Proceedings of the 2004 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 10–11 June 2004

An Investigation of a Compromised Host on a Honeynet Being Used to Increase the Security of a Large Enterprise Network

Timothy R. Jackson, John G. Levine, Julian B. Grizzard, Henry L. Owen
whitehat@resnet.gatech.edu, levine@ece.gatech.edu, grizzard@ece.gatech.edu,
henry.owen@ece.gatech.edu
Georgia Institute of Technology; Atlanta, Georgia 30332-0250

Abstract—
The growth of network intrusions on large enterprise networks continues to increase, creating an epidemic of compromised hosts. The deployment of firewalls and intrusion detection systems has not slowed the growth of intrusions to an acceptable rate. Investigating the compromise of a production machine is both difficult and time-consuming due to the mixing of attack and production traffic, while similar investigations of compromised machines on honeynets are much less complex since there is no real production traffic. We discuss why these investigations are easier on a honeynet and how honeynets may be used to make investigations of compromised production machines faster and recovery easier. We include a description of an attack and the analysis that was conducted.

I. Introduction

Computer networks today are vulnerable to a variety of attacks, with recent worms such as Blaster, Nachi, and MyDoom demonstrating the ability of malicious software to spread quickly through these networks. There are an estimated 80,000 viruses in existence today, a rapidly growing number[1]. Thousands of hackers probe and attack computer networks each day. These attacks range from relatively benign ping sweeps (where each computer on a network is asked to respond to a special message) to sophisticated techniques exploiting security vulnerabilities in both software and hardware. Further complicating matters for the network administrators charged with defending these networks, a "relatively benign" ping sweep may be preparation for a more sophisticated attack[2]. As the amount of traffic on a network grows, monitoring that traffic and isolating threats from normal or otherwise harmless activity becomes increasingly difficult. This situation is even worse for large universities where the need for academic freedom prevents network administrators from implementing strict controls on their networks[3].

II. Background

To help prevent intrusions, network administrators may deploy a variety of defenses, including firewalls and Intrusion Detection Systems. The functions of these defenses vary, but they all have the same objective: increase the security of the network they defend. A recent work proposed using honeynets to secure large enterprise networks, such as the network at Georgia Tech[3]. In this paper we discuss specific examples of how data recovered from honeynets may be used to increase the security of such networks.

A. Firewalls

One defense network administrators may employ is a firewall, which acts as sieves on traffic passing through it. The objective is to prevent malicious traffic from reaching the rest of the network. Very often firewalls are deployed at the network edges, where they connect to the Internet. The objective then is to prevent malicious code and hackers on the Internet from ever reaching the protected network[4].

B. Limitations of Firewalls

Unfortunately for the System Administrator, there are numerous and ever more ingenious techniques for circumventing firewalls[5]. For instance, firewalls do not protect against the transfer of an email carrying a virus infected attachment, or they might be fooled by packets that have been intentionally fragmented, in ways designed to bypass firewalls. Some networks (i.e. those at large public universities) cannot have extremely protective firewalls because of their need to maintain academic openness. Networks with extremely large amounts of traffic, such as those at major corporations and universities, may be impractical to firewall due to the shear volume of traffic.

C. Intrusion Detection Systems

Since firewalls alone are not a complete solution, network administrators also often install an Intrusion Detection System (IDS). An IDS is similar to a burglar alarm in that it does not seek to prevent an attack but warns a network administrator if it detects an attack[6].

D. Limitations of an IDS

While this warning system is useful and an extremely powerful defensive tool, especially combined with a fire-

wall, it too has shortcomings. Neither a firewall or an IDS can defend against an attack that bypasses the defense, such as unauthorized modems or even encrypted tunnels passing through them. Depending on its placement within a network, a firewall or IDS may not protect a network from internal attacks[3]. Yet, the largest shortcoming inherent to both firewalls and Intrusion Detection Systems is their reliance on signatures of known attacks. The rules and filters used by both firewalls and intrusion detection systems are based on information gathered from previous attacks. Any attack sufficiently different from those the rules are based on will likely bypass both a firewall and an IDS without raising an alarm (false negatives)[6]. Alternatively, an IDS may report an attack when there is not one (false positive).

E. Responding to Compromises

Since networks connected to the Internet cannot be defended against all possible avenues of attack, it seems reasonable to assume any vulnerable computer connected to the Internet has a high probability of being compromised. If a production machine is compromised, it can take a long time to identify the method of compromise. If valuable information such as credit card numbers, customer records, or trade secrets might have been exposed, the recovery process may take even longer as verifying that the machine is clean is not a simple matter. Ensuring that no rootkit or trojan (a collection of programs designed to give a hacker access to a machine without needing an authorized account and without the System Administrator's knowledge) has been left on the system is a time-consuming and uncertain process. Replacement risks missing some or all of the rootkit, while re-installing the OS does not provide any clues as to the extent of the compromise; especially about what other machines may have been compromised[7]. We believe honeynets may be used to aid the investigation of and response to a network intrusion.

F. Definition of a Honeynet

A honeynet is a collection of honeypots. A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.[8]. The honeypots are placed behind a firewall in such a manner as to allow all traffic to and from the honeynet to be logged (data capture). The firewall limits the number of outbound connections to keep honeypots from attacking other machines (data control). This configuration ensures that all traffic to the honeynet is contained, captured, and controlled. Typically, hosts similar to those on the rest of the network are used on the honeynet to make the honeynet indistinguishable from the rest of the network.

III. Investigations on a Honeynet

When a honeypot is compromised, the nature of the honeynet facilitates the investigation, a process that is often difficult on a production machine[9]. The reasons fall into four main categories: suspicious traffic, logging, non-production machines, and confirmation of cleaning. We propose the use of honeynets to bring these advantages to production machines as well. The potential benefits of this process include: greatly speeding the investigation of compromised hosts, helping establish signatures for possible future attacks, enabling the development of cleaning procedures for compromised hosts, allowing easier cleaning of hosts compromised in the same way, and better assurance that formerly compromised hosts are now clean.

A. Suspicious Traffic

Since a honeynet is not made up of actual production machines, there should theoretically be no traffic to or from the honeynet. All traffic in the logs is suspicious and often much less in volume than similar logs from production machines. This makes finding important data in the logs much easier. It also means that any host on the network being monitored that attempts to connect to the honeynet is likely compromised. By identifying these hosts, the System Administrator can increase the security of the network.

B. Logs

Every packet to and from the honeynet is logged, giving the system administrator valuable information about an attack that is normally targeted against a production machine. This information includes: the exploit used, commands run on the system, and files the attacker downloaded onto the host, and may be recovered even if the actual attack traffic was encrypted. Having these files allows a System Administrator to examine them and even install them on other machines for further analysis.

From the logs, one may learn about who or what the attacker was, which exploit was used, what the compromised host did in response, and what if anything the attacker did after compromising the machine. Once the offending IP address has been identified, it is possible to search the honeynet logs for all instances of that IP. This may help determine if this incident is part of a series of attacks or just an isolated attack of opportunity. System Administrators may also wish to correlate the honeynet logs with the results of other defenses (e.g. Intrusion Detection Systems and security scanners) as is currently done at Georgia Tech to evaluate the effectiveness of these defenses. The logs may be used to ascertain, with a high degree of probability, whether the attacker is a human, a script, or a worm.

This determination is made mostly through an analysis of the attack's timing. First, one considers how many other hosts were trying this same attack. With the infection rate worms achieve, almost any worm that compromises

a honeynet will also be on other machines that will attack the rest of the honeynet immediately, while humans tend to explore a compromised machine before moving on to other hosts. If many hosts try the same attack in a short period of time, the attack is likely coming from a worm. If the attack is coming from only a small number of hosts, one may perform a packet time-stamp analysis on the attack traffic. A script will typically be able to scan all the hosts in a class C subnet in a matter of seconds. A human will take much longer. This last measurement is of course not a perfect metric, but it is a general rule that provides a starting place for more analysis.

If the machine is compromised, the logs help us even further. The behavior of worms, scripts, and humans after the initial infection is often easily differentiable. Worms will infect, possibly open a back-door, and then begin attempting to propagate. This propagation traffic will then show up in the honeynet logs. Scripts will typically infect, set up a back-door, make a note for their owner that a machine has been compromised, and move on. This leaves a distinct lack of propagation traffic in the logs. If a human is running the process manually, they usually see the compromise, then a back-door is installed (in the form of a remote shell or GUI program), and then the attacker will begin to run commands. They will install their rootkits, then typically either look around for data to take or set up some sort of server, and then clean up. While a script could do these things, a packet time-stamp analysis (easily performed using the logs) will give a good indication of whether or not a person was typing. In addition, mistakes in commands and "colorful metaphors" occasionally appear in the command sequences, indicating a human typing.

The primary benefit of this analysis to production machines is the isolation of patterns or signatures. Once an attack is found on the honeynet, a signature for that attack is known. Signatures of known attacks are essential to the building of both firewall and IDS rules. If a System Administrator can determine the signature of exploit code, it can be detected or blocked by firewalls, intrusion detection systems, or other network filters (in the case of email born viruses, the mail server may filter the virus payload). By providing reliable data on the attack, the logs also allow network administrators to better determine which alerts from their intrusion detection systems are actual attacks and which are not.

C. Non-production Machines

While production machines need to be available to their users as much as possible, honeypots have no such requirements[10]. The system administrator is free to take the honeypots offline and conduct analysis for as long as necessary. If they are available, other machines may even be put up on the honeynet to replace the compromised host. Alternatively, the compromised host may have its hard drive copied onto another machine, for distribution and analysis, while the host is left online to study any further activity by the attacker[11].

Using the logs, one is able to determine exactly how a host was infected, what software was downloaded to it, and what commands it was instructed to run. This gives us a much better understanding of any rootkits that may have been installed. In fact, one has the actual binary or source code as it was sent to the host, which may be reconstructed to build a working copy of any files that were sent. Since these are non-production machines, one does not mind too much if an attempted cleaning causes a catastrophic failure of the system and forces a reinstall of the operating system. The system administrator is free to experiment with the system to find out how to clean it. With the logs as a guide, one may even reinfect a system multiple times to perfect the cleaning process. The System Administrator is also free to experiment with the composition of the honeynet and may place machines of interest on the honeynet.

Once cleaning procedures and patterns of behaviors have been established for hosts compromised in a specific way, System Administrators are able to use them to diagnose and clean similarly compromised production machines. This is very similar to the system used by anti-virus companies. They study a virus on their own machines, and then release information on how to identify and remove that virus. The same process works for rootkits and other software and is currently the subject of much research[3].

D. Ensuring Machines are Clean

Once a honeypot is cleaned, it may be returned to the honeynet. If the system was not cleaned properly, the firewall prevents the host from doing much damage (data control) while the logs reveal any activity resulting from the continued compromise (data capture). The primary benefit to production machines relies on the principle that any host on the protected network that sends traffic to the honeynet is probably compromised. (This is a slight oversimplification as some hosts on a network, such as DNS servers, may have legitimate reasons for sending traffic to the honeynet. This traffic however is very small and easily identified by the System Administrator.) Thus, if a supposedly cleaned production machine begins sending traffic to the honeynet, particularly malicious traffic, the System Administrator is immediately aware that the production machine is either still compromised or has been re-compromised.

To further enhance this process, we speculate that it would be possible to implement a virtual honeynet using a virtual local area network (VLAN) such that hosts could be dynamically added or removed from this network. The use of such a VLAN would allow any compromised host to be instantly placed inside the virtual honeynet, thus gaining most of the advantages inherent to honeynet machines, without greatly affecting the usability of the host.

IV. Results

At the Georgia Institute of Technology, we have been using many of these techniques for over a year. The Georgia Tech network has some 33,000 hosts, and is constantly scanned using a variety of security auditing tools. Data from our honeynet helps to identify compromised hosts throughout the Georgia Tech network. In many cases we are able to identify compromised hosts before other security scans, sometimes within minutes of the attack. To demonstrate our techniques, we include an analysis of an attack in the fall of 2003[12].

On November 1, 2003, a Microsoft Windows 2000 Pro machine on the Georgia Tech honeynet was compromised by an attacker. The attack originated from a Georgia Tech student's computer. However, analysis of the data seems to indicate that this host was only a relay for the attack and not the attacker's actual machine.

The attack first appeared as a standard Nachi attack, but after an initial attempt to compromise the machine revealed to the attacker that the machine had already been infected, he or she switched tactics and used an MSBlaster style exploit to open port 4444 with Administrator privileges, thus indicating by the sophistication and timing that this was a live attacker not an automated program. He or she then began setting up a rootkit on the machine.

```
Microsoft Windows 2000 [Version 5.00.2195].
(C) Copyright 1985-1999 Microsoft Corp...

.
C:\WINNT\system32>cd setup
echo open [ftp server IP] >> fcs0.txt
echo [username] >> fcs0.txt
echo [password] >> fcs0.txt
echo get c.exe>> fcs0.txt
echo get x.exe >> fcs0.txt
echo bye >> fcs0.txt
ftp -i -s:fcs0.txt
```

From these logs, we can see the rootkit is made up of two self-extracting .exe files. This attacker names them c.exe and x.exe. By obtaining our own copy of these files from the logs, we are able to extract them ourselves and learn more about them. The former .exe extracts to a directory named "svchost" with a subdirectory "service" while the later extracts to "service" and "spools." The "svchost" directory contains WinMngr.EXE, ident.bat, one.exe, svc.bat, win.dll, cygwin1.dll, lsass.exe, regsvc.exe services.exe, and svchost.exe. These form the core of the rootkit. The subdirectory "svchost\service" is used for storage of warez, but because the attacker does not want disk usage to be noticed, he or she only places a few files on each compromised machine. The "service" directory created by x.exe contains mostly duplicate files from the "svchost" directory (possibly to avoid path issues), but it does have one important file in thug.bat.

```
C:\WINNT\system32\Setup>cd svchost
cd svchost
.
C:\WINNT\system32\Setup\svchost>
C:\WINNT\system32\Setup\svchost>dir
Volume in drive C has no label..
Volume Serial Number is 30A6-BFBE.
.
Directory of C:\WINNT\system32\Setup\svchost.
.
11/01/2003 05:57p <DIR> ..
11/01/2003 05:57p <DIR> ...
06/02/2002 11:24p 246,272 cygwin1.dll.
10/28/2003 03:01p 44 ident.bat.
11/26/2001 02:38p 4,608 lsass.exe.
10/18/2003 07:57p 48,640 one.exe.
04/06/2003 07:38p 9,665 regsvc.exe.
10/28/2003 03:00p <DIR> service.
08/29/2002 09:13p 32,256 services.exe.
10/11/2003 08:08p 591 svc.bat.
04/06/2003 07:30p 98,892 svchost.exe.
10/28/2003 03:02p 1,666 win.dll.
06/19/2002 01:22p 81,165 WinMngr.EXE.
10 File(s) 523,799 bytes.
3 Dir(s) 3,578,163,200 bytes free.
.
C:\WINNT\system32\Setup\svchost>
C:\WINNT\system32\Setup\svchost>svc <IRC username>

svc <IRC username>
S3rv1c3 RRB yay =) .
ok lemme try ......
patched and in there .
The Remote Registry Backup service is starting..
The Remote Registry Backup service was started successfully...
.
S3rv1c3 MNK yay =) .
ok lemme try ......
patched and in there .
The Microsoft Networks service is starting..
The Microsoft Networks service was started successfully...
```

Again, working from the logs it is easy to see how our attacker extracts these files and directories to "C:\WINNT\system32\Setup." The attacker then moves into the "svchost" directory and executes svc.bat. This file is the primary installer of the rootkit. The svc.bat file sets the user name of the IRC bot in win.dll (which is actually just a plain text file) and starts both the "Remote Registry Backup" service and the "Microsoft Networks" service. By looking at the services control panel under Administrative Tools, we see that both of these new services are bound to the attacker's "svchost\lsass.exe" file. This results in 3 processes called lsass.exe, though only one is legitimate. The "Remote Registry Backup" service is also bound to "svchost\ident.bat" which executes WinMngr.EXE, while "Microsoft Networks" is also tied to "svchost\regsvc.exe." The effect of starting all of these files as services is to make them impossible to kill via the Windows(R) Task Manager. It is necessary to stop the services these files are attached to in order to bring them down.

Since we have our own copies of these files, we are able to find that the next step taken by svc.bat is to hide the directories created by the zip file. Using the "attrib"

Fig. 1. Windows Explorer view of system compromise

command, the bat file runs: "attrib +S +H spools" (aka C:\WINNT\system32\Setup\spools) and "attrib +S +H svchost." These commands set both the system flag and the hidden flag on the directories causing Windows to hide them unless the flags are unset or someone checks "show hidden files and directories" and unchecks "hide protected operating system files" in the tools->file options->view menu of any Windows Explorer window.

That svc.bat which came from c.exe hides a directory "spools" that came from x.exe suggests that both files were created by the same person and intended to work together as a single rootkit.

The other half of this rootkit, x.exe, provides a number of utilities useful for hacking, such as wget, netcat, fport, and fscan. None of these are called directly by the services that are set up, suggesting the attacker intends to hack other machines from this one. Given this evidence, and the sophistication of this attack, as well as the number of compromised boxes found (some 25 machines on Georgia Tech's campus alone, including the attacking host) it seems likely that the attacking host was being used a relay and the owner is not our attacker.

The third directory created by our attacker contains more interesting tools, including one (kill.exe) that is very useful in purging the system of this rootkit. The first file of importance is thug.bat. This is where the "Virtual Guide Numbering" service is created and bound to "service\lsass.exe" (which gives us a total of 4 processes named lsass running) and "services\winampa.exe" (which the name of the popular media player Winamp's background executable). This gives us two more processes running that are attached to services and can't be killed from the Task Manager. The batch file then hides the "C:\WINNT\system32\service" directory in the same

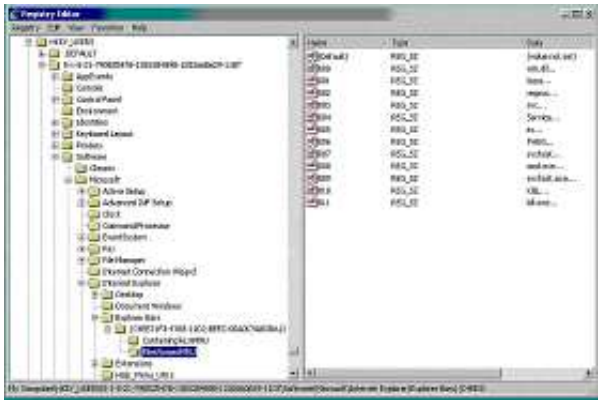

Fig. 2. Registry entry of compromised system

manner as the other two and removes x.exe.

At some point in running these various programs, one of them creates a number of registry keys. This is the rootkit's signature and can be found on any machine infected by an unaltered version. The keys are placed in HKEY_USERS–>S-1-5-21-79052478-1383384898-1202660629-1107(this number may be unique to each install) –>Software–>Microsoft–>Internet Explorer–>Explorer Bars–>{C4EE31F3-4768-11D2-BE5C-00A0C9A83DA1}(again possibly unique to each install)–>FilesNamedMRU. Each one registers the name of a file the attacker installed but not its full path. Currently, it seems logical to conclude that the programs themselves both set and read these keys, thus executing all of the files listed if any one of them is started.

This rootkit does not appear to cause any actual damage to the attacked system (in fact it patches the system against future attacks on port 135), but instead sets the machine up as a warez server via IRC. The bot installed connects to an IRC server and joins a channel, where it broadcasts repeatedly the files it has available for download from the svchost\service directory.

Fortunately, removing this rootkit is not especially difficult after it is understood. Having completed our analysis, we found the following steps to be sufficient:

1: Using the Administrative Tools, stop the services that are infected and set them to manual start.

2: Edit the C:\WINNT\system32\Setup\svchost\x.pid file to find the process id (PID) of the IRC daemon.

3: Using the kill.exe found in either "service" or "spools," kill the pid using C:\kill.exe <pid>. This stops the IRC daemon from running. You may also safely kill all WinMngr.EXE, winampa.exe and cmd.exe processes. You may kill the lsass and svchost processes, but the legitimate versions of these need to be running and may or may not restart properly if killed.

4: Delete, or move the directories the attacker created:

```
C:\WINNT\system32\Setup\spools
C:\WINNT\system32\Setup\service
C:\WINNT\system32\Setup\svchost
```

5: Using regedit, remove all the keys placed by the attacker.

6: If you have not stopped all the attacker's processes, or if you wish to be sure they are all gone, reboot the machine.

7: Upon first booting the machine, ensure that you have only one copy of lsass.exe running, the registry keys are gone, and that none of the illegal services started. This is your indication that the machine is clean. (When we originally did this, we reinfected the machine in order to establish that we had found everything, and to learn to start/stop the ircd process.)

From this analysis, we found and reported 25 compromised hosts on Georgia Tech's network, which were illegally sharing copyrighted material. In addition, we were able to provide Georgia Tech's Office of Information Technology (the managing authority for Georgia Tech's network) with specific instructions for the effective diagnosis and cleaning of any similarly compromised hosts they might find. We also have the initial exploit code used, and so could instruct our firewall or IDS to defend against it.

V. Conclusions

With the current insecurity of networks on the Internet and the shortcomings of current tools, honeynets are useful to system administrators. As the above example shows, honeynets are a powerful instrument for finding compromised hosts and learning how to repair them. System Administrators can use honeynets along with more traditional network defenses (i.e. firewalls and intrusion detection systems) to help secure their networks. honeynets then are not just useful to the researcher, but may be used to tangibly improve the security of large enterprise networks.

References

[1] Network Associates Technology, Inc. `http://www.nai.com/us/security/home.asp`.

[2] McClure, Scambray, and Kurtz, *Hacking Exposed: Network Security Secrets & Solutions, Fourth Edition*. New York, NY: McGraw-Hill/Osborne, 2003, pp. 36-68.

[3] J. Levine, R. LaBella, H. Owen, D. Contis, and B. Culver, "The use of honeynets to detect exploited systems across large enterprise networks," in *Proceedings of the 2003 IEEE Workshop on Information Assurance and Security*, (United States Military Academy, West Point, NY), June 2003.

[4] P. Kaufman and Spencier, *Network Security Private Communications in a PUBLIC World*. Upper Saddle River, NJ: Prentice Hall PTR, 2002, pp. 586-593.

[5] McClure, Scambray, and Kurtz, *Hacking Exposed: Network Security Secrets & Solutions, Fourth Edition*. New York, NY: McGraw-Hill/Osborne, 2003, pp. 481-502.

[6] Spitzner, *Honeypots: Tracking Hackers*. New York, NY: Addison-Wesley, 2003, pp. 58-64.

[7] The Honeynet Alliance. `http://project.honeynet.org/challenge/results/index.html`.

[8] The Honeynet Alliance, "Honeypots Mailing List."

[9] The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. New York, NY: Addison-Wesley, 2002, pp. 75-94.

[10] Spitzner, *Honeypots: Tracking Hackers*. New York, NY: Addison-Wesley, 2003, pp. 65-68.

[11] The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. New York, NY: Addison-Wesley, 2002, pp. 104-106.

[12] T. Jackson, "Windows 2000 rootkit analysis." `http://users.ece.gatech.edu/~owen/Research/HoneyNet/Quarterly/Analysis_of_Windows_2000_root-kit.htm`, January 2004.