;login:

THE MAGAZINE OF USENIX & SAGE

November 2000 • volume 25 • number 7

THEME ISSUE: SECURITY

edited by Rik Farrow

inside:

CONFERENCE REPORT

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

Conference Reports

This issue's report is on the 9th USENIX-Security Symposium held in Denver, Colorado, August 14-17, 2000.

Thanks to the summarizers:

Mike Brown, Doug Fales, Ove Heigre, Philip S. Holt, Himanshu Khurana, Radostina K. Koleva, Admir Kulin, Xinzhou Qin, Algis Rudys, and David Wragg.

The 9th USENIX Security Symposium
August 14-17, 2000
Denver, Colorado, USA

INVITED TALK

Computer System Security: Is There Really a Threat?

Dave Dittrich, University of Washington

Summarized by Radostina K. Koleva

Dave Dittrich gave a truly intriguing talk on Distributed Denial of Service (DDoS) attacks, which attracted a lot of attention after the February 2000 attacks on several e-commerce cites. Dittrich, who has a great deal of experience in identifying and analyzing distributed attack tools, presented the DDoS attack-tool timeline in detail. He also presented the typical phases of an attack, pointing out why anyone would launch such an attack and what makes it possible to do so. Dittrich ended the talk by showing what can be done to stop attacks.

The talk started with a brief history of DoS showing its development from classic resource-consumption attacks to remote resource consumption. Next were coordinated types of remote attacks and, finally, distributed attack tools. Dittrich presented the characteristics of the identified DDoS tools, including: when they appeared, what type of code was used, what operating systems were targeted, what communication protocols were used, whether encryption protection was used, to what extent control features were developed, and, most important, how specifically the attack was performed. The outlined tools included: fapi, fuck_them, trinoo, TFN, TFN2K, Stacheldraht, Stacheldraht v2.666, shaft, mstream, and omegav3.

The DDoS attack-tool timeline first mentioned the primitive DDoS tools affecting small networks in May 1998. One year after the introduction of the first DDoS tools, CERT began to see and report on widespread intrusions to Solaris systems. August of the same year brought the first indications of large-scale intrusions at the University of Washington and later the attack on the University of Minnesota. In September the content of a stolen account used to cache files was recovered by Dave Dittrich, and soon after he provided CERT and the FBI with the first draft of the trinoo analysis. CERT reevaluated hundreds of Solaris intrusion reports and saw that they fit the attack profile outlined in the trinoo analysis.

In mid-October CERT mailed the invitations for the DSIT workshop, which had been designed to deal with new types of attack tools. The end of October brought the final trinoo and TFN analysis. Shortly after that, when the DSIT workshop was held in Pittsburgh, the participants decided not to panic people and suggested how to resist the new threat. The final report was released in early December (<*http://www.cert.org/reports/dsit_workshop.pdf*>).

Things became frantic with the approach of 2000, and for first time the FBI director and US attorney general were briefed on DDoS tools. At the end of December the analysis of "Stacheldraht" was finished and CERT issued an advisory on DDoS attacks. To everyone's relief, New Year's day passed with no incidents. Early January marked the release of another CERT advisory and the development and distribution of scanning and detecting tools. In the middle of January, an attack on OZ.net occurred without making it to the national press. ISCA.net organized a Birds of a Feather session on DDoS shortly after that. Ironically, a talk by Steve Bellovin on DDoS at a NANOG meeting was being presented at the time when the well-known attack on e-commerce sites began in February. Sometime after that several other attacks were launched abroad (Brazil, New Zealand), but did not receive wide media attention. Till the day of this talk on August 16,

2000, the reports of more attacks kept coming in.

Next the talk provided some insight into the significance of the timeline. It was pointed out that the government issued its first advisory in December, right after the analyses were made publicly available on BugTraq, while other sources of information and analysis had also been available by the beginning of February when the attack on e-commerce sites happened.

The DDoS attacks can be considered to consist of two phases. The first phase, or the initial intrusion, consists of initial root compromise, which can be achieved in variety of ways. Tens or hundreds of thousands of potential targets are first scanned, resulting in a set of high-probability targets. An attack is launched shortly thereafter. The attack involves installing DDoS tools after breaking root, and often some means of concealing traces are employed. The second phase is the actual attack. The attack makes the victim network unresponsive and may lead to router failures.

The proper identification is particularly difficult for a variety of reasons, including the fact that most sites are unprepared to analyze packets, that the attacks may look like hardware failure, that coordination with an upstream provider is necessary, and that it is difficult to identify all agents.

The next question answered in this talk was – why would anyone do it? It was pointed out that these types of attacks are a direct result of IRC channel takeovers and retaliation. Attackers often want to see if they can do it, and sometimes do it just because they can. Dittrich made it clear that such attacks may happen at bad times, one example being bringing down a computer system that is used to supply information and help during surgeries at a hospital.

The next issue addressed was – what allowed all this to happen? The reasons shown were: a target-rich environment, poor understanding of network monitoring, primary focus on service restoration without data gathering, software and OS designed with priority of ease of use over security, speed and complexity of intrusion overwhelming, and poor network and forensic data gathering.

In order to stop DDoS attacks it is necessary to employ ingress and egress filtering, improve intrusion-detection capabilities, audit hosts and networks for DDoS tools, have incident-response teams, enforce policies for securing hosts in the network, be able to receive the cooperation of the upstream provider, and provide insurance for covering service disruption.

Dittrich then presented an evaluation of where the current situation seems to be headed. About 21 million new hosts are added to the Internet each year, while the increase of the number of system administrators is not nearly that drastic. The DDoS tools are evolving, techniques for post-compromise concealment are improving, and efficiency of compromising systems is growing. Law enforcement seeks stronger laws, while software vendors continue to avoid government regulation. Meanwhile the trend is for businesses to use the Internet.

The talk concluded with Dittrich's opinion on what we need in order to deal with DDoS attacks. He suggested that every organization needs a chief hacking officer and that it is necessary to accept that the system administrators are essential for the New Economy. He pointed out the importance of acknowledging that security is a cost of doing business, and that speed should no longer be put before security. It is also important for the software and OS vendors to adopt the same kinds of standards as other mature industries. It should be realized that the Internet, as it is now built, is not a reliable place to do "important" things and needs to be improved. While the user demands for new features and services on the Internet will continue to grow, there should also be a trend of educating the users about how to deal with the insecurities of a hostile Internet.

The presentation and a lot more information about DDoS is available at <*http://staff.washington.edu/dittrich/misc/ddos/*>.

REFEREED PAPERS

SESSION: OS SECURITY

Summarized by Doug Fales

MAPBox: Using Parameterized Behavior Classes to Confine Untrusted Applications

Anurag Acharya and Mandar Raje, University of California at Santa Barbara

Confined execution environments, also known as sandboxes, are one approach to protecting a system from untrusted (possibly malicious) applications. Unfortunately, the compromises between ease of use and integrity are numerous in this approach, especially if the implementation aims toward a usable interface. In his presentation, Anurag Acharya discussed MAPBox, a confinement mechanism that groups applications into behavior-specific classes.

MAPBox depends on the application providers to specify the functionality of the program, and the user is responsible for providing a set of resources that satisfies that functionality. Acharya noted that the idea of MAPBox is loosely derived from MIME types. Thus, the providers supply the user with a MAP type. The user is then able to associate a specific sandbox with that MAP type. If the application attempts to access a resource that was not part of the MAP type's description or not in the sandbox, it is not allowed to run.

Acharya noted that while MAPBox is very customizable (via a sandbox description language), it is also relatively easy to use, since the sandbox allocated for a process is predetermined by its MAP type. Acharya concluded by saying

that MAPBox performed well both in terms of overhead and by stopping only those programs that attempted to violate the terms of their MAP type.

A Secure Java Virtual Machine

Leendert van Doorn, IBM T. J. Watson Research Center

Leendert van Doorn has designed a Java Virtual Machine that provides hardware fault isolation of protection domains – namely, the Java classes. In addition, van Doorn's JVM provides access control for method invocations, inheritance, and system resources; a minimal trusted computing base (TCB); and security mechanisms that do not depend on the correct implementation of the bytecode verifier.

The trusted base in van Doorn's JVM comprises a Java nucleus and a Paramecium kernel. The former offers services like memory allocation, garbage collection, and verification of method invocations that cross protection domains. The Paramecium kernel provides such things as event, memory, and namespace service.

One interesting example of the JVM that van Doorn presented in his talk involved the issue of data-sharing across classes that belong to separate protection domains. In such a case, dereferencing a variable that belongs to a domain other than the class in which it is dereferenced causes a page fault. That page fault is intercepted, at which point a copy of the variable is copied to a new page, where the copy is shared and all future references are updated. Van Doorn noted that since this occurs at binding-time only, the overhead is a one-time expense.

Encrypting Virtual Machine

Niels Provos, University of Michigan

Niels Provos presented a very interesting paper dealing with the security of virtual memory and backing store. Over the course of the presentation, Provos demonstrated the problem by sharing the results of a dissection of the backing store on several systems that had been running at CITI. In those swap partitions, Provos discovered login passwords (some several months old), PGP passphrases, and keys from an ssh-agent, among other things. Thus, the need for a mechanism to protect the backing store was evident.

Instead of depending on users to provide their own encrypting pagers or requiring the VM system to page out to a cryptographic filesystem file, Provos decided to adapt the UVM virtual-memory system of OpenBSD. He discussed his rationalization for choosing Rijndael as the cipher for his system, how volatile keys are created from OpenBSD's entropy pool (using arc4random), and the overhead of the implemented system. As to the overhead, Provos said he runs the encrypted virtual-memory system on all of his machines and does not notice a difference in performance.

As is often the case with Peter Honeyman's students, Provos did not escape the presentation without having to answer one of his advisor's questions. Honeyman questioned why Provos made no mention of pertinent work by Peter Chen (University of Michigan) concerning the persistence of RAM after poweroff. The audience was amused by Provos's response: he had prepared a slide exactly on that topic, just in case his advisor decided to put him on the spot.

Déjà Vu – A User Study: Using Images for Authentication

Rachna Dhamija and Adrian Perrig, University of California at Berkeley

In computer security systems, humans are often the weakest link. This is especially true when the average user must juggle up to 50 different PINS and passwords – they resort to using one common, usually guessable, password. If this sounds ridiculous, the user study in the paper may amaze you. Clearly, reasoned Rachna Dhamija in her introduction to Déjà Vu, password-based authentication is far from ideal.

Instead of focusing authentication schemes on remembering certain exact phrases and character strings, Dhamija decided to exploit a human strength: recognition. Thus, the Déjà Vu system is based on a user selecting his "portfolio" of images (one system used randomly generated ones, another used photographs), and being able to recognize that portfolio when mixed with other images. A challenge set is produced, partly from a set of foreign images and partly from the user's portfolio. The user then must select the images that belong to his portfolio in order to authenticate himself.

In their test sets, more users forgot their usernames (let alone their passwords) than their portfolios. Aside from this obstacle, Dhamija and Perrig discovered that while photos were easier for users to recognize, they were also substantially less secure, since several users from the Bay Area chose a photograph of the Golden Gate bridge as one of the images in their portfolio. After just one week, the image-based authentication scheme was outperforming password and PIN authentication in terms of users remembering their passwords/portfolios and successfully logging in.

INVITED TALK

The Insecurity Industry

Duncan Campbell, IPTV Ltd., EPIC, and International Consortium of Investigative Journalists

Summarized by Mike Brown

The fact that governments around the world spy on their citizens and the citizens of other countries is not new. The extent to which they do it and the methods that they use are shocking, however. Duncan Campbell, a well-spoken journalist, gave an eye-opening look at the

past, present, and future of Communications Intelligence (COMINT) throughout the world.

Campbell has been reporting on the intelligence community for over 25 years and has inspired the wrath of some of the organizations he studies. Britain's Government Communications Headquarters (GCHQ) once tried to put him in prison for 30 years because he reported on them. Now GCHQ, the National Security Agency (NSA), and similar groups around the world are actively promoting themselves. As Campbell reported, they need to hire people, too, and so they are competing against private corporations.

Campbell discussed the development of COMINT over the past 50 years. Originally based around high-frequency collection, agencies currently use submarines, microwave towers, and fiber optics to collect information.

One of Campbell's claims to fame was breaking the story of the Echelon network to the rest of the world. Strictly speaking, Echelon is not the worldwide COMINT and Signals Intelligence (SIGINT) networks, but refers to the collection of information from commercial satellites. The Echelon network involves the U.S., Canada, the UK, Australia, and New Zealand and includes sites around the world for tracking communications. The most fascinating aspect of this network, though, is that the listening stations are mostly automated, as illustrated by a video from New Zealand TV shown by Campbell. The Echelon sites consist of computers tied into satellites that listen in on communications and report their findings. At last count, the Echelon network may include up to 140 ground stations around the world. Even as we speak, new sites pop up all the time.

Campbell's talk then moved on to other methods of collection. One of the more intriguing ways of gathering information revolves around the tapping of fiber optic communications lines on the ocean floor. Ships such as the older USS Halibut would place equipment onto submarine cables to allow the NSA to listen to the signals being sent along these lines. The *USS Jimmy Carter* is having hundreds of millions of dollars of overhauls done to it for similar purposes. Considering how much money is involved this seems to imply that the U.S. and other countries have effective methods of tapping into fiber-optic cables.

The Internet is the next obvious place for governments to gather information. Campbell noted that Internet traffic often routes through the United States, because of Net topography as well as traffic levels. As a result, the U.S. government had ten network interception sites set up as of 1995. Even companies like Bell and MCI were involved with these sites.

Intercepting communications on the Internet presents an interesting problem. A massive amount of traffic flows across the Internet every day, so intelligence agencies need special-purpose systems to filter and find the information that is relevant to them. Dictionary computers address this problem. One example Campbell gave was of the TextFinder computer. It can search trillions of bytes of text for patterns and words, and filter gigabytes of live-stream data each day looking for complex patterns. This system can handle data and fax transmissions but not voice.

Unlike what movies seem to indicate, the NSA can't search through phone communications for voice keywords. They can do voice recognition, but they need samples to train the system with to build a voiceprint. The problem is that telephone conversations are often hard to understand and include shorthand that people use in conversation every day. Machines have trouble understanding this. So instead of going for voice keywords, research now is on topic recognition. In that system, a computer uses a statistical model to determine if a current conversation is of an "interesting" topic. The conversation then gets fed to an analyst who finishes the processing.

To put the amount of work involved into perspective, Campbell gave some hard numbers about the systems used. DERA, the UK Defense Evaluation and Research Agency, had constructed a 1-terabyte system that is used to store 90 days of USENET traffic. The NSA is planning a 1000-terabyte system that will be used to store months of Internet traffic. It should be delivered sometime in 2001.

The Echelon system also captures a lot of traffic. The following figures are from 1992 (the most recent available) and relate to a single intelligence-collection system. Each half hour the site produces 1,000,000 inputs. Of these, only 6,500 pass through the filters and 2,000 of the remaining are forwarded to analysts. They study 20 of those and produce two reports. Imagine how much traffic is being processed now.

It isn't just communications-intelligence groups that are interested in gathering information. The International Law Enforcement Telecommunications Seminar (ILETS) meets yearly to discuss ways to keep wiretapping and key escrow built into telecommunications standards. The FBI's Carnivore system is a result of the ideas from these meetings.

What does the future hold for Communications Intelligence? Some new methods include information-stealing viruses, purposely adding bugs to software, and adding backdoors to products. Campbell gave Lotus Notes as an older example that used a 64-bit key for encryption but sent the first 24 bits of the key with each message, encrypted under the NSA's public key. This allows the NSA to easily break and read any message they want.

In conclusion, Campbell discussed the laws covering COMINT and SIGINT.

The NSA has a mandate not to spy on US citizens, but in the wired world, the rules of who and in what circumstances someone is a citizen are blurred. Current COMINT and SIGINT methods violate the privacy afforded people under the Universal Declaration of Human Rights. Campbell suggests that the infrastructure is already in place, so, for example, the NSA would not need to spy on the European Union; they could ask the state in question to find the information for them. The key then is cooperation.

As would be expected from this talk, the questions from the audience were quite varied and specific. One person asked Campbell what the solution to the privacy problem is. Campbell suggested that it would take 10 to 20 years for any changes to take place, and the solution must revolve around the standard of law. A citizen's right to privacy is very important. Campbell would hope to see the outlawing of SIGINT against another state and replace it with a collaborative system. But the most important goal is public awareness. It is through public awareness that changes occur.

Another question was about the UK's new wiretapping and key escrow laws. Campbell acknowledged that there is a conflict there between the UK accepting the EU's declaration of human rights at the same time as it passes a law forcing its citizens to turn over their encryption keys if asked to. Campbell suggested that it would probably take litigation against the government before the new human rights laws are enforced.

For more information about Duncan Campbell and his work, I'd recommend visiting his informative Web site at <*http://www.gn.apc.org/duncan/*>, or reading the document he prepared for the European Parliament, at <*http://www.europarl.eu.int/dg4/stoa/en/publi/pdf/98-14-01-2en.pdf*>. I would also recommend the documents on surveillance technology at <*http://www.europarl.eu.int/dg4/stoa/en/publi/default.htm#up*>.

REFEREED PAPERS

SESSION: DEMOCRACY

Summarized by Doug Fales

PUBLIUS: A ROBUST, TAMPER-EVIDENT, CENSORSHIP-RESISTANT WEB PUBLISHING SYSTEM

Marc Waldman, New York University; and Aviel D. Rubin and Lorrie Faith Cranor, AT&T Labs – Research

Publius is a complete Web-publishing system, offering anonymity, editability, and security for the authors of online documents who wish to remain out of the public eye. It does this with ordinary browsers and a client-side proxy to facilitate viewing. The publisher encrypts documents, then splits the key into several shares, distributing a share and a copy of the encrypted document to an array of servers. The document arrives at the server as a jumble of encrypted data – certainly nothing that the hosting site could trace to a source or examine for content.

In analyzing the possible weaknesses of Publius, Waldman pointed out that in order to reduce the threat of DoS attacks, each publishing command was limited to 100K. In addition, Publius does not protect the publisher from identifying himself in the content. However, Waldman did make an interesting point about the permanence of a Publius document. Once a document is published without the option to update or delete, it is impossible for the publisher to remove or update it.

The Perl source code (about 1,500 lines) is available at: <*http://www.cs.nyu.edu/~waldman/publius.html*>.

PROBABILISTIC COUNTING OF LARGE DIGITAL SIGNATURE COLLECTIONS

Markus G. Kuhn, University of Cambridge, UK

In an effort to make electronic partitions more practical, Markus Kuhn presented a method to count signatures probabilistically. In effect, Kuhn's method condenses millions of signatures into thousands or hundreds, eliminating duplicates in the process.

Kuhn made the distinction between voting and petitioning clear; his method works because the exact number of signatures is not critical to the outcome of the petition. Whereas an election might be drastically affected by a miscount of one, a petition may be off by several signatures and still serve its purpose. Kuhn's scheme produces verifiable results from a very large collection of signatures that fit into a file of less than 100 kilobytes. This method does depend on the difficulty of generating more than one unique key per user per message.

Kuhn brought up some interesting points regarding the security of his method. Because the distribution of signatures in the counting "slots" are dependent on the text of the message, a group of signers might conspire with the authors to produce a document for which their signatures produce an abnormally high count. Also, certain signatures might be highly valued for their ability to fill a slot, and therefore some signers might be susceptible to being bribed for their signatures.

Among the possible applications that Kuhn mentioned are Web-page metering, TV ratings, and ranking newsgroup contributions.

CAN PSEUDONYMITY REALLY GUARANTEE PRIVACY?

Josyula R. Rao and Pankaj Rohatgi, IBM T. J. Watson Research Center

Using techniques to analyze linguistics and stylometry, Pankaj Rohatgi demon-

strated in his presentation that a substantial amount of identifying information may be gained from supposedly pseudonymous text. Anonymizing agents, text filters (which remove obvious identifying information from text), and traffic shaping are all important to maintaining one's anonymity, yet they all ignore one very serious source of identity information leakage: the text of the document.

Rohatgi showed how pseudonymously signed text could be linked to text of the same author by syntactic and semantic analysis of the writing. With large enough text samples, observing details like vocabulary, sentence size and structure, and spelling errors can tell much about an author. Rohatgi and Rao used a technique whereby a group of function words (e.g., "about," "are," "does," "more," "such") were observed in the text to determine their usage and frequency. Their methods allowed them to correctly group by author as many as 80% of pseudonymous newsgroup postings. Not surprisingly, when the same tests were run on RFCs, the results were not very good. Rohatgi speculated that this may be due to the difference in the formality of the two datasets – newsgroup postings allow more of an author's style and syntax habits to leak through.

The two classes of identity-information leakage, syntax and semantics, can be guarded against in several ways. As for syntax, Rohatgi suggested that spell-checking and thesaurus tools (to avoid being linked with certain vocabulary usage) would be a big improvement. Semantic leakage, on the other hand, is a more difficult issue. One humorous (but possible) suggestion Rohatgi made was to use a language translator to put the document from English to a foreign language and back to English again. In general, though, if your documents are below a certain word limit, it is quite difficult to use stylometric techniques to identify you, Rohatgi said.

INVITED TALK

Trust-Management Pitfalls of PKI

Mark Chen, Securify

Summarized by Doug Fales

In his critical analysis of public-key infrastructure, Mark Chen presented a central, recurring theme: liability management. His talk was a clear presentation of what PKI is, justifications for its use, common public-key algorithms, types of PKI systems, and how to select effective certification-authority policy.

The two strong justifications Chen cited for using a public-key system were (1) the need for explicit, self-authenticating data transactions and (2) the need for nonrepudiation. If these are not central to your reasons for considering such a system, said Chen, you may want to rethink your plans.

As for specifics of public-key systems, Chen mentioned at least five different algorithms, and explained briefly the similarities and differences among them. Further, he emphasized that for all the algorithms, correct implementation is just as crucial as their cryptographic strength. He also went over the basic models of authentication, including hierarchical and relational systems. Throughout, Chen maintained that the verification model must match the liability model. Otherwise, he stated, you are receiving a worthless service.

Chen spent a good portion of the presentation going over the characteristics of good and bad certificate policies. The more explicit a policy, the better. Certificate extensions merely introduce complexity and are therefore a bad idea. Furthermore, a good policy manages liability as well as technology. Finally, if a certification authority seems unwilling to take responsibility for its own security failures, or would rather claim compliance with a policy as its only security obligation, then it may not be offering you anything at all.

In summary, Chen reiterated that certification is about liability management and stressed that PKI is not a universal solution to the authentication problem. During the questions, Chen said that he believed PKI is a very useful technology (despite the tone of his presentation), although he still cautioned that our ambitions sometimes ignore the actual capabilities of our technology.

REFEREED PAPERS

SESSION: HARDWARE

Summarized by David Wragg

An Open-Source Cryptographic Coprocessor

Peter Gutmann, University of Auckland, New Zealand

Peter Gutmann began by describing the motivations for the use of cryptographic coprocessors. Current popular general-purpose operating systems do not provide a high degree of protection for cryptovariables, making it difficult to ensure the security of software-only crypto implementations. In order to avoid this problem, certain parts of a crypto implementation are moved from the host computer into a cryptographic coprocessor.

The types of coprocessors are categorized into tiers according to the operations delegated to them by the host. Higher tiers take on more crypto-related functionality; this gives better protection for cryptovariables and better assurances that the coprocessor will not perform undesirable operations (signing a false message, for example).

Tier 1 coprocessors only store the private key and perform private-key operations. Smartcards, with their limited computing resources and storage capacity, typically fall into this class.
Tier 2 coprocessors also take on bulk encryption operations, thus preventing all cryptovariables from being exposed on the host.

Tier 3 coprocessors perform higher-level operations, such as certificate generation and signing or encryption of a message. Tier 4 coprocessors provide facilities for command verification, so that the device will only act on commands from the host with direct approval from the user. Tier 5 coprocessors provide application-level functionality (though at this point the coprocessor may well require a general-purpose operating system, with the security weaknesses those tend to contain).

In his description of these categories, Gutmann exhibited typical devices from some of the tiers. Next, he went on to give an overview of the options available for constructing cryptographic coprocessors from COTS hardware running open-source operating systems, ranging from tiny (and expensive) embedded PCs to conventional PCs connected to the host computer via the parallel ports or a dedicated Ethernet connection. After describing the software requirements of such coprocessors, Gutmann introduced the design issues for programming interfaces on the host; principally, the interface should avoid complex techniques that might lead to security problems that are due to implementation bugs.

Gutmann then talked about some other issues related to coprocessors. A trusted I/O path between the user and the coprocessor may be needed in order to pass passwords or PINs without exposing them to the host. Physical security may also be a problem; Gutmann described the measures used by the tamper-proof case of one high-end coprocessor.

Gutmann concluded by describing approaches to accelerating public-key encryption in a coprocessor using commodity hardware, with FPGAs, general purpose CPUs, or DSPs.

Secure Coprocessor Integration with Kerberos V5

Naomaru Itoi, University of Michigan

Naomaru Itoi's talk described work he carried out during his internship at IBM's T. J. Watson Research Center in the summer of 1999. Kerberos is a Trusted Third Party-based protocol; the Key Distribution Center (KDC) is trusted with the keys of all the users in a Kerberos realm. With a conventional KDC implementation, if the KDC host is compromised, then the keys of all the users may be exposed. Itoi integrated an IBM 4758 secure coprocessor into the Kerberos KDC, so that the security of the keys is ensured even when the KDC host is compromised with the attacker gaining full administrator privileges. The 4758 takes the form of a PCI card, and (in the 4758 Model 1) contains 4MB of volatile RAM, 8.5KB of battery-backed-up nonvolatile RAM, and 1MB of nonvolatile Flash memory. It is both tamper-resistant and tamper-responding, with layers of epoxy and metal shielding. It detects attempts to open it and other physical attacks, and responds by wiping the contents of its RAM and battery-backed-up RAM. The coprocessor is fully programmable, and contains a cryptographic accelerator.

Itoi described the design of his implementation, based on MIT Kerberos V5. Since the KDC for a large Kerberos realm may contain more keys than would fit on the 4758, these are stored on the KDC host rather than the coprocessor. However, they are encrypted with a master key. The master key is stored in the battery-backed-up RAM of the 4758, and is never exposed to the host. When servicing a Kerberos request, the KDC host passes the relevant keys, encrypted with the master key, to the coprocessor, which performs the appropriate operation and then returns the results to the host. The 4758 also performs generation of the session keys.

After explaining the exchanges between the clients of the KDC, the KDC host, and the coprocessor, Itoi outlined his security analysis. He described his assumptions (which included the possibility of compromise of the KDC host), then went through various attacks and showed how the use of the coprocessor prevented them.

Next, Itoi covered the performance of his implementation compared with the original MIT Kerberos V5 implementation. His measurements showed that in his current implementation the overhead of communication between the KDC host and the 4758 was higher than the time the coprocessor actually took to perform the operations. Some of the calls from the KDC host to the coprocessor could be combined to reduce this overhead. However, achieving this would require large-scale changes to the current KDC implementation.

Itoi concluded with some of the limitations of the prototype and future work that would need completing before the project could be deployed. In particular, password changing and the administration protocol, used to maintain the Kerberos database on the KDC, have not yet been modified to work with the coprocessor.

Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor

John Scott Robin, US Air Force; and Cynthia E. Irvine, Naval Postgraduate School

John Scott Robin began his talk by explaining the advantages of a secure virtual machine monitor (VMM). A VMM provides multiple virtual machines (VMs) on a single hardware platform, each of which provides the illusion of a full machine to the software running within it. Thus, on a single real machine, separate VMs can run separate operating systems. By constraining the VMs, a

secure VMM can impose an overarching security policy that affects all the operating systems and applications running in the VMs, including popular operating systems that might not be capable of enforcing such security policies themselves.

Robin described the classification of VMMs. Type I VMMs run directly on a bare machine. Type II VMMs run as an application underneath another operating system (the host OS).

Next, he identified the requirements that a processor must meet in order to support Type I and Type II VMMs. One requirement states that the processor must be able to signal to the VMM the execution of instructions that access or change the state of the VMM or host OS (sensitive instructions), so that the VMM can ensure that these instructions are used safely. Unfortunately, Intel's Pentium does not meet this requirement: It has sensitive instructions that are not privileged, that is, instructions that do not trap when the processor is running in nonprivileged mode. By examining each instruction in the Pentium set, 17 sensitive but unprivileged instructions were found. All of these are described in the paper, but in the talk the SMSW instruction was used as an example.

Robin mentioned the possibility of changing the Intel Pentium architecture to make it virtualizable (that is, to make it support the requirements for supporting a VMM). A suggested approach was to allow the processor to be configured to trap on certain instructions, as with the Alpha processor, so that they could be handled by the underlying VMM. This could be implemented by a bitmap with one bit corresponding to each instruction and designating whether the instruction is privileged or not. The bitmap could be initialized for full compatibility with the current architecture, but a VMM could change it in order to meet its requirements.

In the questions following the talk, one person asked whether there had been discussions with any of the manufacturers of x86 processors to find out whether they were interested in making their processors virtualizable. The reply was that no contact had been made with any vendor, but that AMD might be a likely choice.

INVITED TALK

The Practical Use of Cryptography in Human-Rights Groups

Suelette Dreyfus, author.

Summarized by Ove Heigre

Suelette Dreyfus's talk outlined which types of cryptographic tools are in use and where; however, the main focus was on why there is a need for such tools in human-rights groups around the world and how information is moved between human-rights field workers. Through case studies, the audience received some insight into how such groups operate when retrieving testimony from, for example, remotely located witnesses of abusive situations and getting this information out to global institutions like truth commissions. Even though the efforts to conceal the information en route have been creative (e.g., the "Origami technique," where one tears up a piece of paper and hides the pieces within clothing), the information, once obtained by the adversary, could be pieced together, putting the lives of the informants and courier at risk.

The benefits of using modern cryptography to conceal sensitive information and to ensure data integrity should be obvious to the reader. It is, however, not always easy to use such modern tools. Consider the following case study from Guatemala.

A grassroots organization operates out of a small, remote village with the aid of a solar panel and a laptop. Testimony is collected from surrounding villages, which may be several days away by foot and without any electricity at all. In these cases one must rely on trusty pen and paper. The information is then brought back to the laptop to be typed up and encrypted, and the notes are then burned to protect the informants. The most dangerous part of the operation consists of getting the information on the laptop to a more central location before it is analyzed and eventually passed on to truth commissions or organizations such as a UN commission. Should the laptop be stolen or lost during this stage, one faces two possible scenarios:

- The information is not properly secured, and the adversaries may obtain the information on the laptop.
- The information is encrypted properly and will not be available to any outsiders.

When cryptographic tools such as the ever-popular PGP are used, no such breaches have been documented. Human-rights groups all over the world now use modern cryptography to protect sensitive information in at least some phases of their operations. Dreyfus illustrated the use of cryptographic tools with a couple of other case studies from human-rights groups in the Congo and Cambodia.

Which lessons have been learned so far?

> It is possible to teach the groups proper use of modern cryptography, but they must be followed up to make sure proper procedures are followed. Sometimes it is hard to make them understand how and why it is essential to use this technology. Modern technology has traditionally been considered an obstacle by grassroots organizations and this makes them a bit wary.
> No breaches have been reported when proper procedures have been followed.

Cheap off-the-shelf strong-cryptography software is now available. This makes it more accessible to grassroots organization around the world.
The use of IT tools, including inexpensive database and cryptographic packages, has helped to shift the balance of power in favor of the human-rights groups.
Some problems, such as computer literacy, still remain. Activists are often not very proficient in handling such technology. The Roman alphabet is another obstacle for some groups, such as the ones in Cambodia where the alphabet is considerably different. Multiple keystroke sequences are often needed for the shortest of words. Problems like these make the use of computers less attractive, and the result is often that more insecure alternatives will be preferred.
Concepts of security are often very naive. A locked front door protecting a computer without any password protection is considered by some to be secure enough. Who would be able or willing to break down the door, anyway?

Dreyfus ended the talk with information about an ongoing volunteer project called Rubberhose. Rubberhose is free, deniable-cryptographic disk-encryption software for human-rights groups. The software is currently in its alpha stage and runs only on the Linux platform. To illustrate how Rubberhose works, picture multiple layers of "dot-pictures" on top of one another to hide the information in the bottom picture. Here the bottom image would be the data saved on the disk. Volunteers are asked to send email to <*rubberhose@rubberhose.org*> or to check out the Web page at <*http://www.rubberhose.org*>. Your help is needed.

The audience at this talk was not large, but it did ask a lot of questions. Some were wondering about the use of different types of technology not discussed in the talk, such as wireless applications and digital cameras. To what extent are such devices in use? Dreyfus replied that she has not yet seen such technology in use, but concurred that they would be useful in the field. Hand-held devices that could be used together with a digital camera or with templates for conducting interviews would be especially useful since an inexperienced interviewer may forget to ask the right questions to complete a database entry when out in the field.

On the issue of whether or not the use of cryptography would be regarded as suspicious, Dreyfus replied that the encrypted material is not normally transferred over monitored channels in particularly repressive countries, such as Burma or Vietnam, where using encryption could land you in hot water with the authorities. It is just a way of hiding the information until it has reached the final destination. The situation is different in countries such as Guatemala where the authorities cannot stop the local truth commission from using it. A physical object such as a laptop would draw more unwanted attention in rural areas in poor countries such as Guatemala. Smaller hand-held appliances would help ease this hazard.

There are still ongoing abuses to document around the world, and it is also important to document the truth about abuses conducted in the past, replied Dreyfus, when asked a question about the extent of ongoing abuses. Others in the audience asked about existing meetings/
conferences for computer security and human-rights groups, or how one could offer one's help. To Dreyfus's knowledge, no such organized meetings exist, but there are talks of starting up. To help, one can make software or participate in "buddy" systems whereby one acts as an advisor to the groups. The Rubberhose project may be a good place to start.

REFEREED PAPERS

SESSION: INTRUSION DETECTION

Summarized by Doug Fales

Detecting and Countering System Intrusions Using Software Wrappers

Calvin Ko, Timothy Fraser, Lee Badger, and Douglas Kilpatrick, NAI Labs

By wrapping system calls with intrusion-detecting code, Calvin Ko et al. hoped to bypass the problems that user-space ID systems must deal with. Using the NAI Labs Generic Software Wrapper Toolkit, Ko implemented several IDSs in the form of system-call and event wrappers. The wrappers they used are managed by a Wrapper Support Subsystem (WSS), itself a kernel module, which dynamically configures and loads the wrappers as modules. This WSS is the control center for managing the wrappers and activating them once a process that meets the wrapper's criteria is found.

Wrappers are created by a Wrapper Definition Language (WDL), which allows a wrapper to define what sort of events it will be invoked to catch, and what it will do based on the outcome of those events. These wrapper definitions may call for an event to be generated for another wrapper, return the system call, halt the process, or collect relevant audit data, which it may later pass to a user-space IDS. The wrappers themselves are highly customizable, and can be used for many applications.

Ko and his team implemented a few IDSs based on their library and its WDL, and then integrated them to form a multi-component IDS built from individual wrappers. One example was a wrapper system to protect imapd from possible attacks. A specification-based wrapper monitors imapd's execution for certain possibly malicious behaviors (opening files it shouldn't or execve-ing anything at all). At the same time, a sequence-based wrapper monitors the same events,

looking for sequences that do not match a database of normal/acceptable sequences. Both these systems feed into a combined wrapper that makes a judgment as to the relative danger based on the input. If the two input wrappers were convinced that an attack was occurring, the combined wrapper would kill the process.

Ko demonstrated that there are several advantages in implementing an IDS at the kernel, and in conclusion he pointed out that the overhead was not a major factor. One of the team's goals was portability, and thus the toolkit is available for FreeBSD, Solaris, Linux, and NT. In the future, Ko hopes to devote some time to developing more wrapper systems that use several different ID methods simultaneously to improve detection accuracy.

Detecting Backdoors and Detecting Stepping Stones

Yin Zhang, Cornell University; and Vern Paxson, AT&T Center for Internet Research at ICSI

Yin Zhang presented a very interesting two-part talk on detecting backdoors and stepping stones by passively monitoring traffic on a network. In the first part, Zhang discussed his algorithm for detecting backdoors based on the packet size and timing of network packets. The algorithm leverages off the fact that keystroke packets are generally between a couple and 20 bytes in most interactive sessions. Furthermore, the timing of said packets follows a Pareto distribution with infinite variance. Using this information, the algorithm can hunt for interactive traffic on ports that are conventionally reserved for noninteractive service.

Because Zhang and Paxson did not look at the content of the network traffic (only the headers), they were able to apply their algorithm to encrypted protocols like SSH. This had the added benefit of keeping the expense of their detection within a reasonable limit. Also, some packets could be discarded based on the direction of the connection; that is, Telnet sessions are not usually initiated by the server (unless the attacker has already set up a callback), and therefore traffic originating from the server may be discarded in some cases.

Zhang and Paxson created several filters, some generic, some crafted to identify specific types of backdoors (rlogin, Telnet, SSH, etc.). When these filters were applied to network traces from University of California at Berkeley (UCB) and Lawrence Berkeley National Laboratories (LBNL), some astonishing results appeared. Over 400 root backdoors were discovered at 291 sites over a period of only 24 hours. Zhang also developed a filter that successfully detected Napster running on FTP ports.

The approach taken for detecting stepping stones (compromised systems that attackers use to access other systems) was similar to that taken in detecting backdoors. Zhang mentioned this as he moved into an explanation of his second paper. Again, traffic was examined in terms of packet size and timing, but not content, both to avoid unnecessary computation and to allow application to encrypted protocols. Some filtering was also applied to the data. The remaining data is examined by correlating packets that were repeated in a pair of (or several) connections. The candidates extracted by this process are eventually inspected visually to determine whether they are all stepping stones. In most cases, they are.

Zhang and Paxson had similar success with detecting stepping stones. From the LBNL trace they found 21 stepping stones; the UCB data produced about 79. When asked what they were doing about the number of abuses found, Paxson confirmed that the sites that had backdoors or were being used as stepping stones were notified – after that, some took action and some did not. One audience member wondered aloud how UCB and LBNL were ever convinced to allow Zhang and Paxson to sniff their networks. Zhang laughed, "That's our secret!"

Automated Response Using System-Call Delays

Anil Somayaji, University of New Mexico; and Stephanie Forrest, Santa Fe Institute

Biologically speaking, homeostasis is the maintenance of a stable state inside an organism. Anil Somayaji's presentation showed how that same idea can be applied to create a mechanism for automated response to attacks on computer systems. In fact, his approach goes beyond just detecting intrusions, as it invokes homeostatic responses that deal with those intrusions directly, rather than ringing an alarm and killing a process.

Somayaji's implementation of this "computer immune system" is called pH for process homeostasis. pH is a set of extensions to a Linux kernel that monitors system calls for anomalous behavior. One difference between pH and similar anomaly-detection systems that intercept system calls is that pH uses delays to counter possibly malicious activity. Somayaji reasoned that small delays in system calls are undetectable or minor annoyances to users, but at the same time, long delays may indirectly result in network timeouts and program termination, effectively eliminating the threat.

pH has the added feature that it can learn normal behavior on a per-program basis by observing the operation of a system that is known to be secure for a period of time. The profile for each program defines normal behavior, and the profile continues to evolve through a maintained data structure. Eventually, that training data is used directly, after approval by the user. Although it is constantly being improved, the current ver-

sion of pH is Open Source Licensed under the GPL. It is available at <*http://www.cs.unm.edu/~soma/pH/*>.

INVITED TALK

Privacy-Degrading Technologies: How Not to Build the Future

Ian Goldberg, Zero-Knowledge Systems

Summarized by Himanshu Khurana

Ian Goldberg is a chief scientist at Zero-Knowledge Systems and is also pursuing a doctoral degree at the University of California, Berkeley. Goldberg discussed the notion of privacy-degrading technologies and promoted the use of effective privacy principles in current/future technological tools. His talk was enlightening, brought out many unknown aspects of privacy, and presented a somewhat formal definition of privacy, which is crucial to its understanding and enforcement.

Goldberg began his talk with an interesting fact about GPS systems on rental cars that demonstrated ineffective privacy policies in today's technology. Apparently, GPS systems have a history feature that enables the passenger to view not only his own travel route, but also that of a few previous rental-car drivers. Further-more, certain vehicles with ON-star security systems permit the car to be remotely controlled by an ON-star agent that authorizes commands from a user given over the phone – a very ineffective authentication mechanism. Another example of today's technology that ignores privacy issues is Web browsing where Web tracking enables Web servers to deny information to certain accessors. Goldberg then presented the main point of his talk, namely, that privacy must be built into technology and cannot be an add-on feature. Unfortunately, it is not a typical part of specifications yet.

Goldberg then introduced the notion of a Nymity Slider, which is a scale that enables a greater understanding of possible levels of privacy. By these levels of privacy one can judge the amount of information about one's identity that is revealed in a transaction. On the two extremes of the scale lie unlinkable complete anonymity such as a cash payment without any identification, and verinymity, which is a true name identifier such as the social security number that uniquely identifies a person and is hard to change. The interesting middle spectrum includes the notion of linkable anonymity where the true identity of the user is not revealed but her previous transactions can be linked and tracked (e.g., via a Safeway Club card or a Pepsi card).

Another privacy level on the scale is persistent pseudonymity where a name is linked socially or cryptographically for a period long enough for a person to be known to a local group over time. A user can, however, have multiple names (pseudonyms), thus not revealing his true identity to all the local groups. This section of the talk concluded with the idea that it is easier to move up the Nymity Slider than down; that is, it is easier to add user identification rather than keep it private.

In order to promote privacy-aware technologies, Goldberg then discussed five principles of privacy that are supported by various standards in Europe and in North America. These principles are: (1) notification, which is the act of notifying the customer that information regarding her identity is being collected; (2) choice, which is the ability of the customer to voluntarily participate in this information-collection process and that this is a meaningful choice; (3) minimization, which requires that only the data required must be collected; (4) use, which requires the customer to be notified about what the data will be used (or not used) for; and (5) security, which requires the customer to be ensured that reasonable measures will be taken to protect the collected information. Goldberg gave examples to demonstrate the lack of support of one or more privacy principles in current technologies; e.g., if a customer is given the choice of giving out information then, he typically will be able to obtain the service only if he gives out the private information.

Goldberg concluded his talk by saying that the ethical way to develop products and to conduct business would be to follow the five principles of privacy and start as low as possible on the Nymity scale – the fundamental notion being that privacy cannot be added later. In the discussion that followed the talk, some more aspects of privacy were brought up by the audience along with a fear that since the common person doesn't care about privacy it may never be important enough in tool development. One member of the audience pointed out that a customer should be able to view the information collected from her at any time and that she should be able to change and delete it at will as well. Regarding the fear, we can only hope that the security designers realize the importance of privacy-aware technologies and promote their development.

REFEREED PAPERS

SESSION: NETWORK PROTECTION

Summarized by Xinzhou Qin

CenterTrack: An IP Overlay Network for Tracking DoS Floods

Robert Stone, UUNET Technologies, Inc.

Robert Stone presented an overlay network, named CenterTrack, for tracking DoS floods. CenterTrack consists of IP tunnels and other connections used to selectively reroute the interesting datagrams from the edge routers to the special-tracking routers. This mechanism can easily determine where the datagrams come from by observing from which tunnel the datagrams arrive.

Source IP addresses of the attacking packets are spoofed in many DoS attacks, so how to trace back the source of the attack has become very important and challenging. In addition, traceback is difficult on large networks with very high-speed and busy routers. By comparing the advantages and disadvantages of several approaches to track the DoS attack – hop-by-hop tracking, hop-by-hop through an overlay network (CenterTrack), and per-interface traffic flow monitoring – Stone pointed out that the more promising method is hop-by-hop tracking through an overlay network

There are several issues and factors to consider in the CenterTrack designs:

(1) IP tunnelling:
- Unaffected by Layer 2 changes
- Lack of IP tunnel support on some routers
- Authentication issues
- Overhead bits

(2) Ways to accomplish routing:
- EBGP over tunnels
- IBGP indirection
- Using an IGP (IS-IS/OSPF)

The CenterTrack system requires input debugging and IP-tunnel support on edge routers and special CenterTrack routers, which are conceptually adjacent only to edge routers and other tracking routers. Traffic for the victim gets rerouted through the overlay network.

Stone also summarized two points regarding dynamic routing with tunnels:

- Tunnel interfaces never announce or accept prefixes from the tunnel termination address space.
- The tracking router's physical interfaces never announce or accept prefixes that are part of the tunnel interface address space.

For a small network, a single tracking router may be sufficient, and a single-level fully meshed network of tracking routers is required for large ISP backbones. Though a two-level system can also be used, the benefits of the scaling may be outweighed by the introduction of an extra hop.

The CenterTrack can be used in static routes and hop-by-hop tracking. In addition, there is a Packet Capture System, which can help catch most traffic for a specific destination in order to analyze a new attack in detail and record evidence of an attack.

The main advantages of the CenterTrack system are: It eliminates the need for transit-router input debugging; required features are available; it can be made to scale; and it is vendor-independent (other than input debugging).

Stone also pointed out the limitations of the CenterTrack system: It still requires input debugging at edge routers; it changes route (attackers may notice it with traceroute); it is local to a particular backbone; it is difficult to track an inside attack or an attack to a backbone router.

See
<http://www.us.uu.net/projects/security/>
for more information.

A Multi-Layer IPSec Protocol

Yongguang Zhang and Bikramjit Singh, HRL Laboratories, LLC

Yongguang Zhang presented a new protocol, called Multi-Layer IPSec (ML-IPSec) Protocol, which uses access control to allow trusted intermediate routers to read/write selected portions of IP datagrams in a secure manner.

Current IPSec protocol provides an end-to-end security protection from which the intermediate nodes in the public Internet can access or modify any information above the IP layer in an IPSec-protected packet. However, with the emerging class of new networking services – such as Internet traffic engineering, application-layer proxies/agents, traffic analysis, etc., which all need to investigate the upper layer protocol information – the original IPSec protection model has become unsuitable due to its restrictiveness of access to the contents of the IP packets by the intermediate nodes. The Multi-Layer IPSec Protocol is designed to grant trusted intermediate routers a secure, controlled, and limited access to a selected portion of certain IP datagrams, while preserving the end-to-end security protection to user data.

Unlike the original IPSec, in which the scope of encryption and authentication applies to the entire IP datagram payload, ML-IPSec divides the IP datagram into zones that are part of the IP datagram under the same security-association protection, and different protection schemes are applied to different zones. Each zone has its own security associations and private keys that are not shared with other zones. In addition, each zone also has its own sets of access-control rules that define which nodes in the network have access to the zone.

The first ML-IPSec gateway/source will rearrange the IP datagram into zones and apply cryptographic protections. The authorized intermediate gateway can decrypt or modify and reencrypt a certain part of the datagram, but the other parts will not be comprised. When the last IPSec gateway/destination gets the packet, ML-IPSec will reconstruct the original datagram. In addition, ML-IPSec defines a complex security relationship that involves sender, receiver, and those selected intermediated nodes along the traffic stream.

Some members of the audience were concerned about the overhead introduced by ML-IPSec. Yongguang Zhang showed some results of performance analysis – for example, the overhead in CPU load is increased by 8%, the penalty in bandwidth is 2%, and the code size is increased by 7%.

One person asked about key management, and Yongguang Zhang replied that

the current key distribution was managed manually and that they will do further research on the automatic keying and multiparty key distribution.

Yongguang Zhang's home page is <*http://www.wins.hrl.com/people/ygz/*>.

Defeating TCP/IP Stack Fingerprinting

Matthew Smart, G. Robert Malan, and Farnam Jahanian, University of Michigan

Matthew Smart presented a TCP/IP stack fingerprint scrubber, which is a tool to prevent a remote user from determining the operating system of the hosts under protection.

Fingerprinting is the process of determining the identity of a remote host's operating system by analyzing the packets from that host. Different operating systems have different implementations of TCP/IP; the ambiguities can be determined by using specially formatted scans. It is very easy to download such tools, such as NMAP, from the Internet freely. System administrators can use such tools to find security weaknesses; hackers use them for finding exploitable systems, and scan the target system in order to collect information on an entire subnet without raising alarms, then gain access or commit a DoS attack. In other words, it is often the first step in a DDoS attack.

Smart said this fingerprint scrubber is transparently interposed between the Internet and the network under protection. The intended use of the scrubber is to place it in front of a set of end hosts or a set of network-infrastructure components and block the majority of stack fingerprinting techniques in a general, transparent manner.

The fingerprint scrubber modifies or drops packets to remove IP and TCP ambiguities from flows. Additionally, it is transparently interposed in a network and built on top of a TCP scrubber, which maintains a small amount of state per flow. In the ICMP scrubbing, Smart mentioned they normalized rates for all hosts, since some stacks implement ICMP message rate limiting and each may have a different rate. In the TCP scrubbing, they also modified the TCP initial sequence number in the outbound/inbound TCP segments.

This fingerprint scrubber can block known fingerprint scans and is also effective against any evolutionary enhancements to fingerprint scanners. Regarding future directions, Smart said they would integrate this fingerprint scrubber into the firewall and increase the performance by reducing data copies. Another aspect of future work is to quantify limitations of timing issues.

INVITED TALK

Methods for Detecting Addressable Promiscuous Devices

Mudge, @stake

Summarized by Algis Rudys

Mudge began by addressing the problem of network sniffing. The problem stems from the nature of Ethernet as a party line. That is, everyone on a segment can listen in on everyone else's traffic. Without encryption, there are no secrets on Ethernet. Most Ethernet network interface cards (NICs) are well-behaved in this regard, however. They discard packets not intended for them. This is not so much a matter of courtesy as of performance.

It is important to note that even if most connections are encrypted (i.e., using SSL and SSH exclusively), network sniffing is still a risk. Attackers can still get SMB and Windows 95/98 file-sharing passwords, notoriously poorly encrypted, NFS file handles, as well as information on network topology and usage.

A common approach to dealing with such attacks is to use system-monitoring tools to search for and fix security vulnerabilities. However, most attackers fix known vulnerabilities on the systems they compromise, inspiring the saying "compromised systems always run the best." A suggestion from the floor was to buy script kiddies sysadmin books and maybe they'd do a better job.

Mudge, on the other hand, espouses the "war college approach," that "the worst-case scenario should never come as a surprise," and consequently proactive measures should be taken to prevent and detect such intrusions.

He then proposed several strategies for detecting promiscuous devices on a network. All these methods exploit, in different ways, the disconnect between second-layer (data-link layer, i.e., Ethernet) and third-layer (network layer, i.e., IP) protocols, and the distinct addresses they use.

The first strategy is to use DNS. Most sniffers routinely do a reverse DNS lookup on IP addresses that are sniffed as the source or destination of a packet. By sending a packet to a bogus MAC (or hardware Ethernet) address (i.e., one known to be not present on the network) and bogus IP address, and sniffing the network for reverse DNS lookups on the bogus IP address, a sniffing computer promptly reveals itself.

An inherent problem with this method is that the sniffer might delay the reverse DNS lookup or collect the addresses for bulk lookup later. To get around this problem, we instead use a DNS server we control, which is authoritative for the bogus IP address we use. Any queries on that IP address will come from sniffing hosts.

This method has the advantages of having few false positives, working across multiple networks, and not saturating local networks. In addition, most of the work is done by sniffer programs themselves. A disadvantage is that it depends

on a feature that may or may not be present in a sniffer. Also, as mentioned above, the first DNS method may fail if the sniffer delays or batches reverse lookups.

A second strategy Mudge discussed is to exploit anomalies in different operating-system TCP/IP stacks. The first one discussed affects only older Linux systems. If a Linux system in promiscuous mode receives a ping with the correct IP address but an incorrect MAC address, it will reply. If an IP broadcast address is used instead, some versions of BSD will reply as well.

To assuage those who would be disappointed at the lack of a Microsoft bug, fear not! In promiscuous mode, many Ethernet drivers for NT will determine whether to forward a packet to the operating system by examining only the first four bytes of the six-byte MAC address. Hence, the driver will assume MAC address ff:ff:ff:ff:00:00 is ff:ff:ff:ff:ff:ff, the Ethernet broadcast address, and forward the packet to the TCP/IP stack for processing.

The advantage of this method is that there are very few false positives. However, it depends on the sniffer running one of a select number of operating systems. It is also limited to a local Ethernet segment.

To get a more universal method, Mudge looked at how packets are processed by a computer system in normal mode versus promiscuous mode. It turns out that the biggest and most noticeable impact is in performance. Hence, this is the target for the final strategy.

We first ping the machines we are testing to establish a baseline latency for normal operation. Then, we flood the network with chaff packets, containing bogus Ethernet addresses. It is important that these packets should be varied in type, destination MAC address, IP destination, and port; this is to exercise the sniffer program, and make it spend as much time as possible in user mode.

We then ping the machines again, and the machines with sufficiently noticeable differences (plus or minus) between the two latency times are most likely in promiscuous mode. It is curious that a machine that experiences a decrease in latency would be in promiscuous mode, but this is largely due to the design of individual Ethernet NICs.

The advantage is that this method is cross-platform. It is fairly accurate over longer periods of time. Using a sufficiently varied selection of packets will also occasionally crash sniffer programs! However, this can quickly congest and slow down a network. It only works on a local Ethernet segment. It also makes an assumption about the cause of the increase in system load that may not be true.

A final technique is for spotting curious crackers. We create packets that appear to log into a "trap" account, using a cleartext protocol (i.e., Telnet, POP, etc.). We then wait for any subsequent attempts to log into that account. This can indicate the presence of a sniffer, if not the machine it is using.

An audience member inquired how using a switch changes things. Mudge first noted that, while most switches will reject packets with bogus MAC addresses, some will generate the chaff for you. Next, he noted that switches are performance devices, not security devices. Some switches can get sufficiently confused by bogus MAC addresses and revert to a bridging mode, where any security properties are lost.

Another question addressed sniffers as kernel modules. Mudge replied that there is still an increase in latency. The sniffer still needs to examine the packet and eventually get any data to userland. The actual increase depends on the speed of the machine.

There was also some discussion of sniffers that disable the port being sniffed so that it cannot be addressed (i.e., so it never sends any packets). Mudge indicated that in this case, it will already be obvious to the admins that something is wrong. In addition, any addressable ports on the same machine will experience a change in latency. Many IDSs have such a configuration, using the addressable port for administrative or maintenance access.

The program AntiSniff, published by L0pht, is a proof of concept of this idea. It is available at *<http://www.l0pht.com/antisniff/>*.

REFEREED PAPERS

SESSION: EMAIL

Summarized by Admir Kulin

A Chosen Ciphertext Attack Against Several EMail Encryption Protocols

Jonathan Katz, Columbia University; and Bruce Schneier, Counterpane Internet Security, Inc.

At the beginning of his talk, the author pointed out that there is a potentially serious security hole in widely used and trusted security protocols for private communication over the Internet like PGP, S/MIME, PKCS#7, CMS, PEM, and MOSS. Any encrypted email can be decrypted using a one-message, adaptive, chosen-ciphertext attack, which exploits the structure of the block-cipher chaining modes used. To analyze this attack, the author gave us his simple definition of security encryption: the attacker can't do better than attack! He suggested several solutions to achieve this simple goal and protect against this class of attack. In any system, there are multiple points at which an adversary can attack; of course, a system is only as secure as its weakest point of attack, and in this paper the authors argue that this attack is entirely feasible in the networked environment in which these email security protocols are

used. Specifically, the attack exploits the symmetric-key modes of encryption used in all these protocols. The details of the chosen-ciphertext attack were clearly described.

An adversary intercepts a PGP-encrypted message sent to a user and wants to determine the contents of this message. The adversary constructs a message according to the given algorithm and sends this message to the user. Then, the user's email handler automatically decrypts and the message appears garbled; he therefore replies to the adversary with, for example, "What were you trying to send me?" but also quotes the garbled message. The adversary receives this plaintext message, which he wanted, and can use this to determine the original message. The author suggested some possible ways to prevent this attack. The simplest solution is for the user not to quote the garbage message in his reply. Another solution is to demand that all encrypted messages be signed, and not to respond to unsigned messages. Another possibility is to generate two session keys: one for encryption and one for authentication.

PGP in Constrained Wireless Devices

Michael Brown and Donny Cheung, University of Waterloo, Canada; Darrel Hankerson, Auburn University; Julio Lopez Hernandez, State University of Campinas, Brazil, and University of Valle, Colombia; and Michael Kirkup and Alfred Menezes, University of Waterloo, Canada

The market for Personal Digital Assistants (PDAs) is growing at a rapid pace. An increasing number of products, such as the PalmPilot, are adding wireless communications capabilities. PDA users are now able to send and receive email just as they would from their networked desktop machines. Because of the inherent insecurity of wireless environments, a system is needed for secure email communications. The requirements for this security system are influenced by the constraints of the PDA, including limited memory, limited processing power, limited bandwidth, and a limited user interface.

This paper describes the authors' experience with porting Pretty Good Privacy (PGP) to the Research in Motion (RIM) two-way pager, which was shown during the presentation, and incorporating elliptic-curve cryptography (ECC) into PGP's suite of public-key ciphers.

The above-mentioned restrictions of PDAs are very rigorous in the case of the RIM pager: It is built around a custom Intel 386 processor running at 10MHz, has 2MB of flash memory and 304KB of SRAM, and has a fairly conventional keyboard with a 6- or 8-line by 28-character graphical display. Although applications for the pager are built as Windows DLLs, the pager is not a Windows-based system.

After this short description of the RIM pager, the presenter compared performance of ECC operations on a Pentium II 400MHz machine, a PalmPilot, and the RIM pager with timings for RSA and discrete log (DL) operations. The performance of all three families of public-key systems (ECC, RSA, and DL) is sufficiently fast for PGP implementations on a Pentium machine. On the pager, RSA public-key operations (encryption and signature verification) are faster than ECC public-key operations. On the other hand, RSA private-key operations (decryption and signature generation) are slower than ECC private key operations. For example, signing with a 1024-bit RSA key takes about 16 seconds, while signing with a 163-bit ECC key takes about 1 second. ECC has a clear advantage over RSA for PGP operations that require both private-key and public-key computations. Similar conclusions are drawn when comparing PSA and ECC performance on the PalmPilot.

The system implementation also has a few weak points: Key management is too simple, the random-number generator is weak, and no serious effort was made to minimize the size of the programs loaded to the pager, etc.

The main conclusion is that PGP is a viable solution for providing secure and interoperable email communications between constrained wireless devices and desktop machines.

Shibboleth: Private Mailing-List Manager

Matt Curtin, Interhack Corporation

At the beginning of his presentation, Matt Curtin gave the motivation for Shibboleth, a program to manage private Internet mailing lists. He asked, "Why yet another mailing-list manager?" Well, differing from other mailing-list managers, Shibboleth manages lists or groups of lists that are closed, or have membership by invitation only. So instead of focusing on automating the process of subscribing and unsubscribing readers, Curtin includes features like SMTP forgery detection, prevention of outsiders' ability to harvest usable email addresses from mailing-list archives, and support for cryptographic-strength user authentication and nonrepudiation.

After that, Curtin explained the terminology and design goals of his system. For example, Shibboleth thinks of lists in groups. These groups of mailing lists on the same machine, managed by the same installation of Shibboleth, are called families.

Each user should have a standardized address, in the form of "prefix-nym," so, nobody knows the user's real address except the list administrator. All mail sent this way is subject to the same defenses as mail sent to a list. Each member of the list has a list of patterns used to identify his known address. When a message arrives, the "From" header is compared to patterns in the profiles in

the database so that the user who sent the message can be identified. Each list has the option of having all of its traffic PGP signed. That is, before Shibboleth sends a message, its PGP signs the message with its own key, so cryptographic strength moderation requires the PGP signature of a valid moderator.

Curtin limited his focus to the implementation details that he believes to be the most relevant to his goals of privacy and security, in other words the features that are not provided by other mailing-list managers.

Curtin also identified some weak areas where Shibboleth could be improved: an error in PGP key storage, reducing necessary trust in administrators, the need to support OpenPGP, intolerance of SMTP irregularities, etc. On the whole, Curtin showed that it is possible for a group of people who wish to keep to themselves can do so, even on today's Internet.

A good time was had by all at the conference reception . . .