JAN GÖBEL, JENS HEKTOR, AND THORSTEN HOLZ

advanced honeypot-based intrusion detection

Jan Göbel has an M.Sc.in computer science from RWTH Aachen University and wrote his diploma thesis on "Advanced Honeynet-based Intrusion Detection." He is currently working at the Center for Computing and Communication at RWTH Aachen.

goebel@rz.rwth-aachen.de

Jens Hektor holds an M.Sc. degree in physics from RWTH Aachen University and afterwards he joined the Center for Computing and Communication there. He is responsible for the network infrastructure of the university and developed the first version of Blast-o-Mat.

hektor@rz.rwth-aachen.de

Thorsten Holz holds an M.Sc. degree in computer science from RWTH Aachen University and is currently a Ph.D. student at the University of Mannheim. His research focuses on honeypots and honeynets and currently one main area of work is botnets.

thorsten.holz@informatik.uni-mannheim.de

AT RWTH AACHEN UNIVERSITY, WITH about 40,000 computer-using people to support, we have built a system to detect infected machines based on honeypots. One important building block of *Blast-o-Mat* is Nepenthes, which we use both to detect malware-infected systems and to collect malware. Nepenthes is a low-interaction honeypot that appears as vulnerable software but instead decodes attack code and downloads malware. We have been successful at uncovering and quarantining infected systems with sensors listening at 0.1% of our address space. Investigation of collected malware has led to discovery of many infected systems and even a huge cache of stolen identity information.

The Internet has evolved into a platform for all kinds of security-sensitive services and applications. Online banking and payment have become part of today's way of life. For this reason, even home computers store valuable information such as passwords to online shops, credit card numbers, account data, and personal identification numbers. Therefore, securing network hosts, learning attack methods, capturing of attack tools, and studying motives of computer criminals are important tasks for network administrators and security engineers.

One important aspect of network attacks is malicious software (*malware*) that spreads autonomously over the network by exploiting known or unknown vulnerabilities. In the form of *network worms* or *bots/botnets*—networks of compromised machines that can be remotely controlled by an attacker—malware poses a severe threat to today's Internet. For example, botnets cause damage from Distributed Denial-of-Service (DDoS) attacks, sending of spam, identity theft, or similar malicious activities.

Within the university network, we want to detect infected hosts as fast as possible. Only if we detect a compromised machine, can we contain it and stop the spreading mechanism. This cessation protects other vulnerable hosts within the university network and also within external networks. Instead of using a classical Intrusion Detection System (IDS), we have built our own solution called *Blast-o-Mat*. This system aims at automatic notification and handling of malware-infected

hosts. The main task of Blast-o-Mat is to determine the responsible person of a system for which it receives an alert, send out a warning to the owner, and, in case an infected host is still active after a certain period of time, block network access to and from this host. It is automatically transferred to a *quarantine network* (i.e., all access to the Internet is rerouted to a certain server). Basically the infected machine can then only access certain sites to download patches and in the future also antivirus software. Within the quarantine network, we can also monitor what happens to the machine from a network point of view. As a result, we have a tool that automatically performs the time-consuming tasks that normally the network administrator has to carry out.

The system consists of several modules that try to detect an infected system:

- Blast-Sniffer continuously reads traffic data from a SPAN or mirror port of a central router of the network and writes it to a MySQL database. This database serves as the input for the next two intrusion detection sensors.
- Blast-PortScan detects hosts that are scanning a large number of IP addresses for certain ports, which could indicate a malware-contaminated machine. To accomplish this task, the module counts the number of TCP SYN packets sent by each host during a preconfigured period of time. Within our environment, a threshold of 50 SYN packets within four minutes has proven to be a reasonable indicator that tends not to generate false positives and is capable of detecting infected hosts efficiently.
- Blast-SpamDet aims at detecting machines that send spam messages. Similar to the portscan detector, it counts the number of initiated connections from a suspicious host, but this time only connections to mailservers are considered. When a certain number of connections is reached, the server entity starts to gather email header information of the suspected host with the help of packet-capture tools. All used sender addresses are filtered and counted. If the number of unique sender addresses exceeds a certain threshold, further actions are initiated.
- Nepenthes is a low-interaction honeypot solution that is capable of automatically downloading malware. We describe its inner workings in a separate section. We use this module to get in-depth information about ongoing network attacks and the analysis of the downloaded binary can help us to further examine the incident.

In the following, we give a brief background of honeypots and then introduce Nepenthes in detail. We show how this low-interaction honeypot can be used to detect infected machines and explain possible ways to isolate such compromised hosts. Afterwards, we highlight some incidents detected within the past couple of months.

Background on Honeypots and Honeynets

A *honeypot* is "an information system resource whose value lies in unauthorized or illicit use of that resource" [1]. This methodology has been used to study attackers and types of attacks in depth, providing valuable information about tools, tactics, and motives of attackers. To learn more about malware, we use low-interaction honeypots such as *Nepenthes* [2] and *high-interaction honeypots* such as GenIII honeynets [3].

Low-interaction honeypots emulate services or operating systems. They allow an attacker a limited interaction with the target system and allow us

to learn mainly quantitative information about attacks. Since low-interaction honeypots use simulation, they construct a controlled environment and thus the risk involved is limited. We give a more detailed introduction to low-interaction honeypots in the following section.

In contrast, high-interaction honeypots do not emulate any services, functionality, or operating systems. Instead, they provides real systems and services, allowing us to capture extensive information on threats. Several honeypots can be combined into a network, called a *honeynet*. We can capture the exploits of attackers as they gain unauthorized access, monitor their keystrokes, recover their tools, or learn what their motives are. The disadvantage to high-interaction solutions is that they have increased risk: Because the attackers can potentially fully access the operating system, they can potentially use it to harm other nonhoneypot systems.

A honeynet creates a fishbowl environment that allows attackers to interact with the system, while giving the operator the ability to capture all of their activity. This fishbowl also controls the attacker's actions, mitigating the risk of them doing harm to any nonhoneypot systems. The key element to a honeynet deployment is called the *Honeywall*, a layer-two bridging device that separates the honeynet from the rest of the network. This device mitigates risk through data control and captures data for analysis. Tools on the Honeywall allow for analysis of an attacker's activities. Any inbound or outbound traffic to the honeypots must pass through the Honeywall. Information is captured using a variety of methods, including passive network sniffers, IDS alerts, firewall logs, and the kernel module known as "Sebek" [4]. The attacker's activities are controlled at the network level, with all outbound connections filtered through both an intrusion prevention system and a connection limiter.

Neither of these two approaches is superior to the other; each has unique advantages and disadvantages.

Collecting Malware with Nepenthes

The low-interaction honeypot Nepenthes aims at capturing malicious software such as networks worms or bots that spread in an automated manner. The main focus of this application is to obtain the malware itself (i.e., to download and store the malware binary for further in-depth analysis). Unlike other low-interaction honeypots, Nepenthes does not emulate full services for an attacker to interact with. The key idea is to offer only as much interaction as is needed to exploit a vulnerability. For this reason, Nepenthes is not designed for any human interaction, as the trap would be easily detected. On the contrary, for the automated attack, just a few general conditions have to be fulfilled, thus maximizing the effectiveness of this approach. These conditions usually include displaying the correct banner information of an emulated service and sending back specific information at certain offsets during the exploitation attempts. Therefore, the resulting service is only partially implemented. This allows deployment of several thousands of virtual honeypots with only moderate requirements in hardware and maintenance.

Nepenthes is designed as a single-threaded core daemon, with a number of different modules, facilitating each task of the malware collection process:

- *Vulnerability modules* emulate the vulnerable parts of network services.
- *Shellcode parsing modules* analyze the payload received by one of the vulnerability modules. These modules analyze the received shellcode,

an assembly language program, and extract information about the propagating malware.

- *Fetch modules* use the information extracted by the shellcode parsing modules to download the malware from a remote location.
- *ubmission modules* take care of the downloaded malware (e.g., by saving the binary to a hard disk, storing it in a database, or sending it to antivirus vendors.)
- *Logging modules* log information about the emulation process and help get an overview of patterns in the collected data.

Besides the modular structure, Nepenthes provides an event-driven notification mechanism. Each step of an attack triggers certain events, which other modules can register to and therefore react on. As a result, Nepenthes can be highly customized to fit into new environments.

We briefly describe the three most important kinds of modules in more detail in order to give a better understanding of the operation of Nepenthes.

Vulnerability modules are the main reason for the efficiency of the Nepenthes platform. The main idea—only emulating the vulnerable parts of a service—has already been explained. We only need to emulate the relevant parts and are thus able to efficiently implement this emulation. Eventually, we receive the actual payload, which is then passed to the next type of module.

Shellcode parsing modules analyze the received payload and automatically extract relevant information about the exploitation attempt. Currently, only one shellcode parsing module is capable of analyzing all shellcodes received in the wild. The module works in the following way: First, it tries to decode the shellcode. Most shellcode is obfuscated with an XOR encoder. An XOR decoder is a common way to "encrypt" the native shellcode in order to evade intrusion detection systems and string-processing functions. After decoding the code itself according to the computed key, this module then extracts more information from the shellcode (e.g., credentials). If enough information can be reconstructed to download the malware from a remote location this information is passed to the next type of module.

Fetch modules have the task of downloading files from remote locations. Currently, there are several different fetch modules. The standard protocols TFTP, HTTP, and FTP are supported. Since some bots use custom protocols for propagation, there are also fetch modules to handle these bot-specific protocols.

The modularity and flexibility of Nepenthes allows for the deployment of unique features not available in high-interaction honeypots. For example, it is possible to emulate the vulnerabilities of different operating systems and computer architectures on a single machine during a single attack (e.g., an emulation can mimic the generic parts of a network conversation and depending on the network traffic decide whether it needs to be a Linux or a Win32 machine).

The source code of Nepenthes is available under GPL at http://nepenthes.mwcollect.org. In addition, a more detailed introduction, together with preliminary results, is also available [2].

NEPENTHES AS PART OF AN IDS

Within the Blast-o-Mat architecture, Nepenthes serves as a sensor to detect infected machines. These machines typically try to propagate further by scanning for vulnerable machines. Thus we have placed Nepenthes sensors all over the network and on each of these IP addresses they emulate common vulnerabilities as already explained. We use about 180 IP addresses to cover three /16 networks, thus covering about 0.1% of all addresses. Nevertheless, preliminary results show the effectiveness of this approach. One important finding is that Nepenthes has not generated any false positives: Whenever Nepenthes signals a successful exploitation attempt, it is not a portscan or misconfigured system, but a real intrusion attempt. To this point, the Blast-o-Mat system has already detected hundreds of infected machines and the automatic containment works without problems.

MITIGATION OF INFECTED SYSTEMS

As soon as an infected system has been detected, the first question entails how to deal with it. Presumably the best way is to immediately take the system offline, giving it as little chance as possible to infect other systems in the network. The inhibition can take place on any of the OSI layers, depending on the given infrastructure. If direct access to the switch port of the conspicuous machine is given, one can disable this port. In this case the host is locked at the physical layer of the OSI model. An inhibition on layer 2 is equal to blocking the MAC address of the hostile host. This approach would also prevent the system from being taken online again on a different switch port. The disadvantage of these two methods is the effort it takes to determine the correct network device to which the contaminated machine is connected. Less costly is the locking of the IP address with the help of access lists (OSI layer 3). In this case, we need to determine the router, which routes the appropriate network. Although the host is properly blocked, it can still infect systems within the same local area network (LAN). On higher layers of the OSI model, it is possible to lock certain TCP or UDP ports or operate different protocol-specific filters to isolate an infected host. However, all modifications to network components have to be reverted, as soon as the problem is solved and the user wants to get back online.

A different approach of taking a contaminated host offline is to place it into a *quarantine network*, isolating it from other systems. Although this requires a certain infrastructure, this is the most effective solution, as additional information can be collected from the quarantined host. Currently, we have implemented a simple form of such a quarantine network: The Blast-o-Mat is capable of redirecting HTTP traffic of infected machines to a special Web server. Before taking a look at the practical implementation of this approach, we introduce two ways to actually block the infected host.

When blocking we differentiate between two different groups of users: static IPs (normally staff people or PC pools) and dynamic IPs (typically WLAN users).

To identify the responsible person(s) for hosts with static IP addresses, we maintain an XML-based database with all relevant information: For each subnet the database contains the registered administrators, their phone number and email address, the net mask, the acronym of the institute, and, if available, the assigned Virtual Local Area Network (VLAN) number. Additionally, for each entry there exists information about the manageable network router, through which the associated subnet is routed. To lock a

host with a static IP address, use is made of a Perl script capable of automatically creating antispoofing access lists. These access lists can be extended with firewall rules or, in our case, with lists of locked machines, thus efficiently blocking contaminated hosts from accessing the network.

To identify the responsible person for a dynamically assigned IP address, we have to ascertain the account name from the authentication or accounting server. Therefore, we have to compare the IP address and the time of the incident with the information stored in the Radius server. We run a slightly modified version of the FreeRadius software, which writes its accounting data to a MySQL database, on a daily basis. Thus, we have a database table for each day, which greatly accelerates the process of searching for specific accounting data. The account locking of an infected host is accomplished by setting a special flag in the LDAP database, which is used for user authentication. Once the flag is set, a user with an infected machine can no longer connect to the campus network and has to contact the helpdesk to be unlocked again.

One of the more complicated tasks in automatic locking of infected systems is to notify the user of the suspected host. Our main method is to notify any responsible person via email. Since every student at RWTH Aachen gets his or her own email address upon enrollment, we have a fairly good possibility of reaching any student. The obvious limitation is that we cannot assure that the students read their university email frequently or even at all. Therefore the Blast-o-Mat is capable of redirecting certain traffic to a specially designed Web server as briefly mentioned above. Because of the network structure at RWTH Aachen, the redirection currently works only for the wireless network, but we hope to extend this in the future. All traffic of wireless hosts has to pass one central gateway. Thus, we are able to efficiently redirect any traffic of hostile hosts at this point, via the use of certain IPTables rules. The main advantage of this approach is that the responsible person of a redirected host is efficiently informed, even if the warning mails of the Blast-o-Mat are not read. Every attempt to open a Web site on a redirected host displays the information site of the quarantine Web server, showing all gathered data about the incident so far. Furthermore, email delivery is still possible, allowing the user to get additional information, provided with the Blast-o-Mat warning messages.

To achieve the redirection of a contaminated host, the Blast-o-Mat remotely executes a Python script on the gateway server and transmits the account name, the IP address, and the time the system was online as parameters. The easiest way would be to do the redirection based on the IP address of the infected host. But since we have to deal with dynamically assigned addresses, this would not prevent the user from logging in again with a different IP address, thus circumventing the redirection measures. Therefore, we have to determine the MAC address of the offending machine. This is accomplished with the help of an additional script, which queries the DHCP server with the time the host was online and its IP address as parameters. Every DHCP server maintains a lease file, containing all MAC addresses of hosts to which it assigned an IP address, together with the time interval the given IP address is valid. With the help of this file, we are able to determine the MAC address of the system that was online with a certain IP during a given time. As a result, the script generates an IPTables rule that redirects any further HTTP traffic of the specified MAC address to the quarantine Web server.

A more advanced solution to build a quarantine network would involve VLANs: As soon as a host is detected by the Blast-o-Mat, the VLAN tag for this machine is changed to the tag of the quarantine network (which could

be a honeynet). As a result, all traffic is redirected. The major drawback of this concept is that it requires the network infrastructure to allow access to the switch port of each host and additionally the switch must support VLAN tagging of certain ports.

In the next section, we take a closer look at one particular alert generated by Nepenthes.

A Modern Trojan: Haxdoor

During one security incident detected by Nepenthes in April 2006 we noticed a strange behavior of the infected machine: It constantly tried to post data to a certain PHP file located at a server in the United States. Since the machine had already been moved into the quarantine network, we could further observe it. We noticed that sensitive data—in this case passwords—were sent to the remote server. A closer examination revealed the URL from the HTTP requests and we quickly noticed that these requests were caused by a variant of *Haxdoor*, one of the most advanced Trojans in the wild.

In addition to the normal Trojan capabilities, such as copying itself to the Windows Installation Directory or start on reboot, Haxdoor also implements rootkit capabilities and advanced identity theft mechanisms. It can, for example, hide its presence on the compromised machine via SSDT (System Service Dispatch Table) hooking as well as steal all information entered into Internet Explorer. All of this captured information can be sent to a central server, which is precisely the activity we observed within the quarantine network.

During further investigation, we found several log files that contained all information stolen from all infected machines. In total, these log files contained more than 6.6 million entries, amounting to 285 MB of data. This data was stolen from the compromised machines between April 19 and April 27, 2006, so within only nine days. In total, we found evidence of more than 39,000 different IP addresses that were victim of this particular Haxdoor infection. These numbers show the effectiveness of this kind of attack.

The log files contained full detailed information about more than 280 bank accounts and several credit card numbers. All major German banks were victim of this incident and several other large brands from the e-commerce sector were also targeted. In addition, the attacker also collected sensitive information such as username and password combinations or other data entered into HTML forms from the victim's computers. More information about this kind of modern data theft can be found in another article in this issue of *;login:*[5].

We handed this information over to DFN-CERT, the Computer Emergency Response Team responsible for German research and education networks. The affected users were warned in cooperation with universities, ISPs, and other affected sites.

BINARY ANALYSIS

Sandboxing is a well-established approach that involves executing the malware in an emulated environment and monitoring its behavior. During preparation of a diploma thesis at our lab, Carsten Willems developed a sandbox named *CWSandbox* [6]. Preliminary results show that CWSandbox is able to efficiently and accurately analyze a given malware binary. The tool is able to extract all important information from a given

binary in an automated way within a short amount of time (usually three minutes). The extracted information includes information about changes to the filesystem or the Windows registry, process access, DLL handling, and network communication. The whole analysis process does not require any human interaction and can be parallelized, allowing for concurrent analysis of large amounts of data.

The information in the following paragraphs is based on the reports generated by CWSandbox, enriched with information retrieved via manual binary analysis. In total, we analyzed eight different variants of Haxdoor. All of them share many characteristics and the following description is a generalization of the different Haxdoor variants.

Typically, this specimen of malware creates several different files in the Windows installation folder. By default, this is either C:\Windows (Windows 2000 and XP) or C:\Winnt (Windows NT). The created files include normally two Dynamic Linked Libraries (DLL), three to four drivers (SYS), and several additional configuration files. For example, the variant Haxdoor.IN (according to Bitdefender) creates the following files: sndu32.dll and qm.dll (same as sndu32.dll), sndu64.sys and qm.sys, and stt82.ini, klgcptini.dat, and stt82.ini.

Upon executing, the binary loads several DLLs. These include the typical Windows DLLs such as kernel32 or ntdll but also network-related DLLs such as wsock32 and the code within the newly created files. One characteristic sign of Haxdoor is the creation of a mutex with the name RasPbFile.

Haxdoor also interacts with the Windows registry to enable a mechanism to be started upon reboot. In contrast to other malware, which commonly adds a registry key under Run or RunService, Haxdoor is more advanced. It uses a mechanism to auto-load via Winlogon or even during SafeBoot. The corresponding registry keys are HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify and HKLM\SYSTEM\CurrentControlSet\Control \SafeBootMinimal and \SafeBootNetwork, respectively.

Via the Windows Service Control Manager (SCM), Haxdoor also adds a service to the infected system that is automatically started upon system startup. The name of the service varies; it can, for example, be "SoundDriver SDB64" or "UDP32 netbios mapping," depending on the variant. In addition, it creates a remote thread within the memory space of Explorer.exe, in order to add some further services. Moreover, Haxdoor has some advanced tricks to hide its presence on the infected system and it is hard to get rid of it. This topic is, however, beyond the scope of this article and more information can be found on the Web sites of different antivirus vendors.

Conclusions and Future Research

The Blast-o-Mat IDS has been running for about seven month now, efficiently handling malware-infected hosts within the campus network of RWTH Aachen. With the help of the honeypot Nepenthes and the additional intrusion sensors, so far, a total of 361 incidents were detected. A little more than one-third were reported by Nepenthes, with the rest split up between the Blast-PortScan and Blast-SpamDet sensors. The PortScan sensor reported the most incidents, owing to its much larger number of monitored ports than vulnerability modules. However, each portscan that was

detected on a port for which a vulnerability module exists was detected by Nepenthes as well.

Although its missing vulnerability modules means that Nepenthes does not recognize exploit attempts on all ports, it has proven to be a great intrusion detection mechanism. The biggest advantage is its accuracy, as no false positives are reported, as well as the high detection ratio, with only a few IP addresses assigned. Currently, we are monitoring with Nepenthes less than 0.1% of the complete IP space and already achieve almost the same results as the Blast-PortScan sensor, which receives its data from a SPAN port of a centralized router.

Because the bandwidth of current large-scale networks such as the one of RWTH Aachen already exceeds 1 gigabit of traffic and can approach 10 gigabits, common SPAN port monitoring will no longer work without the use of specialized and expensive hardware. However, Nepenthes will still deliver the same quantitative results with just 180 IP addresses. Therefore, it serves as a future-proof intrusion detection sensor, capable of running on a normal off-the-shelf computer.

In addition to the detection of contaminated hosts, Nepenthes also captures the malware that is trying to exploit the emulated vulnerabilities. Thus, we are able to submit the collected binaries for further analysis to different applications, such as virus scanners, to determine the kind of malware, or to the CWSandbox, to find out more about the behavior of malicious software. As a result, we are able to supply a qualitative high-class report for the detected incidents, both to help clean infected machines and to raise the user's security awareness.

ACKNOWLEDGMENTS

We would like to thank Sam Stover for reading a previous version of this paper and giving valuable feedback that substantially improved its presentation. In addition, we would like to thank the Nepenthes Development Team for implementing such a wonderful tool.

REFERENCES

[1] The Honeynet Project. Know Your Enemy: Honeynets (May 2005): http://www.honeynet.org/papers/honeynet/.

[2] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The Nepenthes Platform: An Efficient Approach to Collect Malware," *9th International Symposium on Recent Advances in Intrusion Detection, RAID06, Hamburg, Germany, September 20-22, 2006, Proceedings, Lecture Notes in Computer Science* (Springer, 2006).

[3]E. Balas and C. Viecco, "Towards a Third Generation Data Capture Architecture for Honeynets," *Proceedings of the 6th IEEE Information Assurance Workshop, West Point* (IEEE, 2005).

[4] The Honeynet Project. Know Your Enemy: Sebek (November 2003): http://www.honeynet.org/papers/sebek.pdf.

[5] Team Cymru, "The Underground Economy: Priceless," *;login:*, this issue.

[6] CWSandbox—behavior-based binary analysis: http://www.cwsandbox.org/.

See also T. Holz, "Spying with Bots," *;login:*, 30 (6) (Berkeley, CA, 2005): 18–23.