5th USENIX Workshop on Offensive Technologies (WOOT '11)

August 8, 2011
San Francisco, CA

Attacks on Networks and Networking Equipment

Summarized by Rik Farrow (rik@usenix.org)

Media Access Control Address Spoofing Attacks against Port Security

Andrew Buhr, Dale Lindskog, Pavol Zavarsky, and Ron Ruhl, Concordia University College of Alberta

Andrew Buhr explained how enabling port security increased the chances of success for an attacker when spoofing MAC addresses. Port security means giving a higher precedence in a switch-based lookup table over non-secure MAC addresses. In their experimental setup there are three switches, with two edge switches set up with port security and a third switch used to connect the edge switches configured without port security. Cisco advises that configuration, for several reasons.

Andrew described two of the three attacks that appear in their paper. In each described attack, the attacker is connected to the same switch as one of the victims. The second victim is connected to the other edge switch. When the first victim sends an ARP request to the second victim, the attacker can replay the same ARP reply. Because the second victim's reply comes via the relaying switch, the response is considered non-secure. So the attacker's spoofed ARP reply results in associating the second victim's MAC address with the attacker's switch port, allowing the attacker to impersonate the second victim.

Andrew suggested several techniques as defense strategies, with the preferred method being segregating trusted and non-trusted nodes into their own broadcast domains. Enabling port security on the interconnecting switch is not recommended by Cisco, because it disables channel bonding and dynamic port reconfiguration.

Mike Ryan (ISI) wondered if the second attack would work if there were two attackers, each connected to a different edge switch, with a private backchannel. Andrew expected this attack would work, as long as the attackers could forward frames quickly enough that there were no retransmissions of frames. David Brumley asked if the version of IOS made any difference to the attacks, and Andrew said that it did. David then asked about availability of code, and Andrew replied that no code is necessary, as all the attacker needs to do is change the MAC address for the attacking system's interface.

Fragmentation Considered Vulnerable: Blindly Intercepting and Discarding Fragments

Yossi Gilad and Amir Herzberg, Bar Ilan University

The researchers took a new look at an old problem. In the 1990s, there were several well-known DoS attacks that relied on problems with IP fragmentation: Teardrop, Rose, and the Ping of Death, all based on implementation mistakes. In this work, the authors rely more on specification issues.

IP fragmentation is best avoided, but still occurs today. With IPv4, any router can fragment packets, and in IPv6, only the sending host can fragment packets. ICMP is used to determine Path MTU to avoid fragmentation, but fragmentation can still occur with UDP and when packets are tunneled.

The key to their attack is to determine the IP ID. This is trivial with Windows, which uses a monotonically increasing IP ID. Linux uses a per-destination IP ID, which makes determining the IP ID more difficult. In their attacks, they make use of a sandboxed script, PuZo, on the victim's network, to watch for fragments that do not show up. The missing fragments must have had a valid IP ID, and thus are not directed to PuZo. Their attack requires O(square root of N) probe packets.

Mike Ryan asked if this attack worked against firewalls, and the speaker said it does. Someone else pointed out that Linux can be patched to use randomized IP IDs, and the speaker replied that this can cause collisions and be as bad as an attack.

Killing the Myth of Cisco IOS Diversity: Recent Advances in Reliable Shellcode Design

Ang Cui, Jatin Kataria, and Salvatore J. Stolfo, Columbia University

Ang Cui presented this paper, which describes a very effective exploitation of Cisco routers. Internet infrastructure is highly reliant on Cisco routers,and cannot be defended against attacks like this as the use of a firewall or IDS is, in many instances, not possible.

Felix Linder (FX) has pointed out that ASLR (address space layout randomization), as well as the many different versions of IOS, make successful exploitation of Cisco routers difficult. The authors estimate that there are over 300,000 binary versions of IOS. Yet IOS is a functional monoculture: in any router you will see the same behavior when you enter the enable command. FX created a disassembling shellcode that relies on finding a known string, then searching for the address of this string in code, and, finally, replacing the

instruction that reports the receipt of the correct password with an instruction that always reports success.

The problem with that attack is that it takes a long time. IOS includes a watchdog timer, to guard against runaway processes, with a two-second limit. The two linear searches through memory used by FX's exploit often trigger the watchdog timer, killing the process.

The authors' approach relies on a single search of a more limited amount of memory. Interrupt handlers use the eret (exception return) instruction, and their shellcode searches for these instructions and replaces them with hooks into their own rootkit. As this search and replace occurs quickly and future execution is distributed across many processes, because it relies on interrupt handlers, the attack does not get caught by the watchguard timer.

The first stage rootkit monitors packets punted to IOS—any that cannot be handled by line cards. If these packets contain a 32-bit magic number, the next four bytes are used as an address, the following byte a flag, and the rest of the packet is loaded into memory as executable code. Using these packets, a more full-featured rootkit can be loaded into the router's memory and controlled using punted packets labeled with the magic number. But before this can be done, the first stage rootkit must return the locations of the eret instructions, as these provide a fingerprint that identifies the specific version of this binary instance of IOS. The second stage rootkit is tailored for this binary instance.

Ang described a possible defense against their attack, the creation of "symbiotes." The symbiotes run checksums on the invariant portions of IOS to detect the installation of rootkits, and this is future research for the authors. A video (http://www.hacktory.cs.columbia.edu/ios-rootkit/) of a successful attack was shown, eliciting applause.

Adam Drew (Qualcomm) asked Ang to explain the principle behind their proposed defensive software. Ang said that their defense was designed to work on blackbox systems, like IOS, where the internal workings are unknown. They can intercept a large number of returns, perform checksumming, and rely on having enough symbiotes to make it very difficult for an attacker to disable or avoid them all. Amiz Herzberb (Bar Ilan University) wondered if increasing the number of IOS versions might help, but Ang said that their attack relies on a database of versions, and increasing the version space makes little difference to their attack. Someone else wondered how successful the first stage attack needs to be to collect fingerprints, and Ang answered that that depends on the particular attack. They used a synthetic attack for their demo. Someone else asked how they created the final rootkit using the fingerprints, and Ang answered that this can be done automatically once the binary version is discovered.

SkyNET: A 3G-Enabled Mobile Attack Drone and Stealth Botmaster

Theodore Reed, Joseph Geis, and Sven Dietrich, Stevens Institute of Technology

Theodore Reed described, and later demonstrated, the use of an inexpensive drone to enlist participants in a botnet. A collection of such drones would be called a SkyNET, but the authors built only a single drone, using an off-the-shelf AR.Drone quadracopter platform, a TS-7550 single board computer with 3G, GPS, and two WiFi cards. The drone is intended to fly around an area searching for WiFi networks. Breaking WEP and WPA encryption is offloaded to the cloud (EC2), and an OpenVPN connection over the GSM link is used for communication back to the command and control (C&C) system.

Their demonstration model can fly 2.7 kilometers. During testing in New York City, they found over 1700 access points near the Empire State Building and another 1100 in a residential area. Ted commented that "just flying the drone attracts victims," who came up to them as they flew the drone from city parks.

They included a couple of mechanisms to protect against the accidental loss of the drone, and potentially the information against the systems it had compromised. The drone includes a list of pairs of asymmetric keys, and the keys are randomly assigned to bots with the ID of the keys kept at the C&C system. Ted concluded by saying that without any engineering background, they had built a usable drone for about $600 that could carry out attacks against WiFi-enabled systems.

David Brumley asked how noisy the drone is. When the drone was demonstrated, its four rotors were about as noisy as a vacuum cleaner; as Ted said, its noise detracts from its "stealthiness."

Crossing into the Real World: Beyond IP-based Attacks

Summarized by Karl Koscher (supersat@cs.washington.edu)

Getting the Face Behind the Squares: Reconstructing Pixelized Video Streams

Ludovico Cavedon, Luca Foschini, and Giovanni Vigna, University of California, Santa Barbara

Ludovico Cavedon presented this paper, which looks at the effectiveness of pixelization filters for video. These filters are often used to obscure private or censored information (e.g.,

faces and license plate numbers) while remaining somewhat aesthetically pleasing and keeping the broader image context intact (as opposed to using a black box, for example). While it is often assumed that these filters cannot be inverted, this paper demonstrates that in many cases, close approximations of the original images can be reconstructed.

The paper makes the following assumptions: first, the image being pixelized must be approximately fixed (e.g., a license plate). Second, the pixelized area of the image must be fixed between frames. Finally, there must be some small motion between frames of the image.

Since pixelization is a linear operation, a naive approach is to simply build a system of equations describing the pixelization and solve it approximately. However, this produces unsatisfactory results (see figure 4(c) in the paper for an example), due to quantization error. Instead, the approach presented uses the maximum a posteriori method, which takes advantage of the fact that the solution is not arbitrary but represents an actual image.

Ludovico concluded his talk with several impressive demos of their technique, which are also shown in the paper. He also discussed issues with real-world video, such as compression artifacts, image rescaling, and moving subjects.

Heat of the Moment: Characterizing the Efficacy of Thermal Camera-Based Attacks

Keaton Mowery, Sarah Meiklejohn, and Stefan Savage, University of California, San Diego

Sarah Meiklejohn presented this paper, which looks at the effectiveness of using thermal imaging technology to recover codes entered into code entry devices (e.g., ATM PIN pads). While previous work demonstrated that this type of attack was feasible against a particular type of safe, this research dives deeper and evaluates the effectiveness along four different axes: different types of material, different types of people and their code entry techniques, different scales of attack (e.g., automatically recovering hundreds of PINs versus manually identifying one), and different degrees of success (e.g., recovering the code with multiple attempts).

The attack works poorly against metal, which reflects and dissipates heat rather well. Therefore all of the results presented were for plastic and rubber keypads only. For their experiments, they recruited 21 people to enter a total of 27 codes (seven of which had repeated digits). They discovered a wide variance in the amount of heat transferred by different people. In all of their experiments, automated recovery outperformed manual analysis. Recovery of perfect codes was rather low (~15% for manual analysis, ~35% for automated analysis), even immediately after code entry. Recovery of codes with one error (a substitution or transposition) was also low (~25% manual, ~50% automatic). Recovery of perfect key combinations (but not necessarily order) fared significantly better, at over 80% right after entry.

Several people asked which would be the safest ATM to use. A busy one with metal keys. Adam Drew wanted to try recording ATMs, to see how well this works, but wondered about the legality.

Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios

Travis Goodspeed, University of Pennsylvania; Sergey Bratus, Ricky Melgares, Rebecca Shapiro, and Ryan Speers, Dartmouth College

Travis Goodspeed presented a new technique that allows an attacker to inject an arbitrary layer one packet into many wireless networks. The attack can be performed on many digital, unencrypted wireless networks where packet lengths are variable and an attacker can cause a higher-layer packet with some arbitrary data to be sent. The idea is simple: if you embed a layer-one packet (including the preamble, sync, and other metadata) in a higher-layer packet and the receiving radio does not detect the outer layer-one packet (e.g., if the sync is corrupted), that receiving radio will often detect and decode the inner layer-one packet crafted by the attacker.

Travis pointed out ways the attacker gets more of an advantage. For example, for power management reasons the outer preamble might be shortened. However, the attacker can generate a significantly longer preamble, increasing the odds that a receiver will lock onto the inner packet. In systems where the sync field is dependent on the recipient, the attack is always successful. Finally, if one receiver has better reception than the other, it's possible to target the receiver with weaker reception without the other receiver noticing.

One proposed countermeasure is to encrypt all wireless links, even if they offer no protection against local attackers. During the Q&A period, Karl Koscher pointed out that using sync patterns that can't be generated by normal data would also be an effective countermeasure. Travis responded that perhaps having different speeds, such as one and six Mbps, would allow you to inject into a network of a different frequency. Someone else asked whether the connection remains misaligned after a successful attack. Travis said no, that the attack works only on a per-packet basis.

Finally, there was a short discussion of how applicable this technique is to Ethernet. While packet corruption is extremely rare over wired Ethernet, Travis hypothesized that finding a source of noise (such as intentionally injecting collisions on an unswitched network) would allow this technique to work.

Targeting the Cloud and Commodity Computing Devices

Summarized by Mihir Nanavati (mihirn@cs.ubc.ca)

Energy Attack on Server Systems

Zhenyu Wu, Mengjun Xie, and Haining Wang, The College of William and Mary

Zhenyu Wu described an attack that forced servers to perform more expensive computations and consume more power. Having made the observation that power consumption was a large percentage of the cost of ownership, he showed that while recent advances in energy-efficient computing had led to significant decreases in the idle power consumption of servers, peak load power consumption has remained more or less unchanged from a few years ago. An attacker could significantly increase operating costs by getting the servers to constantly execute at full load.

This attack was simulated against a local MediaWiki server. Profiling the server showed that less frequent requests were absent from the object cache and took approximately six times as much power to satisfy, compared to requests that could be satisfied from the object cache. By generating such requests, they were able to force a 6–40% increase in power consumption. To achieve stealthiness, the requests were capped at a level that would make them appear to originate from a human user. The number of malicious requesters were also limited, so as not to degrade latency significantly.

Adam Drew compared the work to John Cleese's "How to Irritate People," where the attacker is just doing enough damage to be an annoyance, but not enough to severely hamper operation. He then asked whether the increased load could lead to greater failure rates for hardware. David Brumley wondered about the cost compounding, where an increase in computation increased the heat and cost of cooling required as well. Wu said that they had started exploring this direction a bit, but it was still in its early stages.

Putting Out a HIT: Crowdsourcing Malware Installs

Chris Kanich, Stephen Checkoway, and Keaton Mowery, University of California, San Diego

Chris Kanich presented this analysis of the economics of attracting and exploiting the systems of Amazon Mechanical Turk workers. Unlike normal use of Mechanical Turk to solve problems that are hard for computers to solve, this focuses on exploiting the systems of the Turkers and monetizing them in pay-per-install markets. For any monetary benefit, enough Turker systems would need to be vulnerable to outweigh the costs of actually running these tasks.

Kanich described the tasks used to determine the vulnerability of the Turkers. One asked responders questions about their antivirus, and for an additional payment to run a JavaScript program that collected information about their system and reported it back. Over 80% of the people, spread across geographical regions, were running a vulnerable configuration. Vulnerability was approximated by the existence of published CVEs—an actual available exploit was not necessary. Running another task showed that while over 90% of the respondents had antivirus, only a small fraction of them had up-to-date signatures.

Kanich then said that Amazon Mechanical Turk allows tasks to be offered on the basis of geographical regions, which is a bonus, because compromised systems can be sold for significantly varying amounts on pay-per-install programs on the basis of geographical location of the clients. Any such attack however, is only successful if there is significant uptake among users. Kanich noted that they had around 400–500 people attempt their task, most in India and in the US, within five days.

The questions revolved around whether the pay-per-install community was honest about paying the rates they advertised, and expressing general incredulity over the high percentage of vulnerable systems and the willingness of users to run untrusted code on their systems. Kanich observed that most people may believe that they are protected against any code by their antivirus. In reference to the payments, he replied that the payment rates corroborated other pay-per-install research and were within an order of magnitude of what others had observed.

All Your Droid Are Belong to Us: A Survey of Current Android Attacks

Timothy Vidas, Daniel Votipka, and Nicolas Christin, Carnegie Mellon University

Daniel Votipka presented this work, which surveys the landscape of attacks on Android smartphones and some of the possible mitigations. The smartphone market has seen huge growth, and Android, being both open source and the fastest-growing platform, with approximately 500,000 daily activations, is an important player in the ecosystem.

Android runs every application in a limited privilege sandbox, while any requests for elevated privileges have to be approved by the user. Unfortunately, the permission model requests can often be confusing to users, who are usually in a hurry to just accept and get the application running. Furthermore, even in cases where the user does try to limit permissions, it is often hard because of the generality of the request. Requests often also grant the application more power than is implied; for instance, allowing an application to receive

SMSes allows it to receive them before the standard messaging application and modify the contents accordingly.

Votipka then discussed how allowing carriers to provide updates created artificially large exploit windows, with some phones being patched a full year after the security update had originally been released by Google. Other attack vectors include the developer and debugging interface, which could be used to exploit any phone one has physical access to.

At this juncture, Votipka switched to discussing potential countermeasures, which range from shortening the exploit window, to better privilege handling by having hierarchical permissions or explicit rule checks to flag dangerous combinations of seemingly innocuous permissions. Another proposed idea was to have multiple tiers of applications in the Marketplace, with applications desiring higher permissions required to undergo verification.

Emery Burger asked if Google had been contacted and, if so, what their response was. Don said that a meeting was forthcoming. Adam Drew asked about the usability of some of these countermeasures. Dan said that while there had been several studies, it was not something they had explicitly looked into.

Exploiting the Hard-Working DWARF: Trojan and Exploit Techniques with No Native Executable Code

James Oakley and Sergey Bratus, Dartmouth College

↘ *Awarded Best Student Paper!*

Sergey Bratus accepted the award, saying that James Oakley had done all the heavy lifting in this work on injecting trojan logic into binary executables using the DWARF bytecode interpreter. While DWARF is traditionally associated with debugging information, it is also used for exception handling, and every process created from a gcc-compiled binary with exception handling enabled will load the DWARF bytecode interpreter at runtime. DWARF bytecode is Turing complete and can be used as a backdoor into any such binary. This is particularly dangerous, because antivirus typically overlooks this type of trojan.

Bratus then described how DWARF bytecode, once the attacker has managed to sneak it in, can read arbitrary process memory, can defeat ASLR, can perform arbitrary computations, and is built to influence the control flow of the program. Using this, he demonstrated how a simple program could be exploited such that all code and data sections remained identical but resulted in a root shell when it threw an exception.

Bratus went on to explain the structure of several undocumented or scantily documented header frames, and how exception handling occurred for these binaries. The C++ exception handler has a "personality routine" that decides whether there is a handler at a particular level of a stack frame or whether the stack needs to be further unwound to catch the exception. Modifying the table the routine uses allows the backdoor to remain hidden until an exception is thrown and to return control to any point in the program or its loaded libraries after that.

Bratus concluded by discussing how this class of attack is currently difficult to detect but extremely powerful, because of the inherent power of DWARF bytecode; however, work is underway to mitigate it. He listed several hacker research projects, such as ElfSh/ERESI, LOCREATE, and several grsecurity/PaX-related papers in Phrack, as inspirations.

Rik Farrow asked about the root prompt displayed during the demo, and Sergey replied that they had the DWARF exploit execute a SUID shell, as the root prompt appears more interesting. He then said that there are real exploits out there, not just for C and C++ but also for some Java implementations.

Advances in Low-Level Exploitation

Summarized by Karl Koscher (supersat@cs.washington.edu)

DieHarder: Securing the Heap

Gene Novark and Emery D. Berger, University of Massachusetts Amherst

In this invited talk, Emery D. Berger revisited the DieHarder memory allocator, originally presented at CCS 2010. He began by analyzing the various ways modern heap allocators can be exploited and then described what's new in DieHarder, namely, that objects are immediately destroyed when freed, and that each object is allocated at a highly random location.

The performance of DieHarder was tested and compared to other allocators under two scenarios: the SPECint2006 benchmark suite, and Firefox loading the Alexa Top 20 Web sites from a local network cache. For SPECint2006, DieHarder was approximately 20% slower than other allocators due to it breaking TLB locality. However, for the Firefox tests, the difference between DieHarder and other allocators was not statistically significant, leading to the conclusion that DieHarder is a practical solution for Internet-facing applications such as browsers.

Rik Farrow asked about how DieHarder works, as the memory allocators he was familiar with used linked lists. Emery replied that DieHarder uses hashing and bitmaps

instead of the more familiar heap allocation techniques. Mike Ryan asked about the TLB problem and Emery said that this was specifically an Intel issue. Other architectures allow software-based TLB control, and if Intel didn't fill in the entire TLB, this more flexible approach would help with DieHarder's performance. Mike then asked about how DieHarder leaves traps ("bombs" in the slide) over the entire address space in OpenBSD. Emery said that Linux does this as well, by using lots of unmapped pages that act as bombs.

Vulnerability Extrapolation: Assisted Discovery of Vulnerabilities Using Machine Learning

Fabian Yamaguchi and Felix "FX" Lindner, Recurity Labs GmbH; Konrad Rieck, Technische Universität Berlin

Fabian Yamaguchi presented this solution to a compelling problem: given a known vulnerable function, find all other functions with similar vulnerabilities. Many code bases repeat the same vulnerability mistakes, making this technique useful for finding additional vulnerabilities. The basic intuition is that functions are composed of different usage patterns, and by comparing the dominant usage patterns, you can find functions with similar vulnerabilities.

In particular, each function is represented by a sparse vector whose dimensions map to a particular type or function name. The value of each of these dimensions is simply a 1 or a 0, depending on whether that type or name is used in the function, weighted by the identifier's TFIDF, a standard weighting term used in information retrieval. Then, principal component analysis is used to find the dominant usage patterns. Functions can then be represented as a combination of these dominant usage patterns, and functions with combinations similar to a vulnerable function are likely candidates for vulnerability exploration.

As a case study, the researchers evaluated their technique on FFmpeg. They provided their system with a previously identified vulnerable decoder function and found that the most similar function (at 96% similarity) was also vulnerable. Although this second function had been patched as well, they found that the fifth-most similar function (at 72% similarity) contained the same vulnerability and was not yet patched.

Someone asked if they had only tried this one example, and Fabian said they had tried another evaluation on this data to prove that it works. Nicholas Carlini asked about the level of dimensionality, and Fabian said that they used 200 for dimensionality, which seems to work. In his thesis work, he found that using more produces more similarity.

Exposing iClass Key Diversification

Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult, Radboud University Nijmegen

Awarded Best Paper!

Gerhard de Koning Gans presented a paper that looks at the key diversification scheme built in to the iClass contactless smart card system. The key diversification scheme was known to involve a single DES operation followed by a key fortification function. Through some amount of reverse engineering, they determined that the key fortification function is highly invertible. For a given output of the fortification function, there are an average of four possible inputs that can be easily determined. Thus, the diversification scheme offers little protection over standard DES.

The reverse engineering involved several steps, including extracting the secret Omnikey reader secure mode key and emulating an ISO 15693 card with the ISO 14443 protocol. The main technique used was to emulate cards with slightly different serial numbers and observe changes in the rekeying command sent. While they did not have a DES cracker to verify their results, they were able to use other recently published techniques to extract the master key from a legitimate reader and found that their attack did indeed find the master key.