;login:

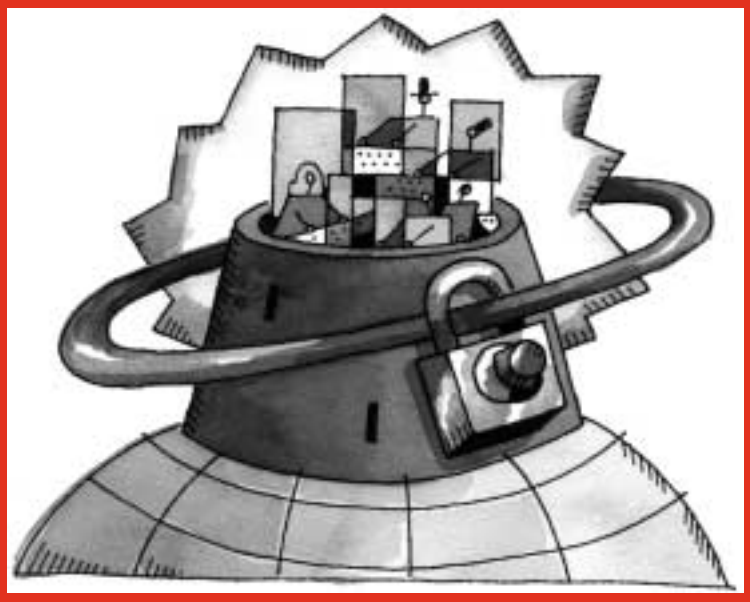
THE MAGAZINE OF USENIX & SAGE

December 2002 • volume 27 • number 6

Focus Issue: Security

Guest Editor: Rik Farrow

inside:



USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

conference reports

Our thanks to the Summarizers:

Akshay Aggarwal
Mihai Christodorescu
Michael Hohmuth
George M. Jones
Lou Katz
Prem Uppuluri
Haining Wang
Seung Yi

11th USENIX Security Symposium

San Francisco, California, USA

August 5-9, 2002

KEYNOTE ADDRESS

Information Security in the 21st Century

Whitfield Diffie, Sun Microsystems

Summarized by George M. Jones

The opening keynote given by Mr. Diffie provided a jam-packed overview of information security in the 20th century and projections for the 21st century, interspersed with sage opinions and observations.

He began by defining security as (1) preventing adverse consequences from illegitimate actions of human beings; (2) protecting yourself against the actions of an intelligent opponent; and (3) something that gives you the appearance of legitimacy.

The history of information security in the 20th century was largely dominated by issues of privacy, with cryptography being the primary tool to enforce privacy. While cryptography has been around at least since the days of Julius Caesar, its importance became crucial with the advent of a new communications technology: radio. "Radio revolutionized warfare. Before radio, a naval fleet commander sent ships out with orders and could communicate with them every few weeks or months at best. With the advent of radio, orders could be communicated in, at most, days. *But* since radio is a broadcast medium, everyone could listen – hence the importance of cryptography for confidentiality."

Accordingly, WWI saw an increase in the use of cryptography, and WWII, an increase in the use of automation (the code clerks just could not keep up). Ensuing decades brought computerization ('50s); reliable computers, time-sharing, and the first "computing communities" ('60s); the advent of the ARPANET and the loss of all the (dis)advantages of locality ('70s); disconnected, password-less PCs ('80s); and the Internet brought to all those password-less PCs (oops), the "computerization of everything," and the migration away from paper ('90s).

Another important focus of information security in the 20th century was the efforts to provide secure voice communication, from the '40s, when exactly two people (Roosevelt and Churchill) could communicate securely using multi-million-dollar 30-ton devices, to the STU phones of the '80s, which "reached their goals but failed because communications expanded beyond the phone (cell phones, voice over IP, PDAs, fax, email, WWW)."

Diffie outlined some trends and observations from the 20th century: computer power keeps increasing, information is now digital, security technology moves closer to the user, DES was developed in secret, AES was developed in public. In the shift to elliptic-curve cryptography, "we're now moving from using 17th-century to 19th-century mathematics." Encryption allows networks to be defined by who has what keys, not by topology (à la today's firewalls). "The minute you begin rolling out crypto, you turn everything into 'us' vs. 'them'."

Some current trends: computer-mediated communication, the rise of the information economy, unification of communication and delivery channels (e.g., Web site download of programs), mobility, and bandwidth on demand. "*The* driving factor is that better security technology draws more valuable traffic, and, conversely, more valuable traffic requires better security."

Diffie had some insightful observations on privacy: "Privacy is a security policy

about personal information. If you don't have any way of controlling information flow, you have no way of enforcing a policy. There is an increasing immediacy to information security. It's important that people be able to recognize each other (authentication) and have private conversations (confidentiality). We are trying to transplant our human culture into a world of computer-mediated communications."

On the "open" vs. "closed" development approach, he noted that "Some argue against this, saying 'open' means the 'bad guys' can look at it. Some argue for it, saying many eyes mean more security. They both miss the point. 'Open' means *you* can look at it and satisfy yourself."

In answer to the question "Why is it taking such a long time to get a working PKI?" he noted that "most of the costs are up front, but most of the benefits accrue once it's deployed. Unlike PCs, it's hard to deploy PKI piecemeal.

"Key escrow is like the One Ring in the *Lord of the Rings*. It is an evil that will be back . . . though perhaps under different names." Data recovery keys are valuable to data owners to ensure the ability to recover private data.

"Today, flows of information are controlled by the movement of people, e.g., it's cheaper to hire away a Microsoft employee than to gain certain information by other means."

Security is about people. "It is never independent of point of view. It often deals with competing interests. It is never value-neutral. It moves power from one group to another."

In closing, Diffie offered the following thoughts: "Security should make doing business easier, not harder. *Nothing* is more important than human-factors engineering. Quality of security is directly proportional to quality of evaluation." Simplicity is the essence of evaluable security design. There are big gains to be had from putting some functions in hardware.

"The question for the 21st century is, 'Can everyone be secure at the same time?'"

Questions and Answers

Q: (Steve Bellovin) Most of the problems we're seeing are not crypto problems e.g., buffer overruns, etc.

A: The fact that we can't implement things right is our Achilles' heel.

Q: (John Ioannidis) What about attempts to legislate security out of existence?

A: My prejudices agree with yours. Security is just one piece in a larger puzzle. Information is becoming a commodity. Societies have always regulated commodities. Decisions made today will shape society for decades.

INVITED TALKS

Wireless Access Point Mapping

Simon D. Byers, AT&T Labs–Research

Summarized by Akshay Aggarwal

Wireless is appearing almost everywhere and comes with no strings attached, literally. Simon Byers spoke about his experiences in wireless access point (AP) mapping. He started off by pointing out the pervasive nature of 802.11b-based wireless LANs, stating that they could be found in your neighborhood McDonald's, Trader Joe's, or just about anywhere. Many laptops now come with built-in support for these networks. "Executives love it," said Byers while illustrating that it was as easy to use wireless LANs in corporate boardrooms as in the female restroom of AT&T Labs' Florham Park, NJ, facility where he works. This property fields wireless LANs as a probable ISP medium, which could solve the last-mile problem.

The wireless LAN protocol uses the free 2.4GHz range and cannot penetrate stones, leaves, or people, though Tupperware is unable to stop it. At a speed of 11Mbps, wireless networks are fast and can be used as access points, relay, or point-to-point links and are "minimally invasive apart from heating up the children a bit." Satellite networks, the only alternative to wireless networks, suffer from poor upload rates and high latency problems. Speaking about the "security" of 802.11b, he pointed to the plethora of literature and scripts available for anyone to break WEP. The end result, according to him, is that WEP is now next to worthless.

According to Byers, open networks exist with the aim of providing free Internet to the people. Some important issues with open networks are the pushback from ISPs, with cable companies persecuting NAT users, patchy coverage, and their susceptibility to DDoS attacks. The reasons to map these networks included the need for a security survey, to find an open network to connect to, to provide and assess network coverage, and to explore the saturation of the free spectrum. To emphasize his point, he gave the example of a war-driving contest at a recently concluded hacker conference (and the basic flaws in the contest). He then showed slides of his mapping efforts made while driving around Las Vegas and New York.

The audience was acquainted with the hardware needed for WAP mapping. He showed them 802.11b wireless cards, the various kinds of antennae (yagi, omni, panel, and dish), and GPS systems and amplifiers. Then came a tutorial on how to build a base station and receiver to capture images from X10 wireless cameras, deployed with the tagline, "You'll never know what you will see!"

While he was driving around Manhattan he found approximately 4000 access points. Of these, 964 had WEP enabled, 156 networks had the default SSID, and many had their addresses as SSIDs. In correlation with other data, analysis of this data provides the location of APs and the comparative reach of the networks. Techniques used to locate APs include max signal-to-noise ratio, triangulation, intersecting spheres, or just the

plain old telephone book in cases where SSID was an address.

To conclude his talk, Byers discussed the business application for mapping APs, which were to set up, manage, and analyze a network for use by all. This included optimizing the deployment and mapping the target customers' footprint. This information would be invaluable to owners of wireless networks.

Freedom to Tinker

Edward W. Felten, Princeton University

Summarized by George M. Jones

Professor Ed Felten of Princeton spent some time this year thinking about the legal and economic aspects of "The Right to Tinker." This follows the presentation last year of his paper on the SDMI challenge (detecting/removing digital watermarks on audio samples), which followed a lawsuit backed by USENIX and EFF to defend his right to present it. This talk outlines some of his conclusions.

"A funny thing's happened in my career," he began. "I've gotten involved in legal issues, or to put it more accurately, those issues have gotten involved with me. Things computer science people have always done are increasingly at risk of becoming illegal. Tinkering benefits everyone, not just techies. We need to sell the idea that the public will lose out as the freedom to tinker is eroded."

"The freedom to tinker," Felten said, "is the freedom to understand, discuss, repair, and modify technological devices that you own."

Felten said that three points need to be stressed:

1. Tinkering is socially important.

Tinkering is rooted in the basic human need to explore and understand the world around us and to control our surroundings. Imagine laws making it illegal to fix your own car. Tools are important to tinkering. Sledgehammers and program debuggers have legitimate and illegitimate uses. Laws such as the Digital Millennium Copyright Act (DMCA) are making useful tools illegal without regard to potential legal uses. Tinkering with products by security researchers benefits the public by disclosing flaws in products they rely on. Tinkering benefits vendors by giving them the opportunity to fix the flaws.

"Increasingly, technology [computers] is controlling access to content. It's no longer just you and the book. Now it's you and your Web browser, and Google, and thousands of Web servers backended by databases connected to networks," said Felten. Tinkering with these technologies should be protected.

Public policy debates often turn on the understanding of technical issues: for example, is a large software vendor simply designing more efficient programs or programs intended to limit competition? Tinkering by independent analysts raises understanding and thus raises the level of public debate.

2. Tinkering is economically efficient.

Most arguments against tinkering boil down to economics, but it is not clear that the arguments are valid when applying generally accepted principles of economic analysis.

Tinkering has many positive side effects (or "externalities"). They include innovation, education, and competition.

If there are barriers to tinkering, such as the DMCA or restrictive End User License Agreements (EULAs), not enough tinkering will occur and the positive side effects will be missed.

3. Tinkering doesn't conflict with "intellectual property."

"Intellectual property is not a single thing [under US law]. It is a combination of copyright, patent, and trade secrets."

Trade secrets protect secret material, but only against disclosure by improper means (bribery, threat, theft). It does not protect what is learned through tinkering or "obvious" things such as hair color.

Copyright and patent are intended "to promote the progress of science and the useful Arts," to maximize total (not individual/corporate) wealth, and to prevent outright copying of a product, but not to prevent study or discussion. None of these should present a barrier to tinkering.

"Our opponents say that the battle is between people who are pro-copyright (them) and anti-copyright (us). We don't have to accept that. Our position should be that we respect the traditional scope of copyright; fair use is important but is not the issue. Laws such as the DMCA do harm to people (tinkerers, the general public) who have no intention to violate copyright. It's about maintaining robust, open, competitive technology."

For more info, see *http://www.freedom-to-tinker.com*.

Questions and Answers

Tinkering was needed to facilitate the first question; the audience microphones didn't work. The techies present fixed them, with no help or permission from the vendor or Congress.

Q: What alternative is there to the DMCA (technological or other)? What can we do to prevent/deter infringement of copyright?

A: The DMCA is the worst of both worlds. It does not prevent infringement and punishes those who have no intent to violate copyright. It goes beyond what is needed to prevent infringement. The main effect of the DMCA has been to cause collateral damage.

Q: Would you be in favor of building a tool whose sole purpose is to circumvent infringement?

A: No.

Q: Does this change things from civil to criminal? What about the standard of evidence?

A: DMCA increases the number of parties who can bring a suit. Anybody who is harmed can bring suit. This is the source of a chilling effect. You as a researcher don't know who might be offended/harmed/bring suit. The incentive is to do nothing (not to tinker).

Q: Are there some circumstances where anti-tinkering terms of use can benefit users? For example, pop-up advertising paying for free network access. Is it OK to prevent tinkering to turn off pop-up advertising?

A: If your question is how to change the law and policy to encourage tinkering, this (repealing the DMCA) is the only way.

Q: Napster did have legitimate uses.

A: Napster had too much of a role in the infringement.

Q: Do you have suggestions of practical things that people can do?

A: Participate in forums such as this [USENIX]. Try to influence/talk to reps. Get involved with EFF. Be vocal.

Q: What about obfuscation that raises the cost of tinkering?

A: What's really dangerous are mandates that require people to build in anti-tampering devices.

Q: What about EULAs and the Uniform Computer Information Transactions Act (UCITA)?"

A: UCITA would strengthen EULAs. An important step is to say that licenses should not be used to prevent tinkering.

Q: Do you see any problem with the use of the term "tinkering"? Will people whose primary concerns in life revolve around junk food and big-screen TVs take it as a serious issue?

A: Lots of people like to tinker. Recall the Thomas Edison stories. It's possible to connect the issue to things that concern the general public (e.g., a better way to use your VCR).

Biometric Authentication Technologies: Hype Meets the Test Results

James L. Wayman, Biometric Test Center, San Jose State University

Summarized by Akshay Aggarwal

What exactly is hyperbole? Jim Wayman pointed out that the Merriam-Webster dictionary defines it as an "extravagant exaggeration ('mile-high ice-cream cones')." Much of the hype surrounding biometric identification is just that – an exaggeration of the truth. To illustrate this point further he referred to two Web sites and proceeded to expose their exaggerations.

The first Web site belonged to an unnamed biometric product vendor. Their claims:

1. "Facial recognition technology is the only biometric capable of identifying known people at a distance." This is contradictory to the fact that DARPA is involved in a project aimed at using iris-scanning technology at a distance. Facial recognition is not the only biometric available for long-distance recognition, though it is one of them. In addition, the vendor admits that the range of facial-recognition technology is currently limited to 10 feet. So what is really meant by distance?

2. "Facial surveillance can yield instant results, verifying the identity of a suspect instantly and checking through millions of records for possible matches quickly, automatically, and reliably." Furthermore it claims, "These investigative tools help to single out known terrorists or criminals." This implies that the technology is accurate when, in fact, it suffers from fairly high rates of false positives and false negatives.

The second was a leading educational Web site. Their claims:

1. "The biometrics industry is mythical." The International Biometric Association exists and its members can be found at *http://www.ibia.org*; the industry is far from mythical.

2. "Publicly available, independent evaluation of technologies and products is extremely rare." Independent evaluations and standard testing procedures can be found at sites such as *http://www.biometrics.org*, *http://www.afb.org.uk*.

Wayland says, "Hype is factually correct but leaves an impression that may not be accurate." He agrees with B. Miller's definition of biometric authentication as "automatic authentication or identity verification of a living human individual based on behavioral and physiological characteristics."

Wayman says that some metrics that should be used to evaluate technical performance of biometric algorithms are failure-to-enroll, failure-to-acquire, false positives, and false negatives. Failure-to-acquire measures how often the device fails to recognize a metric, such as when a facial-recognition system fails to recognize a face against a pale background. Failure-to-enroll is a more important metric, measuring whether a biometric precludes certain groups of people; for example, fingerprint scanners cannot be effectively used on the old and the very young, groups that tend to have a much less distinct fingerprint. Thus biometrics cannot be used on all segments of society equally.

Current biometric evaluation involves technology, scenario, vulnerability, security, and operational testing. Cost-benefit analysis, environment testing, human perception response, and user attitude also need to be evaluated in the future. Test results are indicative only of people in a particular environment. A hand geometry system when tested at the Sandia Labs and the nearby Kirkland Air Base produced different results in these two biometric environments.

Wayman was asked about the methods used to search through the large databases; he replied that the databases were usually partitioned on the basis of criteria like gender and, further, on some biometric characteristics. In response to a question about the vulnerability testing of such systems, Wayman pointed out that such tests were needed; he gave the example of the inability of a facial recognition system to differentiate between a human face and a photograph.

In conclusion, he reiterated the long road ahead for biometric devices and research.

Network Telescopes: Observing Small or Distant Security Events

David Moore, CAIDA, San Diego Supercomputer Center

Summarized by Lou Katz

David Moore gave an interesting report on experiments with monitoring remote network events through examination of unexpected packets on some address spaces he monitors. This arrangement, a network telescope, uses a portion of the globally routed IP address space on which little or no legitimate traffic is expected. Monitoring the traffic which does arrive gives a view of certain remote events.

An analogy to monitoring with astronomical telescopes helped convey the operation and properties of a network telescope. In network monitoring, a larger address space increases the "lens size" of the network telescope, as does noncontiguous address spaces. Larger network telescopes can see shorter time durations and lower packet rates, and have a larger field of view with better accuracy for start and end times of events (e.g., Code Red spread at about 10 packets/sec). Both Code Red and global DoS attacks could be seen. The data were collected using a passive tap ahead of the net(s) being monitored.

Attackers spoof source addresses randomly, and it is this "backscatter" that is being seen at the telescope. Analysis of the backscatter could give a quantitative measurement of the DoS. Interestingly, the portion of address space monitored can affect the traffic seen, both positively and negatively, since some types of events attempt to preferentially use address spaces adjacent to their source in order to spread. It is not known how randomly these addresses are chosen. In the initial operation of the network telescope, the deployers of the attacks were unaware of the telescope. Later on, there seemed to be evidence of either deliberate avoidance of the telescope-monitored IP space or of attacks on the telescopes themselves.

Detecting an event is a function of the size of the monitored network. An /8 telescope could detect an attack in a minute or two, while a /24 might take 58 days. A /8 network can track an infection accurately, but a /16 has a time lag and the shape of the curve is wrong. On a log plot, the slope for a /16 is OK but the times are wrong. Work on deconvolving a /16 curve into the /8 curve is being pursued.

Conclusions reached so far: there are lots of attacks; some exceed 600,000 packets/sec. Most attacks are short, but there are some that are continuous for over a week. The attacks don't seem to load the network or major peering points, but some embedded devices (routers, printers, etc.) had servers that crashed and had to be rebooted or power-cycled. A steady stream of new packets into the telescope net has been observed at about 20/hr. These are mostly TCP but there are some ICMP floods and some evidence of ICMP black-holing. Eighty percent of attacks last 10 minutes or less. Attacks seem to happen on a human time scale, with peaks at 5 minutes, 10 minutes, 30 minutes, and 8 hours (human control intervals). The victims are mostly commercial businesses, with minor efforts against home machines. There are odd peaks in .ro and .br space, which may be revenge attacks of one ISP upon another.

Code Red spread has been charted by recording machines sending TCP SYN to port 80 of nonexistent machines. Such sending machines are considered to be infected; the data show 359,000 hosts infected in 24 hours. Characteristics of the infection show that 47% of infected hosts have no reverse DNS; there were 136 .mil and 213 .gov hosts infected. Code Red II, by probing local nets, spreads very rapidly on internal nets. Most of the infected hosts were home/small business machines on cable modems.

The reappearance phase of Code Red was also observed; even though there was lots of press coverage – everyone should have known it was coming back, considerable infection occurred. Daily fluctuations were plotted by rough normalization of the IP addresses to time zones. Interestingly enough, at about 9 a.m. every day in every time zone, hosts come up; activity degraded in the evening and on weekends. A great animated map of the world, which showed the spread of Code Red as growing red splotches, was projected. Really scary to see the world mostly turn red in a very short time.

One of the problems with these measurements is that it is difficult to distinguish computers vs. IP addresses. There were a maximum of 180,000 unique IP addresses infected in a two-hour period but 2,000,000 in a week. There is a DHCP effect over long periods. Old computers get new addresses. So far they have not been able to get a good handle on NAT, and it is hard to get a good estimate.

The author concludes that network telescopes can see and give insight into nonlocal events; you don't have to be there, but small telescopes can't see certain types of small events. This is an example of surveillance without a known purpose or target; data are collected first,

and then you work backward after an event. The slides for this presentation should be available on *http://www.caida.org*.

Illusions of Security

Paul Kocher, Cryptography Research, Inc.

Summarized by Lou Katz

Paul Kocher gave an overview of security evaluated from the point of view of a company, such as his, which is focused on cryptography, and of the problems faced by high-risk commercial systems and big companies. The talk was a review of common problems and misconceptions and an exposition of possible rules to live by.

The standard yardstick for measuring cryptographic security, key length, does not really address the problems posed by real adversaries, who lack the propriety to limit themselves to tidy attacks such as brute force, factoring, or differential cryptanalysis. The crux of the problem is that in assessing the security infrastructure, security implies a zero tolerance for flaws in the face of software developer acceptance of bugs proportional to complexity. Since the testing side of system development can't keep up with the complexity of the products, it is often the case that the front door is strong, but it is easy to break in through the window.

In measuring security one must consider the probability of breaking in vs. the cost of the attack. For commercial products there is a negligible probability of being very secure against creative attackers, especially since systems of exponentially increasing complexity are being created, aided by Moore's Law, but security experts are not compensating by becoming exponentially smarter. Is there an upper bound or expected/mean resistance? What is the risk curve, and against whom are we defending? It is important to evaluate what the resistance against an initial attack might be vs. repeated attacks.

In outlining the characteristics of most flaws, complexity and component interactions were among the obvious dangers. When the evaluation of the security of a software system is to be performed, the goal is either to prove that security of the system is bad by finding a flaw, or lacking that, to do an inclusive analysis to assess the likelihood of additional security problems and to advise whether a product is worth deploying. All the while we are faced with the realities that attacking is easier than designing or verifying; and prevention/testing is hard. A very thorough evaluation is expensive, so the constraints on the evaluation process, time, budget, availability and quality of technical information, and evaluator capabilities, experience, and knowledge of the threat model can compromise the results.

Paul posited that the best work is done before the project is started, by careful definition of the target system's security objectives and a review of the implementation details. A checklist of many single points of failure should be developed (he showed an extensive chart of these) along with a long list of reviewable information, such as the open literature, published specs, network and bus I/O, timing, power consumption, defective computations (errors in computation can be used to compromise keys), error messages, failure codes, examination of disk and memory contents, swap files, and RNG seed data. Even chip imaging should be explored. Of course adversaries might engage in illegal/questionable activities such as dumpster diving, so this must also be taken into account.

What you can include in your checklist is to conduct code reviews, which are useful but boring and hard to do in volume. Code review should include algorithms, usage considerations, and protocol analysis, specific details of which were outlined. The increasing connectedness and complexity in the system, a common source of difficulty, increases weaknesses. Exponential growth in transaction volumes means that unusual transactions are hard to break out by hand and lead to an increased use of computers to do the recognition and analysis.

Security is improved when you design for testability, even though testing is expensive. Security design goals to live by were outlined, and their expense and difficulty were not overlooked. Spend money rationally; don't underspend or overspend on security, hire experienced people, and spend early! Avoid what doesn't work – e.g., design by committee, which is flawed by conflicting objectives and no responsibility. Utilize committees later, as they seem to be fine in keeping a design alive after it is done.

Future directions for improving security focus on people. Vendors need to be convinced to spend on prevention. Weak systems, which allow profits from fraud, will lead to more crime, which will fund more crime. Something needs to be done about the moral hazard that there is currently little vendor incentive for security.

Some of the questions focused on the time frame for security resistance to attack – how long after a system is deployed is it usually attacked? (Others may be ahead of them in the attack queue.) Even a flawed system may be stronger than an alternative, or it may not be economically worth attacking compared to others – breakable systems may still be useful.

In summary, this useful talk, rather than providing any specific insights or giving a recipe or checklist for improving security, highlighted many useful and important concepts to consider and evaluate in designing and establishing a system's security.

Formal Methods and Computer Security

John C. Mitchell, Stanford University

Summarized by Mihai Christodorescu

Mr. Mitchell tried to span the existing gap between the formal methods and the computer security communities, because "theoreticians and coders don't talk to each other." The talk described several applications, the different types of formal methods, and their specific strengths and weaknesses.

A formal method is a technique to analyze a system from its description, without putting the system in motion. For example, it means analyzing executable code and trying to ascertain various properties without actually executing the code. In the big picture, formal methods are meant to help to produce good software efficiently: formal methods are precise and automatable, and they usually capture previous experience. There are several current weaknesses: subtleties are hard to formalize and the tools are cumbersome to use. Most of the formal-methods work is now focused on eliminating these weaknesses.

The goal in formal methods research is to reduce the number of unfeasible problems and extend the set of problems and properties that can be checked. Initially, formal methods were applied to hardware verification, as it has a finite number of states. Currently, program verification is the focus of most researchers, but it is not as successful as hardware verification; due to infinite state space, it can only verify simple things about programs. Computer security is itself a subset of type analysis; a well-typed program should not have security flaws.

One of the applications detailed in the talk was the verification of the Java Virtual Machine verifier, which checks Java bytecode after loading into memory and before execution. Since the verifier is the only check performed on the code *before* it runs, it is essential that the verifier itself is correct and implemented according to the specification. To prove the verifier's correctness, abstract instructions were used to reduce the time and space needed in modeling the verifier. The verification of the verifier entailed two phases: verifying the behavior in the verifier specification, and checking the verifier implementation against the specification. In the second step, the research group led by Mr. Mitchell discovered several implementation bugs in the Sun JVM.

Another area of application is protocol security, which looks at simple network protocols (SSL, SSH, authentication, signing) and checks for exploitable flaws. Most of these protocols are fairly simple in their design and involve a limited number of steps. The complexity of unbounded number of states appears when several sessions of the protocol are considered in parallel: the attacker might conduct several parallel sessions and copy messages from one to another. This area of research created several methods, some less formal (cryptographic-based proofs, Communicating Turing Machines) and harder to reuse or automate, and some formal methods (BAN & related logics, operations semantics, automatic theorem proving, symbolic search for an attack, exhaustive finite-state analysis).

Four formal methods were presented in further detail: model checking, multiset rewriting, probabilistic polynomial time, and protocol logic. Model checking was used in proving that contract-signing protocols were fair, noncoercive, and accountable. Examples of such protocols include Asokan-Shoup-Waidner and Garay-Jacobson-MacKenzie.

Multiset rewriting (MSR) is related to mathematical logic and deals with sets of facts known about the system and transition (or rewrite) rules that modify the system and the facts about the system. MSR has a simple tractable model, but it tends to overlook many things, such as initial conditions. On the upside, MSR is accurate: if an error shows up in the MSR model, the error is present in the protocol. This means that MSR can prove security of a protocol up to a certain set of assumptions but that it will not detect attacks that do not follow these assumptions. MSR usually employs a common intruder model, the Dolev-Yao model, that assumes the adversary is non-deterministic and has no partial knowledge (e.g., adversary either has the encryption key or no key at all).

The probabilistic polynomial time (PPoly) formal method applies the concept of observational equivalence: a protocol is secure if the adversary cannot distinguish its trace from a trace of some idealized version of the protocol. This way, PPoly specifies security by comparing the protocol to a zero-knowledge protocol.

In conclusion, formal methods provide very powerful tools for verifying certain security properties. Most useful right now is the checking of a not too complicated property about a not too complicated protocol or piece of code. The goal of formal methods research is to extend the range of feasible analysis, while keeping them automatable.

"How Come We Still Don't Have IPSec, Dammit?"

John Ioannidis, AT&T Labs–Research

Summarized by George M. Jones

The moderator informed us that "John wants this to be a slugfest . . . so reach deep down inside and find your inner Peter Honeyman."

John Ioannidis then told us that we were really getting four or five talks for the price of one: this talk would mostly work as "How come we still don't have {PKI, IPv6, Mobile-IP, DNSSec, secure email}, dammit?"

He started the talk by contradicting his own title: "We sort of do have IPSec . . .

the question is, why isn't anyone using it?" The rest of the talk was structured around a series of interrogatives.

What is IPSec?

IPSec is a network layer security protocol for IP. It means different things to different people. To some, it's just the wire protocols; to others, it also includes key management, GUIs, and tools.

Why IPSec?

IPSec provides end-to-end communications security at the network layer. It addresses authentication, integrity, and confidentiality. It does not address authorization, privacy, non-repudiation, or perfect forward secrecy.

Why network layer?

The network layer is the choke-point. Putting security in the network layers allows both higher and lower layer protocols to use it. "The seven-layer model is a bit of poison left over from OSI."

What are the benefits of IPSec?

Link encryptors become obsolete. IPSec provides link security to applications "for free." Applications don't need to do their own link security. IPSec allows decoupling of security policies and centralization of management.

While IPSec . . .

During the decade-long saga of defining and deploying IPSec, other security technologies sprang up that may not have been necessary if IPSec were deployed. Among these were the Clipper chip (1993), SSL (1995), SSH, firewalls ("bad"), NAT ("very bad"), and layer-4 re-directors.

Why isn't IPSec? Part I

It's taking too long. SSL and SSH removed the urgency. There are many incompatible implementations. There is no agreement on key management. "OK, we'll just deploy it with IPv6. . . ." Other IETF working groups "rolled their own." "It's all a mess."

Where is IPSec?

"Everywhere and nowhere": *BSD, Linux (Free S/WAN), Solaris, Win2K, VPNs, remote access, academic research.

How is IPSec?

The wire protocols are here and work perfectly. IKE still doesn't have interoperability; there are about 8,000 option combinations. There are no standard APIs. Policy support is rudimentary.

Why isn't IPSec? Part II

IKE is too complex to implement. The docs stink. The configuration of key management and policy are smooshed together. There is no good remote key management and distribution. There is no good evangelizing. Ioannidis informed us that "evangelize," in Greek, means "to bring a good message," but do we have a "good message"?

Why isn't IPSec? Part III

We still have problems integrating with RADIUS, Diameter, and Tokens. We don't have a good PKI. Most of the Internet edge is Windows.

<flame-bait>

"Trying to configure IPSec for Windows has been one of the most harrowing experiences of my life, and I live in NYC!!! There is no good command line interface for Windows IPSec. What good is running a secure protocol on an insecure operating system?"

Whither IPSec?

NAT is an abomination. NAT is broken . . . but I can buy a NAT box for less than $100 and plug in lots of hosts with one IP address now. We need to standardize remote access. We need to work on the APIs. We need better configuration management tools – not just pretty GUIs but something that scales to thousands of systems; these tools just don't exist. We need to play nice with other protocols and host routing. We should work on opportunistic IPSec (Free S/WAN). VPNs are going to be the largest user of IPSec for some time to come. Ubiquitous IPSec would challenge the current firewall model by defining "inside" vs. "outside" with keys, not topology (see Bellovin paper of two years ago). But none of this is any use in the face of buffer overflows and viruses. What to do?

Questions and answers ("Let the games begin.")

Ioannidis got his slugfest, thanks to Microsoft (and his own misunderstandings):

(Dan Simon, Microsoft Research) Q: Perhaps the problem is in the wine-glass model. People want to secure things that IPSec doesn't secure. IPSec started out securing everything and wound up securing nothing.

A: Perhaps we need an N-layer shadow security stack with security at each layer . . . but avoid encrypting N times.

(Dave LeBlanc, Microsoft) Q: We are using IPSec on thousands of machines. We find it quite manageable. We're not going around setting it up on every machine.

A: There are these things called standards; maybe you've never heard of them.

LeBlanc: There is a command line interface, RTFM.

A: Send me a pointer. I don't have a language problem.

[Editor's note: Microsoft uses Active Directory to make this work internally. When I asked Ionnidis months later, LeBlanc still had not provided a URL].

Q: Have you heard about IKE2, JFK, other work at IBM?

A: Yes. I'm one of the authors. A smaller protocol with fewer options was one of the goals and results in fewer lines of code, fewer bugs, better security; simplicity of the spec was the driving force. When you see the doc, we hope it will be unambiguous.

Implications of the DMCA Anti-Circumvention for Security, Research, and Innovation

Pam Samuelson, University of California at Berkeley

Summarized by Mihai Christodorescu

Ms. Samuelson presented an overview of the DMCA and its implications for research, focusing specifically on computer security research. The presentation first covered the rules part of the DMCA, followed by actual cases where the DMCA was used, and closed with possible legal alternatives. The DMCA makes illegal the circumvention of technical measures, with several exceptions, and the circumvention of access controls, with no exceptions (not even for fair use). It was noted that Congress enacted the DMCA as a blanket law with exceptions in place, instead of a less restrictive law that would enumerate illegal actions.

The exceptions are very complex and very narrowly defined. The interoperability exception, meant to allow data exchange between programs from various vendors, is present, but with no indication whether circumvention to gain information useful in attaining interoperability is allowed. The exception for cryptographic research imposes several burdens on the researcher: he or she must be a lawful acquirer of encrypted copy, must get permission to research from the copyright owner, and must have a Ph.D.

The DMCA bans the making and distribution of tools that bypass access controls and copy controls, with the exception of reverse-engineering tools necessary for building interoperability. The problem is in determining the boundary between a description of a technique and a tool implementing that technique. It is unclear whether distributing information (through a Web site, for example) on circumventing a given technical measure is "as illegal as" creating and distributing a tool that performs the circumvention. The Ed Felten vs. RIAA case over the watermarking schemes proposed for safeguarding digital music brought up the question of whether presenting a result at a conference is a circumvention device. In the same case, the exception for cryptographic research could not be applied, as watermarks are not usually considered cryptographic research. Thus, Congress might have created an overly narrow exception, but in the current form of the DMCA, it is up to the court to decide what cryptographic research means.

Another point of contention in the DMCA is the definition of access controls. Tools circumventing access controls are illegal to make or distribute. In many cases, the lawyers forced some technical measures to be considered as access control measures, and thus made them illegal to circumvent. For example, the region-coding of DVDs or the encoding of console games for certain markets are technical measures meant to control the market – these measures overreach and prevent owners of legal copies to use them as they wish (a US citizen cannot play games bought in Japan). The effect is not only limiting to users of the technology but also to competing technologies. *Sony v. Connectix* and *Sony v. GameMaster* illustrated how access controls (e.g., country codes) can be used in an anti-competitive fashion to shut down competing products that bypass access controls, even without allowing piracy.

In the various cases where DMCA was applied (*RIAA v. Felten, US v. Sklyarov, HP v. SnoSoft, Microsoft v. Huan, Edelmen v. N2H2, Sony v. Connectix v. Bleem, Sony v. GameMaster, RealNetworks v. Streambox, Universal v. Corley, DeCSS*), mixed results have emerged from the courts' interpretations of the law. On one hand, the courts have decided programs were protected as speech by the First Amendment, regardless of the form of the program (source or object code). On the downside, fair use rules were not considered applicable to tools that allow both good and bad uses (e.g., DeCSS can be used to play legally acquired DVDs on Linux but can also be used to pirate DVDs). What is worse, the interoperability clause was almost forgotten, with the access control rules overriding any exception – this can lead to legally sanctioned control of data formats.

While it is not the "worst law in the world" (other countries are considering or already have stricter laws), the DMCA is only a stepping stone toward more restrictive laws and more restrictive technologies (CBDTPA, TCPA, Palladium). What the research community can do is to act through established channels to influence the lawmakers and make its case heard: support EFF, write your congressional representatives, participate in ACM and IEEE policy-making. There is also an upcoming conference on law and policy of digital rights management at Berkeley, Feb. 27 – Mar. 1, 2003. The Q&A session focused on two topics: how did the content industry manage to get the DMCA enacted? By using a catchy slogan – "piracy must stop" – and lots of lobbying $$$. The second question was what can the computer industry and academia do? Rally behind a strong clear theme and lobby policy makers.

Special Evening Panel on Palladium

Lucky Green, Cypherpunks; Peter Biddle, Microsoft; Seth Schoen, EFF

Summarized by Seung Yi

First, Peter Biddle provided a brief overview of Microsoft's approach for the trusted computing project named Palladium. Palladium is an architecture to protect software from other software (even Windows :) and provide a trusted computing platform. Palladium is a security architecture that will be deployed with newer versions of Windows running on machines with tamper-proof hardware components as described in TCPA. Based on this trusted component or Secure Computing Platform (SCP), as Microsoft names it, authenticated booting procedure and

SCP acts as the core of a security architecture that even the machine's owner cannot bypass. By relying on SCP and other trusted software components built on top of SCP, there are certain parts of the operating system that can be trusted by third parties, and with this capability Microsoft claims to be providing trusted computing. More details on Palladium can be found in an article by Seth Schoen at *http://www.activewin.com/articles/2002/pd.shtml*. Also, Microsoft has a Q&A on Palladium available at *http://www.microsoft.com/presspass/features/2002/jul02/07-01palladium.asp*.

Lucky Green was our second speaker. He used his slides to present the concern he had with the proposed TCPA/Palladium architectures. Basically, his points are:

1. TCPA/Palladium is driven by the vendors to make the PC the core of home entertainment by providing a tamper-proof support for digital rights management (DRM), although it is carefully marketed as the solution for trusted computing.

2. TCPA/Palladium can be used to stifle competition that does not have such support. Green gave an example of Windows vs. Linux today. Even though a user can install Linux on a system, there are certain things that can't be done unless the user also installs Windows. By the same logic, it will be still possible to use a TCPA-equipped PC without installing Palladium OS or other similar operating systems, but the user will not be able to access digital music, digital movies, or even her/his own Word file protected by TCPA. Green pointed out a couple of potential abuses of such systems, not surprisingly things not mentioned in the Palladium specification. By invalidating access to Word documents, for example, the vendor can force the users to buy a newer, accessible version of Word. An OS vendor may be able to block certain "undesirable" applications from running on any user's machines. Green's slides are available at the Cypherpunks Web site at *http://www.cypherpunks.to*.

Seth Schoen maintained a somewhat neutral position between Peter Biddle and Lucky Green, pointing out the potential benefits of the proposed architectures and some concerns.

One of the biggest concerns expressed by members of the audience was the possibility of Palladium being used as a DRM platform or, even more alarming, the base platform to implement a 21st-century Big Brother capability. There were also a couple of questions on what part of these proposed architectures is actually new. Most of the concepts proposed in the architectures were already proposed and implemented a couple of decades ago in trusted computing base efforts like KSOS.

For those who wish to learn more about the issue, Ross Anderson provides a nice FAQ on TCPA/Palladium at *http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html*.

Steven Levy wrote an article on the issue in *MSNBC/Newsweek*, which is available at *http://cryptome.org/palladium-sl.htm*.

Panelists also pointed the audience to the discussions on two mailing lists: *cryptography@wasabisystems.com* and *cypherpunks@lne.com*. Archives of these two mailing lists are available at *http://www.mail-archive.com/cryptography@wasabisystems.com/* and *http://www.inet-one.com/cypherpunks/*.

Refereed Papers

OS Security

Summarized by Prem Uppuluri

Security in Plan 9

Russ Cox, MIT; Eric Grosse, Rob Pike, Sean Quinlan, Bell Labs; Dave Presotto, Avaya Labs

This won the Best Paper award. The chair of the session noted, interestingly, that the three authors who were at Lucent when the paper was published are still at Lucent.

Russ Cox emphasized that the main contribution of this paper is a simple security architecture built on a small trusted code base that is easy to verify, understand, and use. The security architecture was developed for the Plan 9 operating system of Lucent Bell Labs.

The authors believe that the main security concern in a system is not the protocols or the algorithms. Instead, buggy servers, confusing software, and poor configurations are usually responsible. Hence, the emphasis of the paper is on the design of a simple security architecture, rather than the algorithms and protocols used, though they have been described for concreteness.

The main component of their architecture is an agent called factotum (derived from the proverbial servant who has the power to act on his master's behalf and has all the keys to the master's possessions). Factotum is built on the same idea as an SSH agent – each user has a factotum process that is responsible for the user's keys. A factotum effectively takes over responsibilities such as authentication and security interactions with other processes. It thus "frees" other software from dealing with these issues. Cryptographic code is no longer compiled with programs but is handled by the factotum, thus allowing for easy updates to crypto software.

An important security consideration is the storage of the secure keys. Factotum stores the keys in the volatile memory, and so the keys need to be backed up. Storing the key encrypted on a shared file system is possible as long as the keys are not the authentication keys. Encrypting the keys with a user password is also not a good solution. since an attacker can use a dictionary attack to break the key. Hence, the authors describe secstore, which is a file server for encrypted data. secstore is based on an encrypted key exchange called PAK.

The paper also describes other security issues, such as protecting factotum from debuggers.

Despite its advantages, there were a few problems. A person from Mitre Corporation asked whether choosing a poor password made factotum susceptible to a dictionary attack. The speaker acknowledged that it did. Another issue, raised by Whitfield Diffie from Sun Microsystems, was whether the architecture could be easily added to UNIX. The authors conceded that it is difficult to add to the existing operating systems but presented an argument that the ideas behind the architecture described can be used in other OSes.

Linux Security Modules: General Security Support for the Linux Kernel

Chris Wright and Crispin Cowan, WireX; Stephen Smalley, NAI Labs; James Morris, Intercode; Greg Kroah-Hartman, IBM Linux Technology Center

LSMs were designed to compensate for the poor security provided by the Linux kernel, which is the same as the classical UNIX security model, in which root is all-powerful. The main goal of the project is to create a security module API that has low overhead (acceptable to Linus, whom Chris Wright called the "dictator"), is minimally invasive, and satisfies the disparate needs of many security projects.

LSM started in April 2001 and involves over 550 people. It basically provides a framework to implement access control models as pluggable kernel modules.

The main design issues that were considered in the design of LSM included: (1) interposing at a level deeper than system-call level, (2) providing a thin mediation layer called hooks that is agnostic with respect to the security model, (3) making LSM restrictive by allowing a module to either allow or deny an access, and (4) allowing module stacking.

In justifying these design decisions, the authors pointed out that system calls, while a natural choice for inter-positioning, are inefficient and may lead to race problems. Hence, they decided to go deeper into the kernel. In particular, LSM provides an interface that allows modules to interact with internal kernel objects. LSM allows a subject to perform a kernel operation on an internal object by placing hooks in the kernel code just ahead of the access to a resource through the system call. LSM is restrictive in its hooks in that a security module intercepting the hooks can either allow the access or deny it. In order to keep the design simple and minimally invasive, the LSM project is limited to supporting core access control functions required by the current security projects. Sometimes security policies need to be composed. The design of LSM forces the decision on how to compose policies on the modules.

The rest of the paper describes the implementation of LSMs. Finally, the speaker concluded that LSM is efficient, producing about 0–2% overhead in micro-benchmarks and 0–0.3% in macro-benchmarks. Currently, LSM is being merged into Kernel 2.5 and the interface is being refined as pieces are submitted to Torvalds. The work is available at *http://lsm.immunix.org*.

There were questions in the audience as to whether any sanity checks were performed for the modules. The speaker said that code reviews and verification of modules were being done by others.

Using CQUAL for Static Analysis of Authorization Hook Placement

Xiaolan Zhang, Antony Edwards, Trent Jaeger, IBM T.J. Watson Research Center

Xiaolan Zhang discussed the use of a static analysis tool, CQUAL, in verifying LSM authorization hook placement. This work revealed potential vulnerabilities in LSM.

Xiaolan first gave a description of a vulnerability in the security hook security_ops->file_ops->llseek(file) as a convincing reason for the need to verify.

She then described the aim of the work, which was to verify the following two problems: complete mediation and complete authorization. For the former, verification involves checking that whenever a user tries to control a resource, some LSM authorization hook mediates. The latter involves verifying that the set of requirements necessary for prior mediation in the authorization process are met in all the paths to the operation that seeks to control the object.

In case of complete mediation, the authors label the resource to be accessed as a controlled object and the operation accessing the resource as a controlled operation. In order to verify that an LSM authorization hook is executed on a controlled object, before it is used they first identify the controlled objects as, for example, files, inodes, superblocks, tasks, or modules. They then use static analysis to associate the authorized object with those used in the controlled operation. In the next step, they identify all possible paths to the controlled operation. They use typical C semantics. All inter-procedural paths are defined by call graphs, and among these paths they identify those that are needed for analysis.

The authors use CQUAL, a type-based static analysis tool that helps find bugs in C programs. As a first step, the authors annotate the data structures in the program with one of two types: unchecked and checked. In particular, all the controlled objects are initialized to the type unchecked, while all function pointers used in a controlled operation are marked as checked. Authorizations

upgrade the object's type to checked. Since the source code is large, annotation by hand was not feasible. Hence the authors extend GCC and use a set of Perl scripts to annotate the code automatically. Type errors indicate possible vulnerabilities.

Using the above techniques they were able to find a couple of exploitable CQUAL type errors. They also had a large number of false positives.

Asked whether there could be other vulnerabilities that may have been missed, the speaker replied that they had some confidence in the result since the approach was generic and wasn't designed to find any one particular error. Another question was on whether the flow insensitivity of CQUAL was a deterrent. The speaker replied that flow insensitivity only increases false positives and does not result in false negatives. The last question was how the work handled function pointers. This was done by manually annotating function pointers in headers. CQUAL can detect function pointers that have been assigned to some variables.

INTRUSION DETECTION/ PROTECTION

Summarized by Haining Wang

Using Text Categorization Techniques for Intrusion Detection

Yihua Liao and V. Rao Vemuri, University of California, Davis

Yihua Liao presented a new approach to modeling program behavior in intrusion detection by using text categorization techniques; this approach eliminates the need to build program behavior databases or learn individual program profiles.

In his talk, he briefly described text categorization, in which text documents are grouped into predefined categories based on their content, and its usage in information retrieval. The vector space model is used to transform documents into vectors. A word-by-document matrix A is used for a collection of documents, where each entry represents the occurrence of a word in a document and can be computed in several different ways – weighting, frequency (f) weighting, and term frequency–inverse document frequency (tf-idf) weighting. They used as a machine-learning method the k-Nearest Neighbor (kNN) classifier, which calculates the similarity between an unknown document and training samples and looks at the class labels of k-nearest neighbors to predict the class of the unknown document.

To profile a program behavior in a much more general and efficient way, the authors treated each system call as a "word" and the set of system calls generated by the process as the "document." Each process is converted to a vector, and the intrusion detection becomes text categorization. Based on the kNN classifier, the program behavior is classified into different categories, which determines normal or intrusive. The advantages include limited system-call vocabulary so that no dimension reduction techniques are needed; use of simple binary categorization; and, as mentioned above, no individual program profiles to learn.

The experiments for testing the kNN classifier were conducted over a 1998 DARPA BSM data set, which provided a large sample of network-based attacks embedded in normal background traffic. The performance of kNN classifier with the tf-idf weighting technique was measured by the Receiver Operating Characteristic (ROC) curve that plots intrusion detection accuracy against false positive probability. The results show that the k=10 is a better choice than other values for achieving a faster detection rate. Also, they compared the tf-idf with f weighting techniques. Although f weighting achieved a higher initial detection rate, tf-idf weighting reached the 100% detection rate much faster. To detect attacks more effectively, the kNN anomaly detection can be easily integrated with signature verification.

Detecting Manipulated Remote Call Streams

Jonathon T. Giffin, Somesh Jha, Barton P. Miller, University of Wisconsin, Madison

Jon Giffin's talk covered how to detect destructive system calls issued by remote execution systems such as Condor and Globus. The detection was based on the pre-execution static analysis of the binary program, in which specifications were automatically generated. A model representing all possible remote call streams that the process could generate was built. As the process executes remotely, the local machine builds optimizations into the model incrementally, ensuring that any call received remains within the model.

The model is a finite-state machine – either a non-deterministic finite-state automaton (NFA) or a push-down automaton (PDA). The construction of the automaton is accomplished in three stages: by (1) deriving the control flow graph (CFG) from each procedure in the binary program; (2) converting the collections of CFGs into a collection of local automata; (3) composing these local automata at points of function calls internal to the application, and then generating the interprocedural automaton that models the application as the whole.

Two metrics determine the usefulness of the model: precision and efficiency. To improve precision, null-call insertion and call-site renaming techniques are employed. To improve efficiency, stack abstractions and null-call insertion are used. During their prototype implementation, they observed that PDA is more precise than NFA because it provides context sensitivity. However, PDA has a state explosion problem – a stack may grow to be unbounded, leading to high overhead. To solve this problem, the

maximum size of the runtime stack is bounded.

Finally, Jon summarized his talk by highlighting the important ideas of the paper: (1) specifications are generated automatically from binary code analysis; (2) a finite-state machine is built that models correct execution; (3) the push-down automaton (PDA) is precise but suffers high overhead; (4) a bounded PDA stack and null calls make the use of a precise PDA model possible.

Type-Assisted Dynamic Buffer Overflow Detection

Kyung-suk Lhee and Steve J. Chapin, Syracuse University

Kyung-suk Lhee gave an introduction to buffer overflow attacks, especially the well-known stack-smashing attack: the return address of a function is overwritten so that the malicious code is injected into the stack, and so the control flow is directed to the malicious code when the function returns. The key idea of the proposed scheme is that a table in the executable file is built at compile time since the size of the buffer can be known, and the sizes of buffers are checked with the table at runtime.

Kyung-suk presented an overview of their implementation, in which they: (1) built the "type table" that holds types (sizes) of automatic and static variables; (2) maintained heap variables in a separate table by intercepting malloc(); and (3) looked up the "type table" to check buffer size using wrapper functions for the vulnerable copy functions in the C library. The prototype was implemented by extending the GNU C compiler on Linux. Each object file was augmented with type information, leaving the source code intact. To delay making the "type table" until runtime, each object file was given a constructor function "ctor" to build the type table. The range checking was done by a function in a shared library.

Their implementation is transparent since source files are unmodified, and programs are compiled normally using the supplied makefile in the source distribution. Type table–appended object files are compatible with native object files. Protected buffers cannot be overflowed or exploited. Moreover, compared with other approaches, this one is harder to bypass and faster than comprehensive range-checking techniques.

The limitations of the scheme include the following: (1) there are two cases where they cannot determine the size of automatic buffer: alloca(), or allocated buffer, and variable-length arrays; (2) the scheme is unable to determine the type of function-scope variables; (3) it is vulnerable to attacks that do not depend on the protected C library functions; and (4) it cannot protect the parameters of the function that defines a nested (function-scope) function. (The fourth point was not mentioned in the paper.)

ACCESS CONTROL

Summarized by Michael Hohmuth

A General and Flexible Access-Control System for the Web

Lujo Bauer, Michael A. Schneider, and Edward W. Felten, Princeton University

Lujo Bauer presented a new access-control system for Web services. He said that there are already many access-control systems that protect an increasing amount of private data, such as photos or medical records. The problem with existing solutions is that many implement only a simple, fixed application-specific policy, and because of that it is hard to express more complex policies or to get these mechanisms to interoperate.

The authors suggest a new, flexible, and general solution that is application- and policy-independent based on proof-carrying authorization (PCA). In this system, clients submit to the Web server all facts that are required to prove that the client is allowed to access a Web page, formulated in higher-order logic. In addition, the client submits a proof of the propositions that are needed before it can access the server. This moves the (generally undecidable) problem of proving the propositions from the server to the client. The server only needs to check the proof (which is decidable), and the client can construct the proof using application-specific, decidable logic.

In their implementation, the authors modified a standard Web server using applets for generating propositions and for checking client-submitted proofs. On the client side, they use an HTTP proxy that hides all server transactions from the standard Web browser. This proxy handles proof challenges from the server by trying to construct proofs for them. If it is missing facts required for constructing the proof, they ask fact servers (which are specialized Web servers). Bauer said the proxy could be integrated into the browser as a plug-in, but they wanted it to be as browser-independent as possible.

Bauer presented performance results for their system. As the performance is bound by the number of transactions between clients, fact servers, and Web servers, the system uses caching and speculative proving to avoid unnecessary transactions. Clients cache protected URLs and facts and try to guess and speculatively prove the server's challenges before the server actually generates them. Servers cache proven propositions and client-generated lemmas. As a result, the performance overhead of the system is promising.

Bauer concluded the talk with the statement that formal tools and methods have a place in the real world.

Jonathan Shapiro (Johns Hopkins University) asked how one would deal with

revocation of facts in the light of caching. Bauer answered that facts can have a timeout by including a reference to the current time.

Another audience member asked whether submitting endless unfinished proofs to the server would be a potential DoS attack on the system. Bauer affirmed but said that a similar attack existed with previous systems, and now that the server does not have to prove access propositions itself, it had, in a sense, "less to do" than previously. Another question was whether access policies have to be stored in the server. Bauer answered that was convenient but not required.

Access and Integrity Control in a Public-Access, High-Assurance Configuration Management System

Jonathan S. Shapiro and John Vanderburgh, Johns Hopkins University

Jonathan Shapiro presented OpenCM, a new configuration management system designed to support high-assurance development in open source projects.

Shapiro started his talk with the question: what is configuration management (CM)? He proposed two different answers that he deemed too limiting (it keeps track of versions of files or collections of files) before he presented his answer: a CM system should keep track of "lattices of DAGs of attributed BLOBs" (i.e., relationships between file-version trees) and bindings from file versions to names in a workspace, together with file metadata.

The authors started developing a new CM system because they needed support for developing an operating system (EROS) that can be certified by the highest of the Common Criteria assurance levels, EAL7 (comparable to the former orange-book level A1). This assurance level requires software development to be traceable, auditable, reproducible, and access-controlled, and it also requires high data integrity. As EROS is developed as an open source project, additional requirements were that the CM system needs to support many contributors, but not all of them should have write access to the main repository. Aside from the fact that no existing CM system supports all of these requirements, Shapiro also mentioned the need for a CM system that "actually worked" and that existing commercial offerings did not support the open source development model very well.

OpenCM is designed to protect against such threats as modifications (of the source code repository) by unauthorized users, modifications from compromised clients, compromises through the underlying operating system, impersonation of a source repository, and falsification of repository content. OpenCM reaches these goals by establishing a chain of integrity and authorization for each change request, and by using transactions to commit changes to the repository.

Shapiro explained that the key idea for meeting the integrity requirement was to realize that most of the objects a CM system stores (such as file contents of a particular revision) never change (because of its archival character); he referred to these objects as frozen objects. Therefore, the cryptographic hash of a frozen object's contents also never changes and can be used as a name to reference the frozen object. Whenever such a name is de-referenced, the contents of the object can immediately be checked for integrity. The integrity of mutable objects is ensured by cryptographic signatures.

A transacted change to the repository is reduced to the addition of new data as frozen objects and the atomic revision of a single mutable object, the branch to which the change is committed. Access to mutables is controlled using access-control lists.

Shapiro then identified a number of possible weaknesses of OpenCM: content compromise using a stolen repository-server key, history disclosure using exposed hash names of previous versions, and separate evolution of database and client-server protocol schemas. He proposed solutions or recovery possibilities for each of these problems.

Shapiro concluded his talk with a demo of OpenCM running on his laptop.

Petros Maniatis (Stanford) asked whether more than one server can be authoritative for a given repository. Shapiro answered that OpenCM does not support this mode of operation, as distributed updates to a single repository would be unfeasibly complex. However, changes can be committed to a (nonauthoritative) replicated repository and merged into the authoritative repository later.

An audience member asked whether changes should be signed. Shapiro replied that they shouldn't, but that the subject would be too complex to discuss as part of his talk. He suggested taking the issue offline.

Richard Wash (CITI Michigan) asked what would happen if two nonidentical frozen objects happened to have the same content hash. Shapiro said that a hash collision would be noticed but could not be recovered from. He said that such a collision would be extremely unlikely, though.

HACKS/ATTACKS

Summarized by George M. Jones

Deanonymizing Users of the SafeWeb Anonymizing Service

David Martin, Boston University; Andrew Schulman, Software Litigation Consultant

This paper presented an analysis of the SafeWeb anonymous Web browsing service. The anonymizing service was halted in November 2001.

The goal of the service was "to help oppressed international users" who wanted to view Web content that their country/organization/ISP/etc. prohibited. It also had appeal to corporate and home users. Requirements appear to have been speed, ease-of-use, unmodified content, and no client-side modifications or settings.

The main method employed was to disguise the connection so that all browsing was proxied through HTTPS connections to SafeWeb.com. Both URL and contents were encrypted. Possible attacks were presented. Some involved sending content (JavaScript) that induced the browser to go directly to the source Web site. SafeWeb's rewrites were not perfect.

Some conclusions: SafeWeb took the wrong default stance by blocking known bad (e.g., java-script) elements and allowing all else. Its use openly defied local policies/laws.

VeriSign CZAG: Privacy Leak in X.509 Certificates

Scott G. Renfro, Yahoo!

Scott Renfro examined VeriSign's CZAG extension as an example of embedding sensitive information into X.509 certificates. He then considered the general case of sharing certified information with multiple parties.

In 1997 VeriSign asked end users to (optionally) include country, zip, age, and gender (CZAG) information when registering for class one certificates. Users assumed that this information would be kept private and only shared with trusted parties. But there were problems. It was protected only by weak encryption (XOR), there was no revocation enforcement, it was available in a public LDAP directory, indexed by email, and easy to crawl.

Next, Renfro listed goals, design constraints, and possible alternate implementations for allowing certificate authorities to share sensitive information without unnecessary disclosure. The "most obvious" solution is to keep sensitive information in a centralized database which responsible parties can query with their own credentials. Another option would be to use better key management and stronger encryption for sensitive information. A third set of options involves various methods of putting control of sensitive information in the user's hands, departing completely from the X.509 certificate approach.

How to Own the Internet in Your Spare Time

Stuart Staniford, Silicon Defense; Vern Paxson, ICSI Center for Internet Research; Nicholas Weaver, University of California at Berkeley

Paxson gave very plausible visions of Internet attacks to come based on recent experiences with Code Red and Nimda and made the case for the creation of a "cyber Center for Disease Control (CDC)."

"What could you do if you owned a million hosts?" Launch DDoS attacks, wipe out disks, rummage through email and credit card databases, crack passwords, send "trusted" messages, stage cyberwarfare between nations or acts of outright terrorism.

"How do you own a million hosts?" Short answer: worms. The Morris Worm owned 10% of the Internet. Code Red (2001) peaked at an infection rate of 1900 infections/minute. Monitoring of two class B networks showed 300,000 infected hosts. The larger the vulnerable population [read: IIS install base], the faster it spreads. Nimda spread itself several ways, including by looking for back doors installed by Code Red. "These viruses form an ecosystem."

"We couldn't resist designing better worms," Paxson said and then outlined several methods future worms could use to spread quickly by intelligently splitting up scans of the IP address space. Peer-to-peer networks and "contagion" worms present other fruitful methods of spreading malicious code. "If you have the entire hit list [vulnerable hosts] and infected a few and divide up the list, then it is possible to infect 1M-10M hosts in seconds. These time-scales are way beyond human response."

So what's the answer? A "cyber CDC" that would identify outbreaks, coordinate response, do rapid analysis, help resist infection, watch traffic, set strategic direction, and foster research. "This may sound hard, but what's the alternative?"

Q: What are you proposing beyond CERT/FIRST?

A: Automated response, instant analysis.

Q: How seriously do you take the threat of embedding viruses in pictures and other file types?

A: Nonexecutable files are probably not a significant worry.

Q: Do we have a need for more centralized analysis of worms?

A: This is very ripe for research. Open community analysis has been very helpful . . . but we still don't know what Nimda does.

Q: Can you comment on the use of worms to patch security holes?

A: That seems like a non-starter. There is a very large liability issue.

SANDBOXING

Summarized by Prem Uppuluri

Setuid Demystified

Hao Chen, David Wagner, University of California at Berkeley; Drew Dean, SRI International

Hao Chen addressed a critical problem with the use of UID-changing calls, asserting that setuid and seteuid suffer from many flaws. They are poorly

designed, lack proper documentation, are widely misunderstood and, hence, misused by programmers. As an example he pointed out that a system-call setuid(0) (setuid to root) shows different behavior in Linux and BSD. In Linux it sets only the UID to 0, whereas in FreeBSD it may set all the three UIDs – SUID, UID, and EUID – to 0. Another problem he illustrated was that sometimes the UID-changing calls may not actually succeed. For instance, the system-call seteuid(geteuid()) seems like an identity function and so is expected to succeed, but may not necessarily do so.

To address such problems, the authors studied the kernel sources for these calls and then compared the precise semantics of the calls across Linux, Solaris, and FreeBSD. They did this by constructing a formal model of user IDs as a finite-state automaton (FSA). This FSA helped them find some of the pitfalls of the UID-changing calls and also helped them identify the semantic differences of these calls across the three operating systems.

The authors describe the model-extraction algorithm which constructs an FSA. The states of the FSA contain the values of UID, SUID, and EUID. A transition is labeled with one of the UID-changing calls. From each state there is one transition labeled with each UID-changing call. Each transition leads to a state which contains the values of the three UIDs after the execution of the UID-changing call associated with the transition.

Using the finite automaton they built, they were able to verify a number of inconsistencies: a man page of RH Linux 7.2 fails to mention setuid capability and a man page of setreuid in FreeBSD 4.4 mentions incorrectly that unprivileged users may change real UID to effective UID. They were also able to identify that the implementations of the calls across the operating systems were different.

At the end of the paper, they provide guidelines to the proper use of these system calls. For instance, they suggest that setesuid be used where available as it has very explicit and clear semantics and sets the three user IDs independently. They also suggested that users check for errors in the return code of system calls. In particular, a good technique to confidently drop privileges is to first drop the privilege permanently, try to regain the privilege, and ensure that the program cannot regain the privileges. Further information on their work is at *http://www.cs.berkeley.edu/~hchen/research/setuid/*.

Secure Execution via Program Shepherding

Vladimir Kiriansky, Derek Bruening, Saman Amarasinghe, MIT

Saman Amarasinghe argued that it is not possible to attain zero bugs in code. Thus it is necessary to look at other techniques to prevent the bugs from being exploited. The key point on which they base their work is that one who owns the program counter controls the code. An attacker who is prevented from hijacking the program counter may overwrite data but cannot control the code. Based on this observation, they described their approach, which they call program shepherding.

In program shepherding, all control-flow transfers during a program execution are monitored, and security policies are defined to determine allowable transfers. Program shepherding can be done in two main ways. One way is to instrument application and library code prior to execution and to add security checks around every branch instantiation. They argue, however, that this approach is not viable or applicable. The approach they took was to use an interpreter.

The naïve approach to interpreting, however, is very slow. Hence they used a dynamic optimizer (DynamoRIO base system built in association with HP labs and MIT) in order to improve the performance of the interpreter. In addition they ensured that non-control flow instructions did not get interpreted. They further reduced overhead using indirect branch lookups.

To measure the effectiveness of their approach, they used a set of vulnerable applications: stunnel, groff, ssh, and sudo. They were able to foil all exploits, with no false positives. Their performance numbers were also very good, with the overhead around 8% due to their interpreter.

Someone asked how this approach differed from fault isolation techniques; Saman replied that in this approach the isolation is at a lower level of granularity.

A Flexible Containment Mechanism for Executing Untrusted Code

David S. Peterson, Matt Bishop, and Raju Pandey, University of California at Davis

David Peterson described a variety of sandboxing techniques and explained the design of their framework, which draws from these different techniques.

Peterson started by describing the different design alternatives available for sandbox creations. In particular he addressed:

1. Representation and organization of privileges in the sandbox. They first identified resources that needed to be protected, including device components, file systems, network components, and signal components. When a sandbox is created, one or more of the components are attached to it. Initially only the sandbox creator is given privileges for these components, but privileges to other processes in the sandbox can be added.

2. Location of enforcement mechanisms. The authors described the various choices to insert the enforcement mechanisms: runtime environment, sandboxed program, user space, and OS

kernel. They chose the OS kernel, as it allowed them to use the system-call API.

3. Passive or active monitoring. Passive monitoring involves changing the system-call execution such that any enforcement mechanisms are checked before the system call is allowed to proceed. This involves modification of the system call. Active monitoring requires that an external process monitors the program. Both these techniques have advantages: active monitoring is flexible, and passive monitoring introduces low overhead. The authors decided to use a mechanism that allows for either or both of the monitoring techniques.

4. Whether to group sandboxes globally or locally.

5. Whether the access control mechanisms must be mandatory or discretionary. Their design provides both options.

6. How to guard access to sandbox-related objects.

Peterson discussed many other options and described the design of their sandbox. The overhead introduced by their system varied from 0.3 to 4.0%.

An audience member wondered whether they were considering making their system into an LSM module. The reply was an affirmative.

WEB SECURITY

Summarized by Haining Wang

SSLACC: A Clustered SSL Accelerator

Eric Rescorla, RTFM; Adam Cain, Nokia; Brian Korver, Xythos Software

SSL is much more CPU intensive than ordinary TCP communication, because of the cryptographic computation, especially the RSA operation in the SSL handshake. To offload the cryptographic overhead, an accelerating proxy is introduced. However, the accelerator becomes a single point of failure, and multiple accelerators are only a partial solution since any connection on a failed accelerator is lost. The real problem with SSL is that the user does not know whether the transaction is complete and so is unwilling to re-submit the transaction.

Eric presented a better approach, a clustered SSL accelerator, in which all nodes in the cluster share the connection state. When any node fails, the remaining nodes are able to take over all connections that terminated on that node with no interruption in service. Failures are invisible to the end user; this process is called active session failover. The design principles of SSLACC are embodied in the three laws of clustering: (1) "all nodes must generate the same data," and all nodes behave as one virtual device; (2) "cluster then commit," which requires tight control of the TCP stack; and (3) it is safe to transmit unclustered data if you can reproduce it.

Note that they do not cluster data but use a clustered TCP relay. Data is automatically buffered by the client. Only full records can be processed at the server, however, and sometimes records are bigger than the TCP window size (especially during slow-start). The proposed solution is to ACK a partial record: cluster the record data read so far and ACK the partial read. To keep cluster updates as small as possible, only a minimal amount of state is transmitted so that the other nodes can reproduce the original state on failover. In conclusion, the most desirable properties in a clustered accelerator are scalability, high availability, and the ability to run on cost-effective hardware.

Infranet: Circumventing Web Censorship and Surveillance

Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger, MIT

This paper won the Best Student Paper award. Nick Feamster presented Infranet, a way to circumvent Web censorship and surveillance that consists of requesters and responders communicating over a covert tunnel. The key idea is that the Web browser requests the censored content via Infranet requester as a local proxy, which in turn sends a message to an Infranet responder. The responder retrieves this content from the appropriate origin Web server and returns it to the requester, then the requester forwards the received content to the browser. The covert communication tunnel securely hides the exchange of censored content in normal, innocuous Web transactions.

Then he described what kind of censors people might want to get around, which include restrictive government, corporate firewall, etc. Basically, there are two classes of attacks mounted by the censor: discover attack, where the censor monitors the Web traffic for unusual-looking access attempts and traffic; and disruptive attack, which blocks communication between endpoints by preventing access to certain Web sites or attempting to block access to circumvention software. Related systems – e.g., Triangle Boy, Peekabooty – and their vulnerabilities were mentioned.

The design goals of Infranet include: (1) deniability for clients – the censor cannot confirm that any client is intentionally downloading information via Infranet; (2) statistical deniability for clients – the browsing patterns are indistinguishable from innocent clients; (3) covertness for servers – the censor cannot discover a server that is serving censored content and so cannot easily block such a server; (4) communication robustness – the Infranet channel should be robust in the presence of censorship activities designed to disrupt request/transfer of censored content; and (5) reasonable performance.

In the downstream communication, censored data is embedded in images

and recovered later by shared secret. However, steganography is not ideal, because it cannot reuse a cover image. Web cams, where images are constantly changing, would be a better choice. In the upstream communication (i.e., requesting), the requester divides the hidden message into multiple fragments, each of which is translated to a visible HTTP request by a modulation function. The mapping function was a design trade-off between covertness and bandwidth consumption. The reasonable performance is achieved by taking advantage of the asymmetric bandwidth requirements of Web transactions, which require significantly less upstream bandwidth than downstream bandwidth.

Trusted Paths for Browsers

Zishuang (Eileen) Ye, Sean Smith, Dartmouth College

Eileen Ye first pointed out that the human user is the true client, not the machine; however, the communication between the Web browser and the user is a neglected component of the server-client channel. Simply ensuring that the machine draws the correct conclusion does not suffice if the adversary can craft material that nevertheless fools the human. According to their definition, Web spoofing is malicious action causing the reality of the browsing session to be significantly different from the mental model a reasonably sophisticated user has of that session.

They tried to reproduce Princeton's Web spoofing experimental work done in 1996, but they did not succeed, due to the advances in Web technology and browsers' user interface. So they conducted their own experiments to demonstrate the weak link between the human user and the Web browser. To foil Web spoofing, a trusted path was created between the browser and its human user. Through this trusted path, the browser can communicate relevant trust signals that the human can easily distinguish from the adversary's attempts at spoof and illusion.

Besides clearly communicating with the security-related information, the attributes of the trusted path should include: inclusiveness (working on all interfaces), effectiveness (expressing the security information in a way the user can easily understand), minimal intrusiveness, and minimal user activity. To meet these requirements, a colored boundary approach was taken, known as synchronized random dynamic (SRD) boundaries. In an SRD environment, all windows have colored boundaries. A blue boundary window (containing server materials) indicates an untrusted window, while an orange boundary window (containing browser materials) indicates a trusted window. The window boundary has two styles: inset and outset. At random intervals, the browser would change the styles on all its windows. The random pattern of the boundary style cannot be predicated by the server, so the server cannot forge a window image to impersonate the real window.

Mozilla was chosen as the base browser for implementing SRD. There are three steps to implement SRD: (1) add special boundaries to all browser windows; (2) make the boundaries change dynamically; and (3) make all windows change synchronously. To resolve the address-blocking problem (i.e., an SSL warning window blocking other windows), a reference window running in a separate process was introduced. The reference window changes its image by random number to indicate the boundary style. In usability studies, three test scenarios were included: (1) without reference window, (2) a full SRD approach, and (3) a CMW-style approach. The conclusions drawn from a user study were: it works! See the paper for additional suggestions.

Generating Keys and Timestamps

Summarized by Michael Hohmuth

Toward Speech-Generated Cryptographic Keys on Resource-Constrained Devices

Fabian Monrose, Qi Li, Daniel P. Lopresti, and Chilin Shih, Bell Labs, Lucent Technologies; Michael Reiter, Carnegie Mellon University

Michael Reiter presented this talk on what he said was fairly speculative research: the extraction of a key usable for cryptographic purposes from a biometric such as voice. The main criteria for a usable system would be that it works reliably and efficiently even with constrained resources such as cell phones, PDAs, and other wearable devices and that key extraction should be difficult even if an attacker gets access to the samples of the biometric.

In this research, the authors concentrated only on voice, since that is the natural interface for many wearables. Also, voice is a dynamic biometric in that the user can change a "passphrase" by speaking a different phrase or changing intonation, and thus can have many different keys. Reiter stated clearly that he indeed meant voice, not the phrase recognized and recovered from voice; the latter would have many fewer features and would mean a loss of information and thus key length when compared to pure voice.

Reiter first presented an overview of their system. It works by taking a voice sample, generating a list of small segments through digital signal processing,

extracting from the segments a vector of binary features (which Reiter called feature descriptors), and, using each feature, selecting a key element from a two-columned key table. As not each repetition of the passphrase yields exactly the same feature descriptor, the algorithm also needs to reconstruct the correct feature descriptor by searching within a given Hamming distance of the extracted feature descriptor (key reconstruction).

Reiter said that he and his colleagues have presented parts of this system earlier in other publications (IEEE S&P 2001, ACM CCS 1999); in this talk, he would focus on the implementation, on the signal-processing part, and on empirical analysis of the strength of generated keys.

The authors first implemented their system on the Yopi, a Linux PDA powered by a 206MHz StrongARM CPU. This implementation suffered from a low-quality microphone built into the device and a poor OSS sound-driver implementation. In a second implementation, the authors switched to the iPAQ 3600, also equipped with a 206MHz StrongARM.

As an illustration of the harsh realities developers face when using resource-constrained devices such as these, Reiter explained that silence elimination was an important step in their signal-processing step and showed waveforms of recorded "silence" generated by these devices. Instead of silence, the Yopi recorded static. The iPAQ's waveform was distorted by the device's automatic gain control.

Using these devices, key reconstruction currently works practically with a Hamming distance of up to five features (on future systems, the authors expect to be able to support six features). Based on typical Hamming distances when comparing the feature descriptor originally recorded and a capture of the passphrase spoken by the real speaker, this limits the number of distinguishing features that can be supported on these platforms to about 30. Using the best-known attack, an adversary that can only randomly guess features needs 2^40 multiplications to recover the key.

The authors also looked at other attacks on the signal-processing part that they deemed more promising than random guessing: another person uttering the same passphrase, and recovery of the original passphrase using the original speaker's voice by way of sophisticated text-to-speech synthesis and diphone cut-and-paste from a huge database of phrases spoken by the original speaker. Reiter mentioned that an attacker does not need a database as large as theirs; 20 minutes of good-quality recordings of the speaker would contain enough phonemes to synthesize 50 percent of the passwords they tried.

Interestingly, these impersonation attacks did not yield better results than random guessing. Reiter said he and his team had expected that these attacks would break their system, and they were surprised that they did not. It is unclear why these attacks do not work. Reiter speculated that he and his coauthors did not carry out the attacks correctly, or that speech synthesis is too immature, but he said that this kind of attack must be expected to become more powerful in the future.

In conclusion, Reiter said that the feasibility of using voice for generating strong keys is still unproven, but their results indicate that the approach is promising and can be implemented.

Paul van Oorschot (Cloakware) asked about the security that can be expected if an attacker obtains a recording of the speaker speaking the passphrase. Reiter replied that the authors would make no claims about that case.

Neil Daswani (Stanford University) asked whether their cut-and-paste attacks included cases in which whole subphrases of the passphrase were concatenated. Reiter said that this type of attack was included in the study.

Another audience member asked whether they tried speech synthesis using AT&T's Natural Voices product, released about one year ago, and how it compares to other speech-synthesis products. Reiter said that he does not know of AT&T's product and hence cannot compare it.

Secure History Preservation Through Timeline Entanglement

Petros Maniatis and Mary Baker, Stanford University

Petros Maniatis started out by referencing Jonathan Shapiro's talk earlier in the conference. He said that Shapiro was concerned with preserving history of a collection of files; his work has the same goals, but in a broader context, that of preserving the sequence of a host of events in a large distributed system.

Maniatis said that in this work, history is defined to be the temporal ordering of system events such as storing a file on a disk or signing a document. Such events can occur in unrelated, distributed components. However, there are circumstances in which the order of two such events is important even if they did not occur in the same system, for instance when referencing prior art in patent disputes.

The speaker went on by giving a more elaborate motivating example in which an investor, Marti, ordered a sell of shares of some company. The next day, something bad happens to the company. Marti's broker sells the shares a day later, just before the stock price plummets prior to the bad news becoming public. Later, the SEC accuses Marti of insider trading, and now Marti would like to prove that he ordered the sell of shares before the bad event occurred. Maniatis insisted that this example was purely fictitious, which amused those audience members who had followed that week's US national news revelations about MCI/Worldcom's creative bookkeeping.

The authors set out to build a system that is designed to preserve the sequence of events "long after the 'historians' leave," under the assumption that no party trusts another. In their approach, each component maintains a local history and a local view of the global history. Components safeguard the integrity of the portions of history they know about and trust only themselves or information that can be proved. Other

requirements on the system were efficiency, scalability, survivability, and aggressive decentralization. To address these requirements, the authors developed a method for "timeweaving," interconnecting local histories with each other so that a global history can be reconstructed.

Maniatis explained that each component's history consisted of a hash chain of commitments of local events. The elements of the chain are called time steps; they contain the current local time, a description of the event, and an authenticator. The authenticator links the time step to the previous one in the timeline using a one-way hash function. Then, precedence can be proven by giving enough information for walking a thus-established hash chain. To avoid having to disclose each and every event between two events of interest, the chain includes special events that reference each other and that form a skip list for jumping over a number of other events.

Timeline entanglement, or timeweaving, works as follows: components regularly publish timeline samples for other components to witness, and witnesses commit published samples in their own timeline. Then witnesses send the originating component an entanglement receipt, which includes a precedency proof stating that all events in the publisher's past occurred before all events in the witnesses' future.

Maniatis then covered implementation aspects. Here, the challenge was to find a balance between storage overhead needed for storing authenticated hash chains and the number of disk accesses and computation steps needed to compute precedence proofs. The authors use a new data structure, RBB-Trees, which bounds the maximal number of disk accesses needed to compute an authenticator to three. Their performance study shows that in a network of 1200 1GHz PCs that generate events every second and in which each pair of hosts entangles every 10 minutes, each PC uses about 8% of its resources.

Matt Blaze (AT&T Labs) asked whether a possible attack on the proposed system would be to add many histories, making entanglement between all of them impractical. Maniatis affirmed, saying that if there was not enough framework to connect two events, no precedence could be proved.

WORK-IN-PROGRESS REPORTS, AKA QUESTIONS FROM PETER HONEYMAN

Summarized by George M. Jones

Session Chair: Kevin Fu

At the work-in-progress (WiPs) session, presenters are given five minutes to talk about current work and take questions. Due to the presentation format and space limits, these summaries are guaranteed to contain omissions, gross inaccuracies, and misrepresentations of presentations on some fine work. You are encouraged to contact the presenters for more complete, less sketchy information. Also see *http://www.usenix.org/events/sec02/wips.html* for the authors' own abstracts.

Preventing Privilege Escalation

Niels Provos, CITI, University of Michigan

Provos presented the idea of separating applications into two parts, privileged and unprivileged, citing the example implementation in OpenSSH, which he claimed had prevented the "gobbles" attacks from taking over CITI.

Memory Accounting Within a Multi-tasking Language System

Dave Price, Rice University

Price talked about a solution to the problem of memory accounting in an environment (Java) where all tasks share a single heap. The solution proposed was to do accounting during garbage collection. This is done by starting at the root of each task and walking the reachable memory tree, charging the first task for shared memory.

Semantics-Aware Transformation and Anonymizing of Network Traces

Ruoming Pang (with Vern Paxson), Princeton University and ICSI Center for Internet Research

This talk presented work on a way to scrub network traces of private information using the BRO IDS. Stream reassembly is done (see work presented by Paxson et al. last year), and users are given the ability to write AWK-like scripts that can tag/scrub their data before it is entered into the trace.

Clilets: Web Applications with Private Client-Side Storage

Robert Fischer, Harvard University

Fischer presented a new system called "clilets" to implement privacy on the Web. The user sends a request to the Web server, the Web server sends a "clilet" to a multi-domain sandbox, the sandbox sends HTML to HTML verifier, HTML verifier sends HTML to Web server, which sends it to client. The server and clilet work together to create the HTML. Peter Honeyman asked, "This sounds like Java VM – what's new?"

Checking Linux Kernel User-Space Pointer Handling with CQUAL

Robert Johnson, and Sailesh Krishnamurthy (with John Kodumal), University of California at Berkeley

Johnson talked about a system called CQUAL that solves the problem of verifying correct uses of user and kernel pointers in the Linux kernel. The C type system does not support this, but CQUAL does. Using this system, an actual bug was found and fixed in the Linux 2.4.19 kernel.

Segmented Deterministic Packet Marking

John-Paul Fryckman, University of California at San Diego

Fryckman proposed a solution for tracing attacks across the Internet. It involves adding "back-pointers" to packets in the IP headers. The first (edge) AS

and every subsequent AS adds its own AS number to the packet. It was claimed that with at most 17 AS numbers, the entire Internet could be covered.

Turing: A Fast Software Stream Cipher

Greg Rose, Qualcomm Australia

Rose presented initial work on a new fast, simple stream cipher called Turing, designed for use in cheap, slow, small CPUs with little memory. It uses keyed non-linear transformation and was inspired by work on "tc24." The net effect: an Athlon can do 3 cycles/byte. "If it works and is secure, it will be the fastest stream cipher in software."

Active Mapping: Resisting NIDS Evasion Without Altering Traffic

Umesh Shankar, University of California at Berkeley

Ways of avoiding IDSes have been known for some time (Ptacek, Newsham, 1998). These problems stem from uncertainty about what packets reach end systems and how they are interpreted. Most of these problem can be overcome by normalizing the traffic and interpreting the TCP stream as the target system would. To do this, the authors built a database of the systems and types of systems on their local net and performed IDS on normalized data as the end system would see it.

Making Software Resistant to DoS Through Defensive Programming

Xiaohu (Tiger) Qie (with Ruoming Pang and Larry Peterson), Princeton University

This talk presented the case for building robust network infrastructure (routers, systems) by applying improved programming techniques and tools. They built a C toolkit, allowing programmers to specify general resource usage policies. It does some flow analysis, performs consistency checks, and uses sensors/actuators. It was used in real software (Linux networking code). Results were mixed.

VFiasco – Toward a Fully Verified Operating-System Kernel

Michael Hohmuth, TU Dresden

Hohmuth and associates believe that "formal methods can be worthwhile," and they deny the conventional wisdom that "OS verification is an intractable problem." With that starting point, he presented their work on Fiasco, a microkernel OS written in a C++ subset and their results in proving one class. To the question of how long it would take to prove the whole OS, Hohmuth answered, Three to four years.

Wormhole Detection in Ad Hoc Networks

Yih-Chun Hu, CMU

Your humble summary writer admits to note-taking failure for this talk and kindly asks that you visit the author's Web site:
http://monarch.cs.rice.edu/papers.html

A Snapshot of Global Internet Worm Activity

Dug Song, Arbor Networks

Song presented work on monitoring Internet worm activity by monitoring large chunks of unused Internet address space. The work is unique in that for 1/N SYNs to port 80, they reply with an ACK and then log payloads. Using this method they can track attacks individually and can see DDoS and backscatter traffic. Song also presented data on the rise, continued prevalence, and interactions of Code Red and Nimda.

Off-the-Record Communication

Nikita Borisov, University of California at Berkeley

In online conversations as in the real world, you may want conversations to be private, but you may want repudiation...the ability to deny that you said something. PGP and friends use long-lived keys that provide non-repudiation. This is not good for casual conversation. The author then presents work on a protocol for instant messaging to solve this problem. It involves frequent key renegotiations, symmetric authentication, revealing the MAC key in the clear, and introduction of delays.

Plutus – Enabling Secure Sharing of Persistent Data

Erik Riedel, Seagate Research

Riedel presented file system work done at HP to address the problems of both sharing and protecting data, dealing with key management, and distributing the encryption workload. Their system pushes key management and encryption to the edge, uses untrusted servers that only do verified writes, supports keys for groups of files, not users, and is client centered. It is built on AFS using secure RPC.

A Signature Matching Engine for BRO

Robin Sommer, TU Munich, ICIR

Sommer said that traditional signature matching just compares signatures to net traffic, whereas BRO reuses existing signatures and uses regular expressions. BRO supports bi-directional signatures and uses knowledge about target (this is Apache server; IIS exploit does not matter).

Honeyd: A Virtual Honeypot Daemon

Niels Provos, CITI, University of Michigan

Provos presented his work on "honeyd," which implements a small, low-interaction virtual honeypot. It can simulate arbitrary TCP services, listen on up to 65,000 IPs at one time. It reads the nmap fingerprint database and can respond appropriately to impersonate anything in nmap DB. It can simulate arbitrary virtual routing topologies, lie to traceroute, and simulate packet loss and various services. You can proxy attackers back to themselves.

Peter Honeyman asked, "This is not part of your research. How do you ever expect to get your Ph.D. [from me] working on stuff like this?"