is routing, particularly focused on BGP. As BGP is arguably the most distributed routing protocol, its complication is aggravated by the policy conflicts among ISPs. Route announcements are not authenticated, which can lead to a problem. Considering the effectiveness of such a simple measure as filtering out all BGP packets whose TTL is lower than 254, it is a pity that not all BGP routers follow such a practice. S-BGP, So-BGP, and IRV address the security issue of BGP, but security is always harder to bolt on later than to build in from the beginning.

PANEL: ELECTRONIC VOTING SECURITY

Summarized by Scott A. Crosby

The panel for electronic voting security included Dan Wallach as moderator, Jim Adler from VoteHere, David Dill from Stanford, David Elliot from Washington State's Office of Secretary of State, Douglas W. Jones from University of Iowa, Sanford Morganstein from Populex, and Aviel Rubin from Johns Hopkins University.

The panel started off with a warning by Dan stating that "no blood will be spilled," and asked if anyone from Diebold was in the audience. This was timely because two weeks before the conference, a highly critical report on Diebold voting machines was released. The authors of that report included Avi Rubin, the first speaker, and Dan Wallach, the moderator.

Aviel Rubin started with highlights from the report and a rebuttal from Diebold. He discussed high-profile disclosures and the risks of these disclosures: legal action against the researchers, restraining orders against publishing, PR blitzes to discredit the researchers, and jeopardizing one's job or career. He also made some firsthand observations of the response he experienced: people criticizing the paper without reading it, people calling his boss to complain, and detractors lying to the press and playing on emotion.

The next speaker was Doug Jones, a computer scientist who also sits on the Iowa Board of Examiners for voting machines. As part of that duty, he has assessed voting machines presented by various vendors, including Global Election Systems (later bought by Diebold). He said that in 1997 he talked to the head developer at GES about its flawed use of a static constant as an encryption key, a flaw that still existed over five years later in the code examined by Avi Rubin. He spent most of his presentation documenting that claim and how it shows that the voting machine certification system is demonstrably flawed.

After Doug Jones came Jim Adler. VoteHere does not make voting machines, but it creates software technology for

L. to R.: Aviel Rubin, Dan Wallach, David Elliott, Jim Adler, Douglas W. Jones, Sanford Morganstein

machines. Doug classified attackers as insiders and outsiders and pointed out the need for threat analysis and open technology. He described his company's voting system which uses cryptographic magic to construct ballots and secure shuffles to preserve anonymity. He emphasized the importance of looking at requirements, design, and risk analysis.

David Elliot described the problems of the current system as the result of benign neglect. It's designed with security based on controlled access to the system. He noted that there were no security standards until 1990, and even those are voluntary. The current testing methodology is hardware stress tests like heat and vibration, not security.

Next came Sanford Morganstein, a representative from Populex, a company that actually manufactures voting machines and is brand new to the field. Their system is a reimplementation based on the Mercuri Method, in which a computer voting machine prints out a human-readable receipt with a bar code.

The final panelist was David Dill. He focused on the cultural gap between computer scientists and voting officials. Voting officials believe that black-box testing can detect malicious code and refuse to listen to computer scientists who say differently. Short-term problems required fixing the regulatory and certification framework and stopping the acquisition of paperless voting machines. He described the long-term problem as satisfying the requirements that lead to touch-screen voting, such as a dislike of paper.

The panel then opened to questions. One questioner inquired whether open source is necessary for secure design. The response from Jim Adler was that all that is needed is verifiability of results. A point was made that even if open source was used, no one could know that a particular program was what was actually being run in the machine. There was disagreement between panelists over the need for a paper trail. Jim Adler was critical of a proposed bill that required one, claiming that it would be stupid and proscriptive; his design uses cryptography that would make paper obsolete. Aviel Rubin disagreed; a mechanism only Ph.D.s understand will be less trusted by the general public than paper.

INVITED TALKS: INTERNET SECURITY: AN OPTIMIST GROPES FOR HOPE

Bill Cheswick, Lumeta

Summarized by Tara Whalen

Bill Cheswick provided a historical perspective on computer security, explaining why he remains optimistic that good security is possible (despite his statement that "an optimistic security person may be an anti-job requirement"). Back in 1993, when he and Steve Bellovin were writing the first edition of *Firewalls and Internet Security*, the Web had not yet arrived on the scene and most network attacks were theoretical. There was no wide-scale sniffing, no massive denial of service attacks, not many worms. But fast-forward only a few years, and these attacks have become prevalent, coupled with a rapid rise in the use of the Internet. Cheswick stated that "there are lot more players, and on average they are a lot less secure."

However, there are also a lot of tools available that weren't around in 1994, such as widely available crypto, SSH, firewalls, and intrusion detection systems. Many of these can be easily deployed, as you don't have to "roll your own" security tools anymore.

There are a number of reasons why Cheswick remains optimistic: Reliable systems can be built from unreliable parts; we have control over the rules we set on our hosts; good encryption is readily available; and "the Bad Guys are giving us lots of practice."

Cheswick pointed out that a cost vs. benefit analysis must be performed: We need to figure out the value of our assets, and how much an attacker is willing to spend. Security is not perfect but only needs to be "good enough." There are some problems that resist easy solutions, such as buggy software, poor password choices, and social engineering. Also, even experts can't always get things right. To illustrate this point, Cheswick displayed some passwords that had been sniffed during the conference from the USENIX wireless network.

To mitigate these problems, he proposed several security strategies. First, stay out of the game if possible ("best block is not be there"), through such means as avoiding the monoculture of homogeneous systems. Second, deploy defense in depth by engineering redundancies into your systems. Third, make security as simple as possible: Set up secure defaults and use hardware tokens. Finally, design security into a system from the start, because it can't simply be added later.

Cheswick continued his talk with a discussion on firewalls. Although they are useful in many situations, they have certain drawbacks. For example, people go around them, and they offer no protection from insiders. Another problem is that firewalls are often used as perimeter defense around very large perimeters; Cheswick believes that smaller enclaves are much safer. Note that you don't have to use a firewall. Cheswick stated that this is like "skinny-dipping on the Internet: somewhat exciting, but with an element of danger." Such an approach requires secure host technology, like the current efforts in *BSD and Linux. One technique is to jail servers (and clients), for example, through chroot. Cheswick provided a list of "routes to root" that should be minimized (such as root network services and setuid programs), and stated that chroot is the only standardized layer of defense that we currently have. He described his experiences with jailing programs, outlined some practical difficulties of this approach, and listed some of the programs he has jailed (Web servers, Samba) and those that probably should be jailed (Apache, NTP).

Cheswick next provided an interesting diversion about "spook networks," telling the audience to talk to spooks for their advice (rather than their secrets). He said that they seem to have a great deal of success running secure networks, and have adopted good practices for maintaining secure systems (e.g., using enclaves and restricting client software). Cheswick finished his talk with a security wish list, which included more work on chroot, formal analysis of crypto, and sandboxes for browsers. He concluded with his contention that things can get better, with enough work and diligence. The audience responded to this talk with anecdotes and opinions, as well as a few questions:

Q: What's the worst thing you could do to the Internet?

A: I don't want to give specifics, but the worst thing I can think of would take it down for weeks. As in the past, I expect that experts would step in to respond immediately, but they'd probably all have to phone each other.

Q: You didn't mention how to help users operate security tools correctly.

A: I don't think that users are going to become more competent. My hand-waving answer is to use tools like USB dongles, obtained from a trusted source.

REFEREED PAPERS: HARDENING I

Summarized by Chris Ries

PointGuard: Protecting Pointers from Buffer Overflow Vulnerabilities

Crispin Cowan, Steve Beattie, John Johansen, and Perry Wagle, WireX Communications

The goal of most attackers in exploiting vulnerabilities is to execute code that they provide, commonly known as shellcode. Usually this is done by overwriting a pointer and aiming it at their own code that has been placed somewhere within the attack space. Different approaches can be taken to protect these pointers, such as surrounding them by canary values that expose an attack if they are overwritten. The authors, however, take the approach that pointers are dangerous until they are loaded into a register, and so their method involves encrypting the pointer until this occurs. During an attack, when the pointer is de-referenced after being overwritten, it will not jump to where it was intended, since the value it was overwritten with will be decrypted. Instead, it points off into some "crazy space" and the program crashes. PointGuard works at compile time and essentially XORs pointers stored in memory with a secret key. This key is kept on its own page, and the page is marked read-only so that the attacker cannot overwrite it. It can be implemented at different times during the compilation process – either at the pre-processor stage, intermediate representation, or the architecture-dependent stage – but the authors decided to implement at the intermediate level. This stage is late enough to avoid most chance of the defenses being optimized away, and early enough to still have type information to distinguish between pointer and non-pointer data. There are some difficult issues that arise while using PointGuard, such as mixing PointGuard-compiled code with code that was compiled without it. For this, the authors added compiler directives.

Another major problem, preventing cleartext leaks when register values are stored on the stack to free them up for other purposes, will be prevented in future implementations.

The authors were surprised to discover that in some situations code compiled with PointGuard actually performed better than code that was not. This is probably because PointGuard uses registers more heavily than the normal compiler. Other performance costs varied from less than 1% to 21% overhead.

Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits

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

Using a language such as C or C++ allows the programmer to have greater control over memory with tools such as pointers, but this opens programs up to memory errors such as buffer overflow vulnerabilities. One possible solution is to use a type-safe language, but C is too widely used today to be completely abandoned. Other solutions involve educating the programmer or ensuring that the programmer's assumptions about input are always valid. Instead, the authors provide a solution that involves directly stopping the attacker. In order to exploit these memory errors to gain control of the program, the attackers need to know such data as the distance between the buffer their input is placed into and the return address or the memory address of the buffer itself. By randomizing the virtual addresses of the memory the program uses, the attacker will jump to a random location and crash the program most of the time, which provides a telltale sign of the attack. Similar solutions are being used in both the PaX project and PointGuard.

Address obfuscation is performed during linking, loading, and execution. The authors' method essentially involves three different ways to obfuscate: (1) Randomize the base address of the stack, heap, and code memory regions, as well as the starting address of dynamically linked libraries; (2) add random gaps within the stack between allocated memory on the heap, and to routines, to be just jumped over; and (3) permute the order of local variables on the stack, static variables, and the routines of dynamically linked libraries and of the program itself. Such obfuscations make exploits that rely either on absolute addresses (stack smashing, return-into-libc, heap overflows, double free, data modification) or on relative addresses (partial overwrites and data modification) much more difficult. Many attacks are very unlikely to be successful when this method of obfuscation is used (the authors report a $4*10^{-5}$ success probability with return-into-libc attacks). It also involves no change to source code, and very little overhead.

High Coverage Detection of Input-Related Security Faults

Eric Larson and Todd Austin, University of Michigan

Many software vulnerabilities, such as the latest MS RPC DCOM vulnerability, arise from the failure to perform proper boundary checking on data. Data received from a network, for example, is often trusted and put into a buffer without first checking that the buffer is big enough. Other bugs occur when string library functions are improperly used, and these bugs can lead to serious security vulnerabilities, such as stack overflows and format string bugs. Searching for these bugs can occur at several different stages. It can be done at compile time, in which case there is no dependence on what input is actually used, but this makes things like keeping track of data on the heap difficult. Instead, the authors decided to take the approach of checking at runtime, but eliminated many common weaknesses of doing it at this stage, such as specifying actual input to check for bugs. Since the method will find a bug even if a dangerous value is not input, it only needs to be used during the software testing phase. When testing is complete, it is no longer used, and therefore there is no performance penalty. The authors' method involves shadowing any variables that contain user input. For example, integers are shadowed by a variable that stores its lower and upper bound, strings are shadowed by a variable that stores its size and whether it is null-terminated, and array references are shadowed by a variable containing possible ranges. These shadow variables are adjusted at any control points, such as loops, if statements, and arithmetic operations. At any use of the input that is potentially dangerous, the possible range of the input, not the input itself, is checked, so even if a dangerous value is not input errors can be detected. Using their implementation, the authors tested eight different programs and discovered 16 bugs, including three in the popular OpenSSH program. Most of these were the result of unbounded integers used in loops. Their method does have some limitations, such as only checking the execution paths taken during runtime, and it also has high runtime performance penalties. Some of their future work will involve optimizing their current implementation.

INVITED TALK: WHEN POLICIES COLLIDE: WILL THE COPYRIGHT WARS ROLL BACK THE COMPUTER REVOLUTION?

Mike Godwin, Public Knowledge

Summarized by Tara Whalen

Mike Godwin described how legislation put forward by content providers (media interests) will impede progress in the computer sector. The fundamental issue is that computers are very good at making copies: this is part of their basic functionality. This is not news to the IT sector, but it was news to the media (particularly the music industry). Content companies did not expect duplication of digital media; the music industry is an object lesson for the rest of the content industry (TV, movies) – they are looking at file trading, fear it will destroy them, and want to stop it.

In the summer of 2000, Michael Eisner told Congress there was a need for more legislation to protect copyrighted works. Only two years earlier, Jack Valenti said, "We aren't going to need legislation, because DCMA will protect us." But Eisner felt that more legislation was necessary, stating, "The problem for us is computers." Not file trading, or broadcasting, but computers. As usual, media players wanted targeted legislation, which is never as narrow as they think it is.

Hollywood can't say, "We have to stop the computer revolution." This won't work at all. But they can propose systems that have the effect of slowing down the pace of the computer revolution. For example, in the mid-1990s, Hollywood proposed legislation to protect video content through a scheme that marked every small set of frames. You'd play the video and the computer would look for the marks – if marks not there, then okay to copy, else abide by copyright rules. This is a very inefficient solution from a computer perspective: You need to dedicate resources to check for copyright marks. After extensive IT and media debate, they eventually derived the DVD standard. This standard (a) doesn't need to look for a mark, and besides which (b) DVD turned out to be a huge windfall.

Godwin went on to say that this experience taught different lessons to IT and media. IT companies concluded that if they negotiate with Hollywood, they can get a good compromise. What Hollywood concluded is that if they muscle hard enough, they can drag IT to the table and get largely what they want. Fast-forward to the Hollings Senate bill. It would have required a copyrighted-work-detecting chip built into every digital device. For this approach to work, there would need to be a regulatory buttress supporting it, because you can beat this scheme fairly easily. This now moves the debate into the arena of policy decisions, which would require re-architecting the digital world from top to bottom.

Godwin is concerned that such legislation will hamper progress: IT has been built on open architectures, which created opportunities and investment. "If Hollywood had its way, computers would be more like consumer electronic devices, with limited, controlled functionality. . . .What's troubling to me is that the content guys don't seem to see what harm they may be doing. They know that marking schemes require making devices untamperable, but they don't seem to realize this has a high cost – to us, but also to them. They will slow down the computer revolution that they themselves use for their business."

Godwin added that this is about making computer platforms acceptable to the content industries. We value user control in the IT community, and we don't want to give that up, but this model is now at risk. He concluded his talk by asking: What do we tell Hollywood? The glib answer is, "Develop another business model." A better answer is, "If you are concerned about content, we can design a more secure system that is leaky but overall will still make you money [like the DVD model]."

The talk was followed by extensive audience commentary and questions:

Q: We seem to be losing the Hollywood battle. What position can we actually hold?

A: The best compromise is probably to tell Hollywood to encrypt content at source and restrict the decoding to software decoding, such as with the DVD model.

Q: But studios aren't happy with the DVD model – can we hold them to it so they won't keep demanding changes?

A: When we negotiate compromises, we need to include consumer interests, not just studio interests.

Q: This situation reminds me of the crypto wars, with lots of regulation set up against a small group of geeks. Progress went ahead only when business came forward as an ally – who's the new ally in this arena?

A: I'm happy to work with any company who wants to work with me on these proposals. There are IT people who are suspicious of Microsoft or Intel, and you shouldn't assume that they are your enemies: use their muscle.

REFEREED PAPERS: DETECTION

Summarized by Chris Ries

STORAGE-BASED INTRUSION DETECTION: WATCHING STORAGE ACTIVITY FOR SUSPICIOUS BEHAVIOR

Adam Pennington, John Strunk, John Griffin, Craig Soules, Garth Goodson, and Gregory Ganger, Carnegie Mellon University

Current intrusion detection systems are often implemented at either the host level or the network level. There are scenarios, however, where both of these can fail. If an attack is too new and there is no signature for it, or worse yet, if it is misconfigured, a network intrusion detection can fail to spot an intruder. Host intrusion detection systems can also be disabled by rootkits after a host has been compromised, and logs can be scrubbed to cover the attacker's footsteps. Adding storage-based intrusion detection could help "augment the capabilities" of already existing systems and spot some of these attacks. The authors' idea was to implement an intrusion detection system on a host used for storage by multiple computers. With this setup, the intrusion detection system sees all persistent activity, and is also on a separate self-contained host, so it is not susceptible to being disabled after one of the hosts that uses it is compromised. It monitors changes to static files or corruption of well-understood files such as /etc/passwd, unexpected changes to the middle of a log file, or any suspicious content. The authors state that the storage IDS is no "silver bullet" and has limitations and weaknesses like any other IDS. One unique limitation is that it only sees storage traffic, so other traffic such as denial-of-service attempts will not be detected. It is also susceptible to general IDS weaknesses, such as false positives and misconfiguration. It does, however, add another level of detection. The authors concluded that storage IDSes provide a new vantage point to watch from, with minimum space and performance costs.

DETECTING MALICIOUS JAVA CODE USING VIRTUAL MACHINE AUDITING

Sunil Soman, Chandra Krintz, and Giovanni Vigna, University of California, Santa Barbara

One of the original goals of Java was to provide dynamic content embedded into a Web page. Since then its use has spread much further, due to such strengths as security, flexibility, and portability. Currently, Java security consists of type system and verification mechanisms, as well as mechanisms that can be used for authentication and access control. Fine-grained auditing and intrusion detection could improve the existing security built into Java, but running a host-based IDS as a separate process will not be effective, since multiple Java applications are run within the same virtual machine. The goal of the authors was to add an event stream at the JVM level with fine-grained response, which would allow for intrusion detection within the JVM. To create their built-in event stream, the authors first began with JikesRVM and extended it to associate a user ID and IP address with each thread. They also built in an auditing system that consists of an event driver, an event queue, and an event logger thread that runs as a system thread and reports to an external audit log. This audit log is then processed by an external IDS based on the STAT framework developed by the authors in a prior work. They extended that STAT core with a language-extension module, an event provider that collects the events, and a response module for reacting to attacks. To test their IDS they created various attack scenarios and showed the reports generated from them. They demonstrated harmful thread intercommunication, unauthorized access detection, and privileged information leakage. The alerts generated from these attacks included time, action, source thread ID and UID, internal network and remote address, sensor that detected the attack, result of the attack, and message-specific data. They also tested the performance with both partial and full logging enabled: performance time increased from 1 to 2.5 seconds (on an Intel Xeon 2.4GHz with 1GB RAM). In the future they plan to reduce this overhead.

Static Analysis of Executables to Detect Malicious Patterns

Mihai Christodorescu and Somesh Jha, University of Wisconsin

Mihai Christodorescu began by discussing how malicious code detection, like many other areas of computer security, has really become an arms race between the good guys and the bad guys. Detection of viruses and other malicious code began with the use of signatures to identify the malicious code. To foil this detection, the bad guys started using techniques such as register renaming, and the good guys countered this by using regex signatures that were essentially templates that allowed the signatures to have gaps. Next, the bad guys started packing and encrypting the malicious code, and the good guys countered this by using emulation and heuristics. Now the bad guys have started to use code reordering and integration to try to keep their malicious code from being detected, and this type of obfuscation is not always detected. The authors obfuscated four viruses using NOP-insertion and code transposition, and then scanned them using three popular antivirus tools. The viruses that they used were Chernobyl, zombie-6.b, f0sf0r0, and Hare, and none of the scanners detected the obfuscated versions. Their tool, however, called SAFE (Static Analyzer for Executables), detected all four of them. A malicious binary is first loaded into SAFE, then a control flow graph (CFG) is created for each procedure. A program annotator then reads a CFG and a set of abstraction patterns from a library. The output is an annotated CFG that has patterns associated with each node of the graph. For example, a NOP instruction would have a pattern matched with it that states that it is an irrelevant instruction. A pattern consists of a list of typed variables, a sequence of instructions, and Boolean expressions. A detector then reads this annotated CFG and compares it to a malicious-code automaton to decide whether it contains any malicious code. The advantage of their approach is that SAFE is able to detect malicious code even if it has been obfuscated. As was pointed out during the Q & A, as long as there is a malicious-code auto-maton for that type of behavior, obfuscation will not fool SAFE. The performance of their implementation seemed to be successful, as no false positives or negatives were encountered. They plan to improve SAFE by having it look for different types of malicious code, such as trojans.

INVITED TALK: PHYSICAL SECURITY: THE GOOD, THE BAD, AND THE UGLY

Mark Seiden, MSB Associates

Summarized by David Molnar

Mark Seiden began by talking about the basics of physical security. In the physical world, unlike online, the concept of "secure perimeter" is real and meaningful. Security audits begin by talking about security analyses. Unfortunately, it's important to make sure that a security analysis is realistic. Too often, whenever someone says, "I did a security analysis and determined that it was not a threat," "it" ends up being something they didn't think about.

Seiden went over the basics of establishing a secure perimeter. Check doors, windows, and also roof and tunnel access. As an undergraduate at Columbia University in the late 1960s, he reported on student demonstrations and police reactions to them by sneaking into the main library at Columbia through the sewers. Because the police had chained and locked the front door, they thought it was "secure."

Another case dealt with a supposedly "ultra secure" co-location facility Seiden had visited as part of his work. The facility boasted motion detectors, alarms, and other measures for notifying security when a breach had occurred. Unfortunately, it also had pull-up floors. Seiden and a colleague attempted to trip the response system by first going under the floor to come up in a server room and then tripping the motion detector. Instead of calling security, the motion detector simply opened the locked door, a common setup to prevent someone from accidentally being locked inside.

Seiden also talked about the divergent security philosophies of the physical and network security communities. For the physical security community, security by obscurity is valid – it means an adversary has to put in more work to determine the layout of your installation, and this may buy you time after a break-in. By contrast, the network security community does not value security by obscurity nearly as much and therefore will share information at conferences such as this one. As a concrete example, the speaker pointed to Matt Blaze's recent work on rights amplification in master-keyed locks. The physical security community had known about this issue for a long time, but their culture was to keep it secret and not talk about it. The network security culture published it so as to effect systematic change.

Another side effect of the culture of secrecy is the suppression of information regarding incompatibilities that affect security. For example, the most popular deadbolt and the most popular electric strike (a device that allows electric operation of a lock, such as by a computerized access control system) are incompatible. Installing the strike disables the deadbolt-locking mechanism, degrading the lock to the security of just an ordinary house lock.

As a case study of the boundary between computer and physical security, the speaker talked about a magstripe access control system he had audited. The system consisted of panels and a central access control list system. Panels had limited memory and communicated with the main system via dialup modem. The system then pushed lists of approved users and pulled access events (such as attempted unauthorized access). Unfortunately, the dialup connection was not authenticated in any way; anyone could call up the panel or alternatively interpose and pretend to be the main access control system. At the least, such a compromise could be used to flush the panel's record of access control events, because panels did not keep records after the first attempt to send events to the main system. At the worst, password guessing or eavesdropping could be used to take complete control of an access panel.

The speaker and his team also took a look at the source code of the access control system. It turned out that the source code contained #ifdefs specific to each customer of the system. From the code snippets, much could be inferred about the structure of these customers' access control needs. The audit also uncovered several cache-consistency issues between panels and the main system. When questioned, the manufacturer claimed that everything was operating "exactly as designed."

REFEREED PAPERS: APPLIED CRYPTO

Summarized by Gelareh Taban

SSL Splitting: Securely Serving Data from Untrusted Caches

Chris Lesniewski-Laas and M. Frans Kaashoek, MIT

This presentation focused on the problem of reducing bandwidth load from a server Web site while still allowing high-content throughput to its clients. One possible solution is to use mirror sites. That is, the server can ask a group of volunteers to proxy the contents of the site and so distribute the bandwidth load. However, this scheme has potential for mischief, whereby an adversary can pose as a volunteer and serve modified contents to the clients. So this scheme requires trust in the proxy from both the client and the server.

A cryptographic solution is for the server to attach a signature to the data being served and allow the client to ensure the integrity and authenticity of this content by verifying the signature. Existing protocols that deal with this problem are not practical, since they either require the client to use a specialized browser (e.g., S-HTTP, SFSRO), thus leading to the depreciation of their deployment, or the protocol operates at the channel level and not the file level.

An example of the latter protocol is the widely deployed browser support for the SSL protocol. The author splits the server end of the SSL connection by introducing a proxy between the server and the client, such that the proxy is not privy to the shared client-server key. In this scheme, the proxy caches the data content of the server. In response to a request from the client, the server sends the proxy the unique identifier of the requested data as well as a message authentication code (MAC) for the data, using the shared secret key. The proxy then sends the cached data and associated MAC to the client. The server thus is able to offload bandwidth to mirrored sites while maintaining the integrity and authenticity of the data. The problem with this scheme, however, is that there is no end-to-end confidentiality; the server only distributes bandwidth load and not CPU.

In short, SSL splitting reduces bandwidth load while guaranteeing end-to-end data integrity and offering transparency to the client. For more information, visit *http://pdos.lcs.mit.edu/barnraising/*.

A New Two-Server Approach for Authentication with Short Secrets

John Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo, RSA Laboratories

Ari Juels opened his presentation by recalling the other name of the project, "Nightingale," explaining, "It was a midnight project." He went on, "The project began as an inquiry into the mind of the hacker." What does a hacker want? Fame, wealth, love . . ? Alas, the hacker wants "root access"!

The sensitive data the project "Nightingale" is intended for are short secrets and "life" questions, commonly used on the Internet today to authenticate users. How are these data stored securely on the application server? There are two main types of protections: frontline defenses such as up-to-date security configuration, authentication, and intrusion detection, and cryptographic rearguards such as hashing and encryption. However, due to the architecture of the scheme, there exists a "single point of compromise." This means that once access is obtained to the server, all data on the server is compromised. The adversary is able to launch various attacks, exploiting the weaknesses of the cryptographic techniques employed.

Nightingale eliminates the weaknesses of a single point of compromise by distributing the secret data between two servers, the application server and the Nightingale server, so that compromise of a single server does not compromise the data. The data can be shared using threshold cryptography and can be used to perform distributed (or blind) secret verification, remote reconstruction of data shares, or management of cryptographic keys.

The architecture of the system is important. Instead of allowing both servers to be equally accessible by the client, it is proposed that the Nightingale server be placed behind the application server, allowing better security protection as well as client transparency of the servers. Furthermore, different implementations of the servers allows different levels of security and protection for the secret data.

Presently, Nightingale is being integrated into the RSA Security Inc. product lines. For more information on the Nightingale system, please refer to *http://developer.rsasecurity.com/labs/nightingale*.

Domain-Based Administration of Identity-Based Cryptosystems for Secure Email and IPSec

D.K. Smetters and Glen Durfee, PARC

How do we make public keys more usable?

Deployment of cryptographic techniques today in such applications as secure email and IPSec has been hampered by the distribution of keys in a public key infrastructure (PKI). More specifically, the sender has to ensure that the receiver has generated a certified key pair, and the sender must also obtain an authenticated copy of the receiver's public key. This means that trust must exist between the sender and the certificate authority (CA) of the receiver. However, there are many difficulties involved in establishing a large-scale PKI, most importantly the question of large-scale trust between users.

The authors automate key distribution in a PKI setting by combining the idea of limited trust in the existing hierarchical DNS server infrastructure (using DNSSec to ensure security) with the usability advantages of identity-based cryptography (IBC). The main idea of IBC is to make the private key a function of the public key and some IBC parameters, thereby allowing the "identity" of a user to be her public key. In practice, however, the global scope of trust and namespace needed in a large-scale infrastructure is unacceptable for most applications. The proposed DNS-based IBC (DNSIBC) allows limited but usable scope.

The authors bootstrap trust from DNSSec such that clients are authenticated in their own local domain. This means that key-escrow, an automatic side effect of IBC, will also be limited to the local domain. The scheme requires no secure servers on the Internet, and only IBC parameters need be stored on DNS. Key revocation can be easily implemented by using time expiry for the keys. DNSIBC is easily deployed and incorporated in S/MIME and IPSec.

INVITED TALK: THE INTERNET AS THE ULTIMATE SURVEILLANCE NETWORK

Richard M. Smith

Summarized by David Molnar

From the abstract, I expected that the talk would be about the current uses of the Internet for surveillance. Instead, the speaker gave a more speculative talk about how the Internet could be used in the future to enable global tracking and surveillance of individuals. This speculative talk managed to provoke a great deal of discussion, particularly in the area of radio frequency identification (RFID) technology.

Smith began by pointing out that more and more devices will become IP-enabled, including phones, laptops, and PDA devices. He then defined devices analogous to a URL but intended to encapsulate information about a target's physical location. These location markers could be generated and then sent surreptitiously through any Internet-enabled device to a central database. The result would be a pervasive infrastructure for keeping track of the locations of people and passing the information back through other devices to a central database.

How can a device recognize what person has come in contact with it? The speaker pointed to several methods, but the method that excited the most comment was that of keeping track of a person by his or her personal RFID profile. The speaker outlined the current state of RFID technology and future trends.

Passive RFID tags consist of small chips connected to antennae. The chips are powered by a radio broadcast from a special RFID reader. These passive RFIDs carry an ID number and not much more, which allows them to be used to replace barcodes and to perform remote tracking. Currently, passive tags cost less than 50 cents (US) per tag, but the eventual goal is to reduce costs by another order of magnitude, to five cents, to enable per-item RFID tagging. Active RFID tags are more expensive, have their own internal battery, and are generally used for containers or other aggregates of goods.

One of the envisaged applications for per-item RFID tagging is theft reduction. The speaker talked about a recent field trial of RFIDs in Gillette Mach3 razors. Mach3s are easily resellable and, consequently, often shoplifted. The trial combined RFIDs on each razor package with a "smart shelf" that took a picture of a customer whenever he or she removed "too many" packages of razors. The idea here is that if the razors are later found to have been shoplifted, there is a picture of the perpetrator.

Smith then demonstrated a Texas Instruments RFID reader and invited audience members to come up afterwards to check whether their devices carried RFID. The reader connected to a laptop; associated display software picked up RFIDs in several items carried by the speaker and displayed them on the screen. The audience could then clearly see how some innocuous-looking items from the speaker's wallet in fact turned out to contain RFID tags.

The discussion of RFIDs turned out to be controversial even before the question and answer session. Perry Metzger interrupted the speaker at several points to underline potential threats involved in pervasive RFID. For example, would it be possible to build a device that allows someone to travel through a crowd and read off the RFID information of all passersby? Metzger claimed such a device was easy with current technology, while the speaker was not so sure. The speaker also disagreed with Metzger about the feasibility of an active device that could "burn" out RFIDs by sending them too much power.

During the question and answer session, much discussion focused on the nature of RFID technology and its implications. Some audience members questioned the speaker's assertion that per-item RFID tagging would be driven by shoplifting protection, since shoplifting is just "part of the cost of doing business." Others wanted to know if the RFID reader demonstrated would catch all RFIDs that might be on their person; the speaker replied that was not true, because multiple standards for RFID currently exist. A reader for one standard may not read tags for another. A great deal of discussion involved the range of RFID readers and the possibility of RFID "burners," but without much agreement.

Summarizer's note: Readers who desire more information on RFIDs may find the following links helpful:
Silicon Valley RFID Yahoo! Group: *http://groups.yahoo.com/group/sv_rfid/*;
MIT Auto-ID Center: *http://www.autoidcenter.org/*
RFID Privacy Symposium at MIT, November 15: *http://www.rfidprivacy.org/* and associated Web log *http://www.rfidprivacy.org/blog*.

PANEL: REVISITING TRUSTED COMPUTING

Panelists: Douglas Barnes, Dave Safford, William Arbaugh, and Peter Biddle
Summarized by Catherine Dodge

The panel began with short comments from each of the panelists: Peter Biddle from Microsoft, Dave Safford from IBM, William Arbaugh from the University of Maryland, and Douglas Barnes founder of the Government Open Technology Information Project.

In his comments, Biddle referred to a technology as NGSCB (pronounced ink-scab), its current moniker within Microsoft. In his presentation, he focused on how NGSCB is being developed to meet customer concerns about the erosion of their IT security perimeter. Increasingly, mobile computers, cell phones, PDAs, and other wireless devices are accessing the network, often bypassing outdated security measures not designed to handle these new technologies. He cited the occurrence of Xbox tournaments over the Microsoft corporate network as evidence that administrators have less and less control over the ways their IT systems are being used. This has led to customers wanting a means of doing application-to-application authentication that will protect intellectual property information, be it a clinical trial database or a secret recipe. Along with this protection, customers are asking for technologies that will allow them to share select information and systems with suppliers, partners, and customers in a secure and controlled way. Micro-soft's vision is that applications must have third-party verification before running, which would prevent an application that has had a keystroke logger added to it, or a bogus user account, from running. Biddle's final point was that Microsoft has heard loud and clear that customers do not want to be "locked in," having once deployed this technology. For further reading, consult a paper on authenticated operation of operating systems (*http://cs-people.bu.edu/mpe/acisp.pdf*) and white papers about the NGSCB architecture (*http://www.microsoft.com/resources/ngscb/productinfo.mspx*).

Dave Safford began his comments by holding up the August 2003 issue of *Linux Journal*, which includes an article on the open source tools for TCPA that IBM has developed. His perspective is that no matter what the vendors say, nothing can convince a skeptical user base that TCPA isn't "evil," therefore any solution must be open source. He noted that the current working document (*http://www.trustedcomputergroup.org*) for the TCG consists of over 320 pages of dense language. Because of this, most people engaged in the debate over TCPA have not actually read the specification. The Trusted Platform Module (TPM, or TCPA chip) upon which the platform is based is essentially an RSA chip, with the key never leaving the chip. A diagram of the chip can be found in the *Linux Journal* article. Safford kept emphasizing that solutions to the trusted computing problem should and will be open source. Further information, including white papers and source code for a TPM device driver, can be found at *http://www.research.ibm.com/gsal/tcpa*.

Prof. William Arbaugh drove home the point that any new technology can have unintended or dual uses, a perspective which placed him fairly middle-of-the-road in the debate. While everyone lumps all trusted computing platform concepts together, in his view this is incorrect. There had been very emotional debate around these platform proposals, namely, concerns voiced by users that they will become "locked-in" to the platform, that free software will be excluded from the playing field, and that the privacy of users will be seriously violated. While a trusted platform could support such usage, the technology alone cannot do these things. The issues raised above result from the policy that the trusted platform would enforce, not from the platform itself. And just as there are potential drawbacks, there are also many potential benefits, such as protected storage and proof of configuration to a third party. Ultimately, users should have the choice of what kind of policy to use – and a policy language and interface that enables them to understand the policy choices they make.

Douglas Barnes gave his perception of what life would be like once a TCPA-like platform became prevalent. In his view, the technology will amplify existing market power, while ending the sense of "ownership" users have come to feel over content and software. He also views any espoused "choices" as an illusion, since third-party validation cannot be done with just anyone if the second party (namely, the one constructing the TCPA) does not allow the user to communicate with them. To ensure that this is not what the future of trusted computing looks like, Barnes urged users to demand that their own benefits take precedence over the more profit-driven benefits to be reaped by commercial parties. Users should also not feel pressured to buy until proper safeguards and assurances are in place. The discussion that followed was mainly fueled by questions around who will ultimately control how a trusted platform is utilized – the vendors or the user? Repeatedly questioned as to how we can be sure that Microsoft won't "do it," interpreted to mean that the company will not suddenly limit the types of policies that can be implemented on NGSCB, Biddle stated that Microsoft would be bound by the contracts it had entered into regarding the NGSCB. They would likely be bound by the most strict of the contracts they had signed. Pressed on development aspects of the NGSCB project, Biddle provided some details. Microsoft is writing the code in C and has people doing formal methods proofs for the project. They will not prove the entire code but will evaluate key components, such as the memory management module. Additionally, the team maintains two development trees, one for code yet to be examined and validated and one for production. The development team is also utilizing tools that do not allow changes to the code interface without making the appropriate changes to the specification. Biddle emphasized that if they do not end up with a system that is comprehensible by a single individual, they will consider the project a failure.

Another interesting point brought to light was how humans interact with security. Most panelists agreed that it is a problem without a good solution at this point. Some technical details of the TCPA were also covered. The panelists noted that the system is not designed to be secure against a physical attack. Safford said that IBM's goal is to get the TCPA platform to interact with virtual machines. Many in the audience were also concerned about how third-party software would be able to run on the platform. Biddle said that the default setting would be to allow any software to run. This can then be configured by the user to narrow down the kinds of credentials the user accepts. His comments indicated that the issue of how a small software developer can "certify" their software comes down to how the majority of users configure their security policies under the TCPA.

REFEREED PAPERS: HARDENING II

Summarized by Clif Flynt

PREVENTING PRIVILEGE ESCALATION

Niels Provos, Peter Honeyman, University of Michigan; Markus Friedl, GeNUA mbH

Two problems with secure computing are bad design and bad implementation. Even with a good, secure design, a program that has a buffer overrun or other implementation error can be compromised. If that program runs in a privileged state, it opens the door for all kinds of trouble. Provos described an application architecture that reduces the potential for trouble by separating those sections of an application that require privileges from the rest of the application. This reduces both the amount of code that must be carefully examined to create a secure application and the impact of implementation errors in the application. For example, applications that perform user validation require access to files that users are not allowed to read (/etc/shadow) and thus must be run as SUID root.

Provos described how the OpenSSH application can be split into a small monitor section that runs with privileges and a larger slave section that performs most of the interaction with a user. The downside to this is that there is no automated way to split an application into privileged and unprivileged sections. Provos' implementation of this concept separates the monitor and slave into two applications, which use a socket to exchange requests and data. He explained that processes under UNIX are protected entities, and only the owner can send signals, debug, or otherwise interact with a process. Requests a slave may send to the monitor include Information (request challenge, provide response from remote, monitor compares) and Capabilities (may access file system for slave). When a slave dies and a monitor creates a new slave with a new ID, there may be state information that must be saved and loaded into the new slave. In this case, the slave exports state to the parent before terminating, and the new slave imports state without the monitor evaluating the data. In practice, exporting the state is pretty messy. This uses XDR-like data marshaling for Global Structures; to handle dynamically allocated state, the application uses shared memory. The master keeps the shared memory open and lets a new slave attach to shared memory. Provos noted that this required a new malloc to allocate memory in shared mem space instead of normal heap. The reimplementation of OpenSSH using this technique included a list of permitted requests; if a slave's request is unrecognized, the master immediately terminates the slave. In this implementation, the slave owns the SSH session socket and sends Information requests to the master for server signature, testing password validity, a challenge to send the SSH client, and to check the SSH client response. Thus, all determination of identity of client is done in the monitor; if a malicious user can manage to compromise the slave with a buffer overrun or a similar attack, the application won't allow malicious login.

Improving Host Security with System Call Policies

Niels Provos, University of Michigan

Niels Provos discussed techniques for preventing applications from performing unexpected actions by reducing their access to system library functions. He pointed out that it's not possible for anyone to have intimate knowledge of all the applications running on their computer. Even given source code, and assuming enough time to examine it completely, there's no guarantee that the running application is actually compiled from that code. We must assume that any vulnerability in a system is known to an attacker, even if it's not known to the user. Any damage to a system must be done through the system call library. Intercepting these calls will prevent a malicious program from modifying a file system, overwriting memory, invoking other programs, or other unpleasant actions. Provos's work resulted in the Systrace system, which catches all system function calls on the fly and determines which calls are allowed (or require permission) to an application. When using Systrace, no applications need to be run with root privileges (SUID root). Instead, applications that require privilege escalation to invoke system functions at a superuser level are granted access to the only necessary subset of the system functions, without exposing other functions to possible abuse. Provos solved the problem of rule complexity by putting a user-friendly front end on Systrace that pops up a query when an application tries to do something the rules don't allow. If the user is willing to allow this operation, a new rule is generated and added to the rule set. With this technique, one can start with a fully locked down system that can't be used for anything, and develop a set of rules that permit work to be done quickly. The Systrace system is implemented as a hybrid of kernel and user-space functions to provide for portability and efficiency. Provos notes that a good policy is one that allows only the actions necessary for the intended functionality of the application and denies everything else. We can generate policies automatically, or a user can define policies interactively via a GUI dialog. In practice, he found that policies with a subset defined, and dialogs when something unknown comes in, converge to good policy fairly quickly.

INVITED TALK: THE INTERNET IS TOO SECURE ALREADY

Eric Rescorla, RTFM, Inc.

Summarized by David Molnar

Note: Slides from the talk are available at *http://www.rtfm.com/TooSecure-usenix.pdf*.

Despite nearly two decades of open research in security and cryptography, the state of real-world system security does not seem to have improved much. Why is this and what can we do about it? Eric Rescorla placed the responsibility for this state of affairs on an incorrect threat model and lack of attention to real-world security problems. Practical software security is much less glamorous than coming up with a new cryptographic hole. As a result, perhaps we are focusing on the wrong areas of research for practical protocols.

The rest of the talk developed this point with specific examples. The speaker considered the SSH, SSL/TLS, IPSec, PKIX, S/MIME, and WEP protocols. These were divided into "wins," "draws," and "losses." SSL/TLS and SSH count as "wins," IPSec, PKIX, and S/Mime as "draws," and WEP as a security "loss."

In each of these protocols, the threat models assume that endpoints are inviolate and links are completely controlled by an adversary. The speaker then argued that in the real world, most of today's problems come from endpoints compromised by virus, worm, or DDoS attack. Therefore these threat models are not well matched to today's real vulnerabilities in protocols. In particular, the practice of requiring public-key certificates and certificate authorities to protect against man-in-the-middle attacks adds significant complexity to deployment.

In SSL, this complexity is mitigated because only servers need certificates, but in the case of encrypted email, the requirement seems to have significantly hindered adoption. The speaker also made the point that perhaps people don't actually want encrypted email, despite the fact that everyone says they want encrypted email and there are plenty of startups past and present for the concept. After all, we are on our third generation (counting from MOSS) or more, and still no one uses it.

The SSH "leap of faith" model, in contrast, was held up as a good example of a usable model that manages to resist most attacks. The speaker noted that SSH key management was considered a very bad idea at the time, but now everyone thinks it's the right way to go!

Rescorla then discussed the difference in credit and glamour for flaws in cryptography and flaws in software quality. He made the point that we will review the entire protocol to deal with one padding attack that has never been implemented, as far as anyone knows, but that we seem to invest comparatively little work in dealing with buffer overflows or usability enhancements. For example, in 1998, Daniel Bleichenbacher found a vulnerability in the padding used by SSL; his name is well known, despite the fact that his attack has never been seen outside the lab. In contrast, the discoverer of the buffer overflow exploited by the Slapper worm, which compromised thousands of servers, is practically unknown. This difference in credit leads to a difference in research emphasis. More recent examples of "theoretical" holes in cryptography that were given great attention included Serge Vaudenay's padding attack on CBC and Phil Rogaway's attack on padding.

Many of these theoretical attacks can be prevented with good "crypto hygiene"; for example, timing attacks are resisted by using the blinding code in OpenSSL. Operational solutions such as this can avoid the need to upgrade the entire protocol to deal with a weakness. This is important because the protocol revision process is time-consuming and often bogs down. Necessary functionality improvements are held up because security issues are considered in the same committee. A secondary point was made about the efficiency (or lack thereof) of standards committees; for example, SSH was introduced to the IETF several years ago and is only now becoming standardized, IKE is two years overdue for a rewrite, etc.

The speaker ended by discussing end-user requirements for security. For example, when they say "security is important," they tend to mean "I need to know what to tell my boss." This led to a discussion of methods for fixing our security research practice and getting market feedback on what is important to fix and what is not. Finally, he left the audience with a list of questions for shaping future research in security.

One audience member commented that he was from the NSA, and that in the classified community they had paid a lot of attention to issues of software quality influencing the reliability of the transaction. The key question is *what* you are relying on the software to do. The audience member commented that attestations of software quality will come to play a greater role in protocols; parties may require certificates stating that the software of the other party has been audited to an acceptable level before entrusting data.

Another audience member commented that the speaker's view of cryptographic considerations in the design of security protocols might be colored by the specific issues of the Internet Engineering Task Force (IETF), where many of the examples originated. The speaker replied that he had seen similar issues outside IETF and in other organizations.

David Larochelle mentioned that SSH also added functionality with regard to xterm forwarding, which helped it dominate its market segment. He then referenced his paper on the subject (*http://www.cpppe.umd.edu/rhsmith3/papers/Final_session3_farahmand.navathe.sharp.enslow.pdf*).

David Molnar asked if the speaker had a good place to learn about crypto hygiene. The speaker replied he was too busy to write a paper on the subject but invited others to do so. "Maybe you can do it."

REFEREED PAPERS: THE ROAD LESS TRAVELED

Summarized by Scott A. Crosby

SCRASH: A SYSTEM FOR GENERATING SECURE CRASH INFORMATION

Peter Broadwell, Matt Harren, and Naveen Sastry, University of California, Berkeley

This presentation dealt with crash dumps. When a program crashes, it may leave behind a dump of its stack, variables, and memory contents to aid in debugging. Many popular programs are now including automated reporting so that users can easily send these dumps to developers. Such dumps are valuable debugging and development aids, but may contain sensitive information such as passwords, bank account numbers, keys, trade secrets, or PINs. Users may want to send these dumps to developers but don't because they don't want to disclose their secrets.

This paper outlines how to make still useful crash dumps available to developers without disclosing user secrets. The approach used is to annotate some of a program's variables as "sensitive" and then to apply a data-flow analysis from CQual to identify all other variables that may become tainted with sensitive data. Those variables are also marked as sensitive. A custom malloc library that supports region-aware allocation is used to store sensitive data in one region, separated from nonsensitive data, which is stored normally. When a crash dump happens, the sensitive region is wiped so that it won't contain any data that was tainted as being sensitive.

In the performance analysis, a micro-benchmark implementing a recursive GCD solver shows a 22% degradation, while the time taken by scp to transfer 100 megabytes of data degraded by 6%. They also tested two interactive programs, the GNOME calendar application and a Palm Pilot synchronization program. For these applications, under 25% of the stack variables were marked as possibly containing sensitive information and no impact on performance was observed.

IMPLEMENTING AND TESTING A VIRUS THROTTLE

Jamie Twycross and Matthew W. Willia, HP Labs, Bristol

Jamie Twycross started this talk by showing the infamous movie of the Sapphire worm infecting tens of thousands of hosts in minutes. He then reminded us that patching systems don't work; patches can be available for months and yet tens of thousands of computers may remain unpatched and infectable. Signature-based worm detection also doesn't work. Worm signatures – e.g., ports they scan – require prior knowledge or active infection to determine. By that time it is too late.

This paper proposes a different scheme: detecting a worm based on its network behavior, in this case, outgoing traffic. The response to detecting adverse behavior is benign – it delays traffic but does not drop it. The detector is based on the idea that most hosts will only regularly contact a small working set of hosts; worms will contact many hosts. If a host attempts to contact a host that is in the working set, the connection goes through immediately. The working set size is set to five hosts. Otherwise, the connection is delayed. The host can assume it is infected if the queue of delayed connections grows too long.

Because connections are delayed, this mechanism does not create false positives, and because it detects scanning behavior, it can work without knowing much about a worm. The authors showed that this scheme takes seconds to stop both simulated and real worms on their test network, and that only a slim portion of their normal desktop traffic was delayed at all.

ESTABLISHING THE GENUINITY OF REMOTE COMPUTER SYSTEMS

Rick Kennell and Leah Jamieson, Purdue University

This won the Best Student Paper award. Rick Kennel started his talk with a pronunciation guide to genuinity, then described why it is desirable. In current systems, forgery is a classic problem. Users can masquerade as other users, computers may be substituted for other computers, and simulators can masquerade as physical machines.

He gave as an extreme example remote NFS clients. NFSv3 depends on client-side trust to be a secure distributed environment. He asked whether clients can be trusted, including remote ones, on an insecure network with ordinary hardware and with no trusted human to look over the remote machine. He then described how an authority could be able to trust remote entities.

The authority can know if the entity is executing the correct code by requesting a checksum over it. Unfortunately, the remote host can lie. The authority can resist this by combining the checksum with a hardware test that can detect whether the checksum operation was actually run. Since computers can simulate other computers, a right answer is

Rick Kennell and Vern Paxson

not enough. The authority must find a hardware test that is deterministic, but hard to simulate quickly and accurately, such as implicit parallelism, data-cache updates, or other aspects of the memory hierarchy. Once such a test is found, its response is regularly included as part of the checksum. The authority checks to make sure that the checksum is both correct and timely. Any attempt to virtualize hardware or alter the program is almost guaranteed to alter those hardware values and create a bad checksum.

INVITED TALK: THE CASE FOR ASSURANCE IN SECURITY PRODUCTS

Brian Snow, National Security Agency

Summarized by Catherine Dodge

Brian Snow spoke about the need for greater assurance in commercial off-the-shelf products. Snow discussed two distinct threat categories: generic and targeted. A generic threat, for example, is one posed by a burglar who wants to steal a VCR. The algorithm for such a task is relatively simple: find house; if house is dark, check door; if door is unlocked, go in; if locked, try next house. The subject of a generic attack must only invest in enough defense to make it cheaper and easier for the burglar to rob the neighbors. On the other hand, if a burglar were after your Picasso, the algorithm would become much more sophisticated: find house; if dark, check door; if door is unlocked, go in; if locked, break window and go in; if alarm visible on windows, rip off siding, punch through gypsum, rip out insulation, punch through dry wall and go in; if motion detectors present, cut phone line that dials the police upon alarm and go in via the above methods. The point is that defense against a well-funded, determined adversary is difficult and must be similarly well thought out and well funded. While the NSA would like to be able to recommend commercial products to its Department of Defense customers, Snow expressed his frustration that "I still haven't found what I'm looking for" and wondered "when will I be secure, nobody knows for sure" (via a video of song commissioned by Jim Bidzos for RSA '99, to the tune of the U2 song by the same name). Snow likened the state of today's computer security to cars built in the '30s, with no anti-lock brakes, no air bags, not even seat belts. Assurance, like an air bag, while generally invisible to the user, becomes prominent on failure. Snow extended the car analogy and recognized that there are not enough customers for GM to build tanks to sell at their dealerships. Yet he does wish that commercial products at least had the same level of security as a car built in 2003 does. While popular wisdom often says otherwise, Snow argued that quality and reliability can sell software. The example he gave was how Toyota and Honda moved into the US car market in the '70s, based solely on such attributes. Yet this kind of assurance cannot simply be slapped on top of a product at the end of the development cycle. Software engineering too often assumes a benign environment. Developers must move beyond this assumption and factor in malice at design time. Above all, Snow was concerned that in another 10 years we'll have more features but not more assurance, a state of affairs which led him to exclaim, "Am I depressed? Yes!" These features may include the ability to counter many hacker attacks, yet targeted attacks will still be overwhelmingly successful. "We will think we are secure, but we will not be."

After finishing this self-described "polemic," Snow outlined six areas in which vendors can address his concerns: operating systems, software modules, hardware features, systems engineering, third-party testing, and legal constraints.

Snow advocated improvements in operating systems, such as digital signature checks on modules prior to execution, utilization of the principle of least privilege, and enforcement of security policies. He also sees room for improvement in software modules. They need to be well documented, be created in certified development environments, and use separate design and review teams. Snow pointed out that a contractor with uncleared foreign national employees could apply good principles by having a German team design the project and a French team review it. Included in this process should be some degree of formal methods, as the tools to support their usage have gotten better. Hardware features are also essential to creating assurance, since they can create isolation in a way that software cannot. The major issue with hardware then becomes trust in the person you are purchasing from. Additionally, systems engineering is a key component of assurance. Questions that need to be answered in this arena include: How can products of unknown quality be used safely? How can components be composed in such a way that the assurance level is a synergistic result of the parts, and not be reduced to a "weakest link in the chain" situation? Third-party testing is another area that can help address assurance, since it provides something beyond "emphatic assertion" that a product provides the desired level of security. A current major failing of such certification processes is that the results are not easily understood by users. Ultimately, though, it may be legal constraints that convince software companies to invest in assurance. Establishing a fitness-for-use criteria and liability beyond the media that the software is distributed on would go a long way toward raising the bar in the software industry. And more insurers may do what some have already done, namely, charge different rates to insure a company based on which operating system and applications they use. Whether these changes really do come about depends on these three aspects: training, research, and assurance built in from the ground up.

Someone asked how those who develop free software can have their products certified, in light of the considerable expense this requires. Snow replied that there is no way of getting around the fact that a certification will cost money. He thought the best path for free software would be to create a foundation which would raise money specifically for funding the certification of such software projects.

WORK-IN-PROGRESS REPORTS

Summarized by David Molnar

Analysis of Electronic Voting System

Adam Stubblefield

Note: For more information, go to *http://www.avirubin.com/vote.*

The speaker discussed his and Avi Rubin's experiences auditing the source code of a widely used voting machine. Among the issues they discovered were that key management is handled by a single #define DES_KEY "XXXX" in the code, where "XXXX" is an ASCII string; this key appears to be the same across all instances of the machine. Another issue was that the code did not appear to have been audited at all for correctness or security; long stretches of code appear without any comments whatsoever. Audit features in the code were extremely weak; while changes do appear in an audit log, nothing seems to protect the log itself from being changed. In response to a write-up on these issues, the company claimed that the "voting system works exactly as designed."

One audience member asked if source code was still available. The speaker replied that it was the last time he checked.

Validator – Testing Firewalls

Clif Flynt

Flynt's framework for firewall testing, Validator, allows for defining "interactions" and "monitors" for firewalls. The framework also allows for the creation of "golems," autonomous agents that take scripted actions for firewall testing, and "probes," which test firewalls with known good or known bad packet traces.

The system is in late beta; it consists mostly of expect scripts. Interested parties are invited to contact the author at *clif@cflynt.com.*

Link Cut Attack

Steve Bellovin

Note: A draft version of the associated paper is available at *http://www.research.att.com/~smb/papers/reroute.ps*. As of this writing, the paper stated, "Please do not mirror or distribute."

The speaker opened by pointing out how we are close to deploying secure BGP and secure OSPF. These protocols address attacks where nodes lie to each other in advertisements about what routes they can provide. Unfortunately, there is an attack that these protocols will not address: cutting links to force routing protocols to route through adversary-controlled nodes.

Suppose that the adversary controls a node N that is in the same network as two communicating nodes S and T and that N is not initially on the path selected for routing between S and T. Further, suppose the adversary can cut a limited number of edges in the graph. Then the adversary can use some graph theory to figure out the edges to cut such that the resulting shortest path between S and T runs through the compromised node N. The graph theory is fairly simple, and the resulting algorithm runs quickly on graphs obtained from real network topologies. Because the attack changes the real topology of the network, mechanisms for preventing false advertisement have no effect. A worked example was shown in which an adversary caused the shortest path between two network nodes to run through the node it controlled.

Stream

Simson L. Garfinkel

URL: *http://stream.simson.net/*

The speaker raised the question, "Why, after all these years, do we not yet have encrypted email?" The answer lies in the cumbersome key management for PGP and the impossibility of obtaining S/MIME certificates. Therefore, the speaker created a new opportunistic encryption layer called Stream. The aim is for deployment as transparently and as widely as possible. Keys are placed in the headers, where they aren't noticed by mail agents. The speaker showed a plug in running with Eudora that talked to a Stream proxy responsible for key management and encryption.

One audience member asked if the software would be freely available. The speaker responded that it could be downloaded right now from his Web page. Precompiled binaries are available for FreeBSD, Linux, and Mac OS X; source is also available.

A Framework for Receipt Issuing, Contendable Remote Poll-Site Voting

Prashanth Bungale

The speaker presented a new design for an electronic voting system. A key feature of this voting system is receipt-freeness while maintaining voter verifiability. Voters can verify that their vote was cast correctly but cannot prove which way they voted to a third party. This prevents them from selling their vote and then making good on the promise. In addition, voters can vote at any poll site. Unlike previous receipt-free solutions, the framework allows a voter to contest his or her vote and ensure that it was counted correctly. The framework also allows for multiple independent tallying agents.

One audience member asked what the provisions were for a third party, such as a candidate, to contest a user's vote without their consent. The speaker replied that such a party can act as a tallier (because any number of talliers are possible) and tally votes according to their own policy.

WLAN Location Sensing for Security Applications

Algis Rudys

Note: The associated paper is available at *http://www.cs.rice.edu/~arudys/papers/wise2003.html.*

At USENIX '01, someone mounted an active attack and took over the wireless network. People who tried to connect were rerouted to a Web page proclaiming "ALL YOUR WAP BELONG TO US." What if we could track the perpetrator by triangulating signal strength? Thanks to Algis Rudys and his collaborators, now you can!

Their implementation can handle both single and multiple adversaries. The speaker showed a jitter diagram and claimed that even in the presence of multiple adversaries their implementation can get location claims within 3.5 meters. A full version will be presented at WISE 2003.

Trends in DoS Attacks

Jose Nazario

URL: *http://www.monkey.org/~jose/presentations/ddos.d/*

The speaker detailed the preliminary results of an experiment in monitoring Internet traffic for traces of DoS events. Over the past few years, 117K events were monitored using blackhole collection with a view of roughly 1/256 of the Internet. The monitoring view consisted of targets geographically distributed over the world on a Class A network. Overall, there was a 50–50 split between spoofed and real source addresses. The speaker showed graphs of TCP vs. UDP traffic year by year, and noted that there has been a dramatic shift: In 2003 nearly all DoS traffic has been UDP with very little TCP, while the opposite was true in 2002. The presentation also showed a graph of durations of DoS attacks.

One audience member asked if the monitor network was a contiguous Class A network. The answer was yes, contiguous and globally announced.

Another audience member asked about trends in spoofed source addresses; how many are spoofed vs. real? The speaker replied that there was a general trend toward DoS adversaries noticing when they are monitored. As a result, the efficiency of the method was beginning to decrease.

SonicKey

Greg Rose

The speaker demonstrated a system in which a public key, challenge-response, or other data is encoded as a sound playable over a phone or computer speaker. With this system, when you want to send your public key to someone, you just play it for them; their computer or phone receives it via the microphone, decodes it, and obtains your public key. A similar mechanism can be used to allow the phone to answer challenge-response queries from a PC equipped with speaker and microphone. The speaker is employed by Qualcomm, a major cellular phone software company – the SonicKey system is in use at the Qualcomm head office for access control to the building.

One audience member asked whether transaction details were necessary before creating a response. The speaker replied that the system could be deployed just like SecureID.

Another audience member asked about the range of the system. The speaker replied that it was spread spectrum, with a range of 8–9 feet.

Someone asked whether there was dedicated hardware in the phone. The speaker replied that the prototype platform phone does have some dedicated hardware for speeding up RSA public-key crypto operations, but not specifically for the encoding itself.

Another audience member asked whether you can do decoding in the handset. The speaker replied that this was for the next phone generation.

Rekeying

David Molnar

Slides: *http://www.cs.berkeley.edu/~dmolnar/tiny-rekey-wip.ppt*

The speaker gave a short overview of the Berkeley mote sensor network platform and the TinySEC mechanism for link layer authentication and encryption in mica2. The main problem with TinySEC is that keys are baked into motes at code image flash time and cannot be changed afterwards. The speaker presented a protocol for rekeying these sensor nodes and outlined its applications in implementing "secure transient associations."

Honeyfarm

Nicholas C. Weaver

URL: *http://www.cs.berkeley.edu/~nweaver/*

The speaker addressed the question of automatic detection of fast worms. After seeing Sapphire, Slammer, and other quick-spreading worms, we know that it really is possible to own the Internet in 15 minutes. What can we do about this? The speaker introduced a "honeyfarm" as a method for automatic detection of quick-spreading worms. In a honeyfarm, multiple machines running honeyd can act as targets for quick worms and provide researchers with early warning and analysis traces. The speaker also outlined the idea of "wormholes," or network segments designed to draw and contain worm activity. The goal of such a system is to give researchers early warning and, more important, provide a platform for automatic response to new worms.

One audience member asked how many distinct physical machines were in the honeyfarm. The speaker replied that he had about 1000 endpoints in the current deployment.

Honeyd

Niels Provos

http://www.citi.umich.edu/u/provos/honeyd/

The speaker gave an update on the honeyd virtual network host daemon. A machine running honeyd looks like multiple hosts on a network; adversaries probing a network will attempt to attack these hosts, leading to valuable analysis traces. More work is needed, so potential users and volunteers should check out the honeyd page.

DoS Through Regexps

Scott Crosby

URL: *http://www.cs.rice.edu/~scrosby/hash/slides/USENIX-RegexpWIP.2.ppt*

The speaker introduced the idea of causing denial of service by forcing applications to take a long time matching inputs to regular expressions; this is an extension of the idea presented by the speaker and others in a regular conference paper on "Algorithmic Denial of Service Attacks." The key here is that instead of converting regexps to Nondeterministic Finite Automata (NFA), determinizing to form Deterministic Finite Automata (DFA), and then running, many implementations simply attempt to match regexps nondeterministically directly. In naive implementations, this can lead to exponential time. Perl has optimizations that claim to limit the running time, but it's not clear whether these are correct.

The speaker then gave a simple example of a regexp that could take exponential time, namely a*[ab]*O, and walked through the process by which the regexp parsing could take this much time. Then the speaker gave an example of a regexp "in the wild" that could take exponential time in the worst case – this one from SpamAssassin. The key issue was the number of "." in the regexp. In theory a spammer could use this to cause a DoS attack on individuals using SpamAssassin.

Someone asked if you could avoid the problem by converting the regexp to an NFA and parsing from the equivalent DFA. The speaker noted that the conversion to DFA introduces an exponential blowup in the size of the state machine.

Readers who want more background on NFAs and DFAs may want to look at Lewis and Papadimitriou's *Elements of the Theory of Computation* for a theoretical perspective. For a practical perspective on regular expressions and their matching, readers may wish to consult Freidl's O'Reilly book *Mastering Regular Expressions.*

Porting Trusted BSD to Darwin

The speaker discussed Apple's approach to migrating mandatory access control and other security features from recent BSDs to Darwin, and the challenges involved. These features have also appeared in systems such as SELinux and Flask/TE. The presentation provided a short overview of the Darwin architecture: Darwin = NextStep, recent Mach (monolithic Mach), parts of FreeBSD 3 and 4, and IOKit. The speaker then went over new security features added to BSD since the creation of Darwin, such as sbuf. One of the key issues involved is the treatment of IPC and threaded processes; Darwin manages a mapping between BSD and Mach notions of processes, and some of the work requires working around the Mach IPC model.

Still Cleartext After All These Years

This talk was a follow-up to Dug Song's presentation at the USENIX Security WiP session in 2001. At that talk, Song showed that many conference attendees were divulging passwords over the open 802.11b network. The presenter asked the question, "Have we learned anything since then?" After all, USENIX Security is supposed to be a gathering of the best of the best security experts. Are we practicing what we preach in security? The speaker mounted active and passive attacks on the (open, unencrypted) conference wireless network. Tools used included dsniff, honeyd, and netics. In addition, the speaker worked with Niels Provos to create an "SSH mirage" – a fake AP that trapped SSH connections and attempted to mount a man-in-the-middle attack on unwary conference goers.

After discussing the methods, the speaker posted a slide full of passwords. "If you see your password up here, you might want to change it." No usernames were provided at that time, to allow victims to change their passwords. Unencrypted Web authentication provided many of the passwords, as well as AOL Instant Messenger and other IM clients.

In a point of good news, no Telnet traffic was observed, only SSH. In addition, only one conference attendee fell for the SSH mirage; the others heeded the warning "HOST KEY HAS CHANGED" and thought better of it (or maybe people just didn't get close enough to the fake AP?). The speaker's conclusion was that many people, even at USENIX Security, are not yet taking wireless security seriously enough, even though there are some marginal improvements over 2001.

At the end of the talk, one audience member pointed out that encrypted instant messaging is available via Jabber or the Trillian AIM client.