First USENIX Workshop on Offensive Technologies (WOOT '07)

Boston, MA
August 6, 2007

Summarized by Dominic Spill (dominicgs@gmail.com) and Robert N.M. Watson (robert.watson@cl.cam.ac.uk)

The First USENIX Workshop on Offensive Technologies was opened by Tal Garfinkel. He thanked the program committee and USENIX and gave an overview of what to expect from the workshop.

INVITED TALK

■ ***Fast-Flux DNS and Overlay Networks Using Botnets***

David Dagon, Georgia Institute of Technology

David got the workshop off to a start with a discussion of his current work on botnets, focusing on botnets that rapidly change their DNS responses to avoid detection.

In botnets, which date back to the 1990s, pieces of malicious code (bots) are often spread using Web sites, email, and vulnerabilities. The bots originally communicated with their controller using IRC channels. The countermeasure to this was based on the DNS requests made to find the IRC channel. As these countermeasures were used the sophistication of the botnets increased, and they began using peer-to-peer applications for communication and replication. These botnets were stopped because they had fixed points in their network, and these could be tracked and stopped.

The current generation of botnets use domain names for which the DNS response changes rapidly, with different bots within the network taking the role of server for the others. This is known as Fast-Flux DNS. The botnets avoid having fixed servers and therefore attempt to avoid being shut down; they use themselves to respond to DNS requests and propagate themselves.

These botnets are then used for a number of different applications. Two of the most well known are sending unsolicited email and distributed denial of service attacks. The botnet provides a platform for these applications, which is sold as a service by the controller of the botnet.

David has investigated the locations of the bots in the networks, using IP addresses. He found that most of the bots were in centers of population, where the use of broadband Internet is greatest. So David mapped the growth of the botnets using the IPs that were given in response to repeated DNS requests. These graphs showed that initially only a very small number of IPs were returned, but shortly afterward there was an explosion in the number of IPs returned, as more systems were infected by the bots.

The reason these botnets persist is that they avoid detection by constantly changing, either by repacking the binary or by downloading an updated version from another bot in the network. They are often observed, by antivirus researchers, within virtual machines, but there is a large amount of research into detecting virtual machines, meaning that as research attempts to stop one exploit it helps botnets continue to go undetected.

David has also analyzed the effectiveness of using existing blacklists and user traffic analysis as predictors of infection, and he observed that although there are troubling social issues associated with usage analysis, based on usage patterns some users are more likely to be exposed to, and hence infected by, malware than others. This prompted a healthy discussion of the interactions between privacy and monitoring in malware prevention.

Robert Watson asked about the feasibility of bots communicating using the Tor network or botnets providing stronger anonymity services to protect their maintainers. David said that there are botnets that do this, but the performance of the Tor network is not high enough for the traffic required by most of the networks. Additionally, the botnet managers may use Tor to control the botnets, but the only anonymity they care about is their own, not that of the systems that they have exploited, so they would not use Tor to hide the systems with bots.

FROM THE METAL TO THE INFRASTRUCTURE

Niels Provos chaired this session.

■ ***Flayer: Exposing Application Internals***

Will Drewry and Tavis Ormandy, Google, Inc.

Will presented work on an advanced fuzzing tool, based on Valgrind, with the ability to taint input and skip over checks in the code. He also showed some of the bugs that it had uncovered in libtiff, openssl, and openssh.

The Flayer tool is a combination of a fuzzer, an auditing tool, and a patch analyzer. It can be used in automated testing scripts or as a stand-alone application. Flayer taints input to an application to allow it to be tracked through the execution of the code. It can also bypass the execution of branches to avoid version checks.

The goal of the tool is to find errors in the code of an application without having to get into the depths of how it works. It can be used with /dev/urandom and also files filled with random data, and it can be beneficial to run the tool with a file of random data and then alter a small amount, as little as one bit, before running it again. So far it has been used to find bugs in libtiff, openssl, and openssh. The tool and source code can be found at http://code.google.com/p/flayer.

Niels Provos asked about the degree of automation, noting that it seemed the process of using Flayer was very manual. Will answered that this is a user-assisted analysis tool, relying on the insights of the user into the code. However, it is significantly less manual than tools that require, for example, coming up with a complete specification of correct behavior to generate fuzzing input.

When asked about performance, Will said that he had measured it to be roughly 20 times slower than the execution of the code, which is faster than other debugging applications; it also has high memory usage, which is one of the factors they are currently trying to reduce.

- ***The ND2DB Attack: Database Content Extraction Using Timing Attacks on the Indexing Algorithms***

  Ariel Futoransky, Damián Saura, and Ariel Waissbein, Core Security Technologies

Ariel presented a new attack technique for extracting data from databases using timing attacks. This attack relies on the variable cost of inserting fields into sorted tables based on the I/O cost of b-tree node splitting, assuming that all keys are unique. This service may be available to anonymous users even if select queries are not, and it offers significant efficiency improvements over a simple but computationally infeasible $O(n)$ search. David analyzed the performance characteristics of the MySQL database using the InnoDB storage format on Windows XP, reporting experimental results in which 64-bit keys in the table were extracted in tens of thousands of insert queries.

Tal Garfinkel asked whether this attack could be performed through Web-based front ends to databases, rather than via direct database queries; Ariel answered that they had not experimented with this yet and it would introduce some amount of noise, but if it didn't introduce disk I/Os this effect may be small compared to the cost of node splitting. A general discussion of the effects of noise on this and other timing attacks ensued.

- ***Exploiting Concurrency Vulnerabilities in System Call Wrappers***

  Robert N.M. Watson, Computer Laboratory, University of Cambridge

Robert was at WOOT to unveil his finding on the vulnerabilities introduced into a system by the use of system call wrappers. He explained how the wrappers function and how they are exploited and gave recommendations as to how the problems can be addressed by the operating system authors. He also showed examples of exploits for single and multiple CPU systems.

System call wrappers are used to increase the portability of applications without needing to recompile them on every target system or for systems that may not have kernel source code available. Many applications use them, notably antivirus tools. Robert gave a comparison to resource managers but said that system call wrappers are not atomic with respect to the system call, and this is where the vulnerability is introduced.

Robert demonstrated the existence of two new types of race condition, introduced by system call wrappers, which can be exploited in addition to the well-known "Time of check to time of use" race. "Time of audit to time of use" and "time of replacement to time of use" are the names given to these new race conditions. He had also written example code to exploit these and other conditions for both uniprocessor and multiprocessor systems, showing that Systrace and GWSTK, two commonly used wrapper toolkits, were both vulnerable.

A number of solutions were proposed, such as additional memory synchronization; however, this was shown to introduce more vulnerabilities. Robert recommends moving toward message passing, a move already made by Linux Security Modules (LSM) and the TrustedBSD MAC framework. With this work Robert has shown that system call wrappers are a threat to the security of any system that uses them, and he calls on developers to change their practices.

- ***Billing Attacks on SIP-Based VoIP Systems***

  Ruishan Zhang, Xinyuan Wang, Xiaohui Yang, and Xuxian Jiang, George Mason University

Xinyuan Wang discussed research that he has been doing in the area of SIP-based VoIP applications, especially the billing systems used. This is important because SIP is now the standard used by most VoIP systems, including those sold by Vonage, AT&T, and Verizon. The number of VoIP users is expected to reach 44 million by 2010, all of whom will want their billing to be accurate.

The call setup procedure is similar to that used for non-VoIP telephone calls. To make a call the caller sends an INVITE message, which is replied to by an OK message from the server, and the server then contacts the recipient of the call. However, once the call is answered the two parties communicate directly. The caller or the recipient must then tell the server that the call has ended so that the server can calculate billing, and this is done with the BYE message. If the recipient is already using the phone a BUSY message is sent to the caller.

Four parts of the call setup and tear-down process were identified as vulnerable to man-in-the-middle attacks: replaying the INVITE message, fake BUSY responses, delayed BYE messages, and dropping BYE messages. Replaying the INVITE message will allow an attacker to create a second

call with the cost being charged to the original caller, assuming the attacker can alter the destination IP of the call. The fake BUSY message attack involves two man-in-the-middle attacks, allowing the two attackers to communicate at the expense of the caller. The likelihood of this attack is low as the two attackers must be online and know that they wish to communicate, so they could simply connect directly.

Xinyuan rounded off the presentation with some suggestions on preventing these attacks. The INVITE replay attack could be prevented with a nonce; this is done by some providers, but others allow replay after a week. To prevent the fake BUSY attack the messages need integrity protection, so that they cannot be altered by an attacker. The group is going to continue research in this area, including using these attacks to consume resources on the server of the service provider.

SNIFFING AND SCANNING

- ***BlueSniff: Eve Meets Alice and Bluetooth***

  Dominic Spill and Andrea Bittau, University College London

Dominic presented his work to produce an affordable Bluetooth sniffing device using easily available hardware. This combines existing and new techniques for breaking Bluetooth encryption and works with the GNU Radio software signal processing framework. The resulting device is affordable (approximately US$700 vs. commercial products at $10,000).

Dominic began his talk by summarizing the Bluetooth protocol, in which master and slave devices build associations based on unique MAC addresses and internal clocks. Bluetooth whitens (scrambles) packets by XORing pseudo-random values with the data stream based on the lower six bits of the internal clock, as well as using frequency hopping spread spectrum (FHSS) at 1600 hops/second chosen using the MAC address and master device's clock, in order to minimize interference.

BlueSniff must determine the MAC address and clock value in order to determine the hopping pattern and whitening sequence. Only a portion of the MAC appears in the packet, although frequently the remainder may be guessed, or relatively easily brute-forced by using the CRC to test candidate values in a single packet.

The primary limitation of this work is that more than one physical radio must be used to track all available Bluetooth channels; the current hardware can monitor only one-eighth of the frequency space.

Tal Garfinkel asked what support plans exist for the tool in the future. Dominic answered that it depended on future deployment, but if time was available, allowing eight devices to be used at once is an important next step. Niels Provos asked about the research impact of this work. Dominic answered that the lack of a promiscuous mode for Bluetooth meant that almost all published work on the Bluetooth protocol was purely theoretical and that with easily accessible sniffing the doors would be opened for a great deal more research. Dominic indicated that the current range limit was several meters. In answer to a question about Bluetooth 2.1, Dominic stated that the modulation technique changes, and current GNU modulators are not yet able to handle the new technique.

- ***Toward Undetected Operating System Fingerprinting***

  Lloyd G. Greenwald and Tavaris J. Thomas, LGS Bell Labs Innovations

Lloyd presented an in-depth analysis of operating system fingerprinting techniques utilizing the differences in TCP/IP implementations. Although this is a commonly known technique, Lloyd presented optimizations drawn from the entropy of information provided by the tests. These are useful for identifying the correct tests to run to pin down the exact version of an operating system with the minimum number of packets sent and received, thus reducing the chances of the fingerprinted system detecting the traffic.

The tool used to perform the fingerprinting was based on nmap (http://www.insecure.org), specifically the second-generation operating system fingerprinting system. The database of operating system characteristics was used as provided with the tool and was not modified for the purposes of calculating information entropy.

The result of the analysis was that it was possible to use fewer than the 16 standard packets to identify some operating systems, especially with the knowledge that the majority of systems are Windows-based, and therefore the test that reveals the most about the Windows TCP/IP stack should be used first.

The work makes it much easier to order the tests for operating system fingerprinting to avoid the more detectable tests and reduce the number of packets sent to and received from the system.

- ***Catch Me, If You Can: Evading Network Signatures with Web-based Polymorphic Worms***

  Matt Van Gundy, University of California, Davis; Davide Balzarotti and Giovanni Vigna, University of California, Santa Barbara

Matt presented his work on Web-based polymorphic worms with a focus on those written in scripting languages such as PHP. He identified the key parts of the PHP language that allow the text of a script to change but the overall execution to remain the same, and he showed how this can be used to fool applications that attempt to find software signatures.

He began with an introduction to the two most common applications for calculating software signatures, Polygraph and Hamsa, and an explanation of the techniques they employ. Matt also explained some features of the PHP script-

ing language, such as the ability to store function names in variables and insert random strings, that are ignored by the interpreter. These allowed him to alter a script while keeping the execution constant. PHP also allows for inline use of compression, which was useful for changing the signature of the script, but it reduced the size and therefore the amount that it could be changed.

He then showed how frequently his morphed script was detected by both Hamsa and Polygraph, given that both had been shown the original script and had created a signature for it. For Hamsa, the morphed worm went undetected almost 100% of the time. Polygraph was able to detect the worm more easily, and it had much lower false negative rates, but it was unable to handle the worm at its full size and could only create a signature for a compressed version of the worm.

Matt was asked if this technique could be applied to other Web-based exploits. He said that it could be used to cloak attacks such as remote file injection. When asked how the use of compression affected the results, he said he was confident that the morphed script could beat the Polygraph detection if it allowed the larger worm to be tested.

HIDDEN ATTACKS

- ***An Encrypted Exploit Payload Protocol and Target-Side Scripting Engine***

  Dino A. Dai Zovi, Two Sigma Investments

Dino presented his work on exploit payloads for mobile clients, introducing a three-stage payload that creates a connection back to a server, downloads an execution environment, and then exploits the host system.

A widely accepted network security model is that there exists a boundary between internal and external hosts, but this is often an oversimplification. Mobile clients may be connected to many different networks, each of which could have different levels of security and allow the systems to be exploited. Dino has produced a payload for these systems that allows them to exploit a network once the client returns.

The payload initially establishes a secure connection to a server in order to download the rest of the payload. This first stage needs to be small: in this implementation it was approximately 1200 bytes. The second stage uses this secure connection to download an execution environment for the Lua scripting language to run the main attack.

The final stage is the attack within the network. It is written in Lua to allow portability, and Lua environments often simply wrap system calls, giving the script a large amount of execution scope. The Lua scripting environment allows the payload access to many common protocols such as HTTP or FTP, allowing the malicious code to spread itself further through the internal "protected" network.

There were some suggestions as to how the payload could be further streamlined by only retrieving a Lua bytecode interpreter rather than the entire execution environment, and there was a discussion on using this proof-of-concept payload with real-world attacks.

- ***Exploiting Redundancy in Natural Language to Penetrate Bayesian Spam Filters***

  Christoph Karlberger, Günther Bayler, Christopher Kruegel, and Engin Kirda, Secure Systems Lab, Technical University Vienna

Christoph presented one of the first papers designed to assist spam producers. He was attempting to pass Bayesian spam filters by reducing the spam score of a message using word lists to replace key words. The resulting messages were run through common spam filtering software to assess the effectiveness of the word replacement.

The most common method for detection of unsolicited email currently is through the use of Bayesian filters. These give each piece of email a score based on factors such as content and sender identification. The filters rate the spam value of each word, and this rating is used to determine whether a message should be filtered.

The principal idea of the work was to use the rating of words to choose alternatives to high-scoring spam words. Using the word lists from WordNet (wordnet.princeton.edu), the high-scoring spam words were replaced; this took some analysis of the grammar to allow for words with multiple meanings.

The resulting email messages were run through spamassassin and dspam filters, giving significantly lower scores for messages that had passed through the word replacement software. The results showed that although this technique helps the messages to achieve a lower score, the inclusion of URLs in most spam messages will still allow the filters to reject the message.

FIVE MINUTE MADNESS

The workshop concluded with a session entitled "Five Minute Madness," an opportunity for brief descriptions of work in progress and suggestions for potential research and ideas based on the works presented. The talks were mostly based on potential attacks using botnets.

First up was a suggestion that blind SQL injection attacks could be made less detectable by using the many nodes of a botnet to reveal table information. The principle of this was to use the distributed power of a botnet to increase the size of the server logs and make tracing such attacks much more difficult; it would also make automated detection more difficult, as no single host would be extracting an entire database entry.

This was followed by an anti-spam-filter technique derived from ASCII art. The idea was to convert the intended mes-

sage to an image and then represent this using ASCII art. The ASCII art could be based on real words that are not considered to be associated with spam and therefore bypass most common spam filters.

Next came two suggestions for finding better uses for botnets, particularly attacks that we have not seen. These ranged from a distributed attempt to find private keys of large organizations, to capturing audio and video on home or office systems for blackmail or fraud purposes. A brief debate followed about physical security between a user and his or her own system, with regard to protecting the user from the system.

The session was rounded off by a suggestion that DNS traffic should be monitored, because the first lookup request for a domain, for example a botnet distribution server or a phishing site, will come from the person who set it up testing their work. This was disputed by some, suggesting that most botnet controllers would route the traffic through the botnet or Tor network to hide their tracks.