Malware, Botnets, and Rootkits: Section 1 Transcript

Introduction

Welcome to the Malware, Botnets, and Rootkits module. During this module, you will learn the basics of malware, how to identify various malware, and how to analyze malware.

Throughout this module, you'll be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Describe the different classes of malware and their key features or abilities,
- Describe the different noise signatures from various malware, and how they can affect your ability to operate remotely,
- Identify malicious programs on a remote host,
- Analyze malware based on its observed fingerprint, and
- Describe the different types of rootkits, their methods of infection, privilege levels, and methods for detection.

This module does not contain a Bypass Exam. Click the Next Section button to proceed to the course material.

Malware, Botnets, and Rootkits: Section 2 Transcript

Malware 1/16

Malicious software, commonly referred to as malware, is software that is designed to damage or do other unwanted actions on a computer system. This software is created by vandals, blackmailers, criminals, or other people with malicious intent. As you can see, there are many different types of malware, with some being more threatening than others. Let's continue on and look at some of the different types of malware.

Malware: Viruses 2/16

The most commonly-known type of malware is the computer virus. Computer viruses are programs that can copy themselves and infect a computer. The term virus is also commonly, but erroneously, used to refer to other types of malware, adware, and spyware programs that do not have reproductive ability. We will cover the other types of malware, adware, and spyware throughout this module. A true virus can only spread from one computer to another, in some form of executable code, when its host is taken to the target computer; for instance, because a user sent it over a network or the Internet, or carried it on a removable medium such as a floppy disk, CD, DVD, or USB drive. Viruses can increase their chances of spreading to other computers by infecting files on a network file system that is accessed by another computer.

Malware: Worms 3/16

Worms are viruses that primarily replicate on networks and execute themselves automatically on a remote machine without any user interaction; however, some worms do require user help, such as mass-mailer worms.

Most worms have one or more exploits built into it, in order to achieve replication throughout the network. They can also provide vectors for other malware, such as spam-bots or backdoors.

Worms tend to get media and vender attention very quickly, which means that companies typically release out-of-cycle patches to combat the spread of a worm. This basically kills the usefulness of the underlying exploit the worm uses. The intrusion detection system, or IDS, developers push out worm signatures very quickly to help defend networks, so even if the target didn't get an antivirus patch update, the exploit may still get caught by the IDS.

Malware: Trojans (Windows) 4/16

Trojan horses, commonly shortened to Trojans, are the simplest malicious programs. Windows-based Trojans are usually executable programs embedded in applications that a user would want to use or try out. Most of the time, the program that is hosting the Trojan performs as advertised, but the Trojan will also extract and run.

Two really popular programs that hackers implant Trojans in are Windows themes and screensavers. It is important to remember that Trojans are not viruses, because they do not have self-replication capabilities.

Malware: Trojans (UNIX)

5/16

The Trojan class of malware is very prevalent on UNIX systems. Primarily, you see this example when a hacker has gained access to a host computer and has either installed modified versions of various commands, such as ps and netstat, or has created wrapper scripts of various commands preventing the user from seeing indications of the hacker's processes or connections. These types of Trojans are referred to as user-mode rootkits.

This class of malware is the reason you cannot trust programs on a UNIX system to report true results. You have to make sure the binary you are calling, `ps` for example, can be trusted by comparing known hash signatures. To guarantee you are using a trusted binary, you can instead upload or use your own trusted version of the program.

The reason this is so prevalent in UNIX, not Windows, is because source code is available to everyone in UNIX. Also, UNIX users are generally command line users and rather computer savvy. They understand what `ps` and netstat is and how to use it. Most Windows users could not find netstat, much less understand the output; so, it has never really been necessary to Trojan these programs on a Windows box. Plus, source code for Windows binaries is not available.

Malware: Backdoors

6/16

Backdoors are the malware choice for hackers all around the world. These programs allow for remote TCP and/or UDP access to the infected computer. There are two types of backdoors: listening and reverse connection, also known as a beaconing backdoor.

Listening backdoors create a network socket, usually TCP, which allows the attacker to connect to the infected client. Once installed, many backdoor programs will send the infected host's network information to a predetermined site, such as an email address, website, or an Internet Relay Chat (IRC) channel.

Malware: Droppers

7/16

Droppers are designed to install some sort of malware (virus, backdoor, etc.) to a target system. Droppers may have additional malware contained within it, as a way to avoid detection by virus scanners. This is known as single stage. Droppers can also download the malware to the target machine from the Internet.

Droppers can be difficult to detect since many of them are designed to self-delete after they successfully perform their intended action, to install or download other malware.

Malware: Keyloggers

8/16

Keyloggers capture whatever is being typed on the target's keyboard. This keystroke information is then sent to the hacker, or the hacker comes back to retrieve the file. Keyloggers capture everything the user types, including deleted input and control characters.

What happens if you gain access to the target and discover that a hacker has already installed a keylogger on it? Well, there's good news and better news. The good news is that when you conduct remote operations, in most cases, you are not typing on their keyboard, so your actions are not being intercepted. The better news is that most keyloggers leave a log file on the system that contains the captured information, so if you can find that log file, you may be able to get the target user's credentials as well.

Malware: Flooders

Flooders, or Distributed Denial of Service, or DDos, tools, use botnets to attack networks or individual devices by sending out massive amounts of network traffic to those targeted systems in hopes of causing a Denial of Service. There are many variants of this type of attack, but some of the popular ones include SYN flood, packet frag, traffic amplification, traffic deflection, and ICMP flood.

An effective variant of the DDoS is the Distributed Reflective Denial of Service, or DRDoS. This is where massive numbers of infected hosts send SYN packets to legitimate servers on the Internet, such as Amazon, Yahoo, or Google, with the SRC IP being spoofed to that of the target system. These legitimate servers respond as they should with a SYN-ACK to the SRC IP.

The massive amount of responses from these legitimate servers can cause the targeted server to become overwhelmed when trying to reply back to all of those SYN-ACK packets and, in effect, cause the targeted server to either slow down dramatically or become unresponsive.

Malware: Logic Bombs

A logic bomb is a piece of code that a hacker inserts into a software system to perform a malicious function when certain conditions are met. Logic bombs can delete files, call home, self-destruct, or even cause physical damage. However, if the code is not malicious and is simply a joke or hidden message, then it is considered an Easter egg, not a logic bomb.

Logic bombs are typically installed by insider threats with direct access to the target system. For example, a programmer might hide a piece of code that starts deleting files, such as a salary database trigger, if they are ever fired from the company.

For additional information on logic bombs, see the Resources for this module.

Malware: Network Sniffers

Network sniffers allow the capture and examination of traffic on a target host or network. This type of malware can be used by system owners for legitimate purposes, or by hackers. Generally speaking, if it has a graphical user interface (GUI) or requires WinPcap libraries to be installed, it is probably being used for legitimate purposes. This is because hackers, more than likely, would never upload and install Wireshark or the entire WinPcap library.

Malware: Spyware, Adware, and Scareware

Spyware is software that collects various pieces of information about a user and transmits that information back to its mothership somewhere on the Internet. Spyware creates lots of privacy issues, as most people do not like their private habits to be sent to someone without consent. Typically, end-user license agreements, or EULAs, are really long and have small print. So when you accept the EULA when installing software, you are actually consenting to sharing your habits with companies without ever realizing it. Both legitimate and illegitimate companies do this. For example, take a look at your network after installing or running iTunes or WinAmp.

There is an upside to spyware. If you are exploiting a system that has spyware installed on it, and you can capture the spyware's traffic leaving the network, then you get lots of useful data about the computer's configuration and other information.

Adware is similar to spyware. It is usually installed without your consent and is most often visible by pop-up or pop-under ads on your computer. Many programs stay resident on your system in order to display pop-ups later.

Scareware uses shock, anxiety, or a perception of threat to scare a target user by convincing them that a virus has infected their system. The scareware tries to get the target user to download and purchase antivirus software that is fake, non-functional, and often malicious.

Malware: Browser Malware

Browser malware, normally installed as plugins, or extensions, also known as Browser Helper Objects, is a class of malware that affects the user's browser and web-surfing experience. The user normally is infected by using a vulnerable browser that is either surfing to an infected site, or is clicking on a link that installs the malware without the user really understanding what is happening. Another method is for the user to install some third-party program, and the adware or object being installed along with it is a default setting.

Malware: Spam

Spam is unsolicited email, usually from someone trying to sell something. The difference is that spammers are not targeting specific individuals.

Here is a typical example of spam; the majority of spam is for cheap pharmaceuticals. It relies on the fact that someone somewhere wants cheap medications. Notice the from and to lines. It is the same address, which is extremely common. They usually do not attempt to make the email look like it was sent from a legitimate company, since they go through so many email addresses to prevent being blocked.

What is interesting about most pharmaceutical spam outfits is that they typically do not commit credit card fraud, because these groups depend on credit card processors to make money. If customers were getting their accounts hacked or stolen, or products were not as advertised, the spammers would lose money due to credit card chargebacks or the credit card processors would charge higher premiums to process payments. That being said, the validity or safety of the actual products is usually suspect.

Malware: Phishing

Phishing is the criminally fraudulent process of attempting to acquire sensitive information, such as usernames, passwords, and credit card details by masquerading as a trustworthy entity in an electronic communication. Many times, phishers will use legit looking bank logos or web sites to increase the charade of legitimacy of the email.

Phishing email sender addresses appear to be from legitimate sources; although, if you looked at the mail source headers, it would reveal the email was actually sent from a foreign country. These emails try to lure the recipient to click on the provided link, to log into what is supposedly a legitimate account; however, if the link is clicked, a mirrored webpage from the legitimate site would ask the user to start inputting information, such as your full name, login, password, etc. The next thing you know, the phisher has access to the user's PayPal or bank account. The surprising thing is that about 5% of all phishing emails are actually successful.

Section Completed

16/16

Malware, Botnets, and Rootkits: Section 3 Transcript

Bots and Botnets

Bots are a type of malware that allows an attacker to gain complete control over the target computer. Computers that are infected with a bot are generally referred to as zombies. There are tens of thousands of computers on the Internet that are infected with some type of bot, without realizing it.

Botnets are collections of bots that report to the same master or controller. Most bots are designed to infect Windows-based hosts, although there are some bots that work on Linux boxes or even network devices. The main method of controlling botnets is via Internet Relay Chat (IRC). Two other methods are via Hypertext Transfer Protocol (HTTP) or over Peer 2 Peer (P2P).

Botnet Communication

Bots communicate through several different ways. Let's focus on three different ways of communication: IRC, HTTP, and P2P. Click each communication type to learn more.

IRC: IRC is considered old school, but is still the most popular form of botnet communication. This form of communication blends in with normal chat users, but is easily discovered.

The standard IRC port is TCP 6667, but bot herders use ports all over available space. A great way to find bots that use the IRC ports is to key in on the PONG response. An IRC server will send a PING message to check to see if the client is still connected, and an active client will respond with a PONG message to confirm activity.

Although bots can use encryption, very few do, so their traffic is relatively easy to discover and block; therefore, herders have moved to other methods of control, such as HTTP and P2P.

HTTP: With HTTP-based botnets, the herders are mainly counting on the ability to hide in network noise to avoid discovery. The web-controlled bots can connect to websites, looking for a specific URL string or hidden comment to receive tasking. They can also be controlled using Facebook, Twitter, Google Ads, and other Web 2.0 projects.

The limitation with this communication type is that they still rely on centralized command and control; so, taking down the websites that control the botnet will cripple the botnet.

P2P: P2P is a decentralized command and control botnet, which is a more difficult botnet to detect and eradicate compared to centralized methods. A good example of this type of botnet is the GameOver Zeus (GOZ) botnet. For more information on the GOZ botnet, see the Resources for this module.

Bots: Why do we Care?

Bots are just like other malware. The presence of bots can cause you to lose access if they take the system offline; bots are very chatty, so this is a very real possibility. They spread by using worm functionality. They cause an epidemic in the local network, which increases the likelihood that your actions will be detected, or that you will lose access to your target.

Malware: Why do we Care?

4/21

The presence of malware on a remote system could lead to several not-so-good things. For example, if the user is infected and installs a virus scanner to clean the problem out, our tools could get caught by the scan.

Here's another example: Let's say the remote box had an annoying virus on it, and the box is part of an enterprise network. The system admins take the box offline and rebuild it. If that was the only machine you own in the network, you just lost your access.

Here's a third example: There is a hacker on the box with you, who has a backdoor, keylogger, etc. The system admins find the hacker's tools and decide to do a forensic investigation of the box to determine the extent of the damage, and they find your tools as well. Now, operations are potentially compromised not just in the target space, but anywhere the hacker's tool is deployed.

Here's one last example to really drive the point home about how dangerous malware can be: The hacker on the wire with you has a sniffer and is sniffing lots of traffic, including all the traffic you are transmitting. You may think to yourself, "No big deal. My backdoor is secure." You are correct that it is secure; however, what about when you use that great 0-day on the compromised host? The packets aren't secure. So, the next thing you know, your 0-day technique is captured and someone else is using it. Their activities then get caught and that exploit goes from 100% working to patched out.

Coexist with Malware

5/21

You may be worried at this point, and even thinking that there is so much malware, how could you possibly coexist with the target malware and conduct an operation? Well, maybe you can, maybe you can't. You have to look at many different aspects on the target host before determining this.

What kind of security posture does the target have? Is there a free antivirus version installed? Is the commercial antivirus software out of date? Does the overall security posture look like it is run by a bunch of interns who appear to work for chips and soda, and use free software?

What is the malware? Is it a dropper with a mass mailer? Spyware? An application virus? These probably won't affect your operations. On the other hand, if it is a remote backdoor or a sniffer, that could be a problem as someone else can log in anytime they want and possibly catch your activities.

At the end of the day, it is not normally your call to make; however, it is your job to find out everything you can about the malware, download the code and related code, and provide a basic understanding of the box and network. You then provide that information to the competent authority so they can make the call on what to do next.

Malware Detection (Windows)

6/21

Once you find malware on the target system, it is important to carefully go through the steps displayed, to properly know if you can coexist with it or not. Next, we will go over each of these steps in more detail.

Malware Detection: Research

7/21

Now let's look at an example situation of something suspicious that could be malware.

During your normal survey of a process list, you notice a couple strange processes. The first thing you need to do is hit the Internet and search for the named processes. Chances are you will get lots of hits. If the hits show the process is malware of some sort, focus on reports from the antivirus vendors. Here is a report from a reliable antivirus vendor. The overview says that seres.exe is a FakeAlert Trojan that was spammed as a free tool to scan for the Conficker.B worm.

Upon execution, the malware copies itself into a couple locations, with names that match what are seen in the process list. You will also see that it downloads another executable and runs a false virus alert to the user's desktop. The malware similarly creates or modifies some registry entries, so you will want to check those out.

Malware Sandbox

8/21

After you have completed your Internet research, you should use a malware sandbox, an automated analysis tool, to see what the malware is doing. There are several commercially available systems and a few free systems that you can download from the Internet.

The malware sandbox allows you to upload a suspicious executable or zip file to the system. Then the system runs the file, conducts an analysis, and emails you the results.

You could also input a hash to see if the sandbox has a matching record. The site will provide an analysis of the program in question.

Do not submit files to online sandboxes or virus scanners during this class. Once more, do not submit files to online sandboxes or virus scanners during this class. Some of the tools on the targets were custom written and are not for public consumption.

Suspicious Processes: Download

9/21

After doing your research and using a sandbox, find out where the executables are written to. There are several tools that you can use to find this information, such as the PS command in Metasploit, which provides the full path. In this example, they are written to the user's Application Data folder. Next, download the executables for further analysis. Do not download an executable and give it an .exe extension. Rename the files' extension to .ex_. That way, it is not possible to run by mistake on your system.

Some Endpoint Security Products actually grab and alert you when you try to download it. Consult your Wiki or other known information repositories before doing this.

Suspicious Processes: Find DLLs

10/21

After downloading the files, you will get a list of DLLs that are associated with the running processes. Most people use the listdlls tool from sysinternals; however, there is a bug in that tool that currently limits its ability to run from a remote command line. So, to get around that, you can use tasklist with the /m switch to get a list of DLLs that are associated with each process. Once you see the listed DLLs for your specific offending process, inspect those DLLs carefully. If they have names that do not look legitimate or if their file times are close to the file time of the malware, they may be related to the malware and you will want to download them for further analysis as well.

Suspicious Processes: File Handles

11/21

After analyzing the suspicious DLLs, you need to see what file handles the processes may have opened. Handles, or more accurately object handles, are used to access object data that the program does not have the ability to access directly. So they are kind of like an extra layer of abstraction that helps keep potentially harmful programs from using other data on the machine.

Each handle has an entry in an internally maintained handle table that contains the addresses of the resources, and means to identify the resource type. Use the handle tool to see what file handles the process has opened. Using this tool, look for the network-aware handles, such as \device\tcp.

An open handle to index.dat tells us this process probably imported functions from wininet.dll, which Internet Explorer uses. Wininet.dll contains library functions for higher protocols, such as HTTP and FTP. This tells us the process probably has some network functionality. It may point to how the malware downloads another program. In this example, it is lizkavd.exe.

Running a handle is very simple; the -a option outputs all handles for all processes. If you want an output for a specific process, use the -p option and a process ID (PID) or process name.

Suspicious Processes: Registry Keys

12/21

The antivirus report shows that some registry entries are attributed to the process that the Endpoint Security Product suspects is malware. What you should do is to run some queries to see if the registry entries match what the report states.

When running queries in Metasploit, quote all strings, use /s to query all keys and subkeys, and use -v to query for a specific registry key value.

Suspicious Processes: Network Use

13/21

Based on the output so far in our example, it does not seem that the malware is using any network ports, but let's check what processes are controlling the various ports. This information can be found by using fport. Fport is really easy to run and requires no options; type fport and you get output that shows you which processes are controlling what ports.

Next, see if the suspicious processes are opening any network connections. In the example, neither seres.exe nor svchst.exe are opening any socket connections. This does not mean that they won't open a port later, or that they had one opened earlier. It just means that, at this moment, they do not have any ports open.

Suspicious Processes: Timestamps

The next step is to search for any other files that have been dropped, created, or downloaded by the suspicious processes. Run dir in the affected directory and look at the time and file size of the malware. In this example, notice they both have the same file size.

Now look for files that match the same date and time. You can search the entire system for files with similar creation times. This is not a perfect solution, but it should work just fine. We run the find, and limit ourselves to all files that were created that hour, or minute, or any other parameter. It all depends on how granular you want your search to be.

In this search, two other files have the same timestamp. To find their exact location, you can use the built-in search function in Metasploit.

Suspicious Processes: Signed Binaries

When looking for malicious files, looking at unsigned exe, DLL, and sys files is a good start. Sigcheck, from Microsoft Sysinternals website, can tell you if the file is signed by the developer or Microsoft, the file date, product name, version, description, and the publisher name. Keep in mind, this tool queries a database at Microsoft to validate files, so you probably do not want to use this if you don't want your activities known to Microsoft.

Nowadays, most legitimate binaries are signed by the publisher or Microsoft to show they are in fact legitimate. Lots of malware is not signed, although some are, since anyone can purchase a signing certificate. It's also important to remember that some legitimate binaries are not signed. Checking for unsigned files helps to narrow down your search for potential malicious files.

Sigcheck is pretty straight forward. For example, if you want to verify if the file spoolmgr.exe is signed or not, all you have to do is input sigcheck c:\windows\system32\spoolmgr.exe. As you can see, the example is definitely suspicious, because it does not provide any information.

You can also check entire directories for unsigned files. Let's check the drivers directory by inputting the address shown here (sigcheck -u c:\windows\system32\drivers >sigcheck.txt). We use the -u option to show only unsigned files, and then redirect the output to a txt file since it could have a lot of output and potentially kill your backdoor.

Suspicious Processes: File Hashes

There are several websites that have the capability of verifying if a particular file is known-bad or not. All you have to do is submit the file's md5 hash, also known as a message digest. An md5 hash is the digital fingerprint of a file.

In the example provided, using the tool, whois, a hash is submitted to one of the online sandboxes. The returned response is the last time the malware was seen (in UNIX epoch time), and the percentage of antivirus products that noted the file as malicious. Once the UNIX epoch time is translated to a more readable format, the website tells us the malware was last seen 04 Aug 2011 at 19:15:24 Greenwich Mean Time.

Suspicious Processes: Sniffers

The next thing you want to do is make sure the Network Interface Card, or NIC , is not in promiscuous mode, or promisc mode, because this is a telltale sign of running a network sniffer. It is important to note that some interfaces are in promiscuous mode by default, such as various VPN interfaces.

To check for promiscuous activity, run the promiscdetect tool. It checks all interfaces and reports which ones are in promiscuous mode, and which ones are not.

If one of the interfaces was in promiscuous mode, the tool would have noted that; however, in this example, we only have standard modes.

Suspicious Processes: Hex Editors

At this point, you should have a pretty good idea if the file is malicious or not, but there are a couple more things to check for.

Open up the downloaded executable in a hex editor and see if there is anything else suspicious about this program, such as:

- Does it have a backdoor? or,
- Is it network aware?

By network aware, we mean, "Do we see function calls that would indicate the ability to connect to the Internet or a network?" The presence of WinInet functions such as "FtpGetFile" or "HttpSendRequest" would be examples of potential network capability by the malware.

Most malware is packed and cannot be read without some additional work, which is beyond the scope of our research. We do see a list of calls that it makes, but none of them stand out as being network aware or a backdoor. If nothing is obvious, look for clues in the strings of the program.

Suspicious Processes: Memory Dump

The last thing you need to do when investigating suspicious processes is to dump the running memory of the process in question.

There is a fair amount of malware out there that doesn't run as a separate process, but instead uses processes or DLL injection to hijack and run under a legitimate process. This type of malware is harder to discover since they are not necessarily apparent in the process list as themselves. Looking for these clues means looking for more than just processes.

- Are there established connections?
- Are there strange values or entries in the registry run keys? And,
- Are there file names that do not look legitimate, such as svch0st.exe or yyu534df.exe?

If you think a process may be hijacked by malware, one of the things you can do is dump the memory of the running process and then analyze the strings for strange or known-bad things.

For this example, we are using a Meterpreter backdoor, which uses DLL injection. Let's see if Meterpreter shows up in svchost.exe, which is what Metasploit usually uses for its injection. As you can see, there are several svchosts running. To narrow it down, since in this example, we have SYSTEM privileges, we can eliminate any svchost.exe that is not running as SYSTEM. Now there are only two processes, we'll try the one running as PID 1008 first.

Using the process_memdump script in Metasploit, we'll get a dump of the memory running under PID 1008. Since we are looking for a known, in this example, any occurrence of metsrv.dll, makes our job easier. Use strings to read the file contents and grep for metsrv.dll. There are at least six instances of metsrv.dll, so it is safe to conclude that svchost.exe PID 1008 was injected with metsrv.dll from Metasploit.

Keep in mind that many of these commands you are running such as handles, listing dll's and the memory dump will be for the benefit of whomever is going to conduct a deeper analysis later on. You may be able to glean certain information from the output of these tools or commands, but it is mainly for the benefit of others after the operation has concluded.

Suspicious Processes: Results

So, you have jumped through all the hoops, and you've followed all the steps to figure out if there is malware in the program. In this example, we can confirm there is malware and it is the FakeAlert Trojan that was spammed as a free antivirus tool to scan for the Conficker.B worm. If there is malware, then it's time for the moment of truth.

- What did you find?
- What does it do?
- When did it get there? or
- Is it a threat to operations?

Take all that information, plus your opinion on whether or not the malware is a threat to operations, and report it all to your chain of command. They will make the call based on your input and other factors. Just remember you do not make the final call; it is someone else's job to do that.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Malware, Botnets, and Rootkits: Section 4 Transcript

Rootkits

1/11

Rootkits are sets of hacker tools that are used after a hacker gains access to a system. They were originally a set of tools that a hacker bundled up and uploaded to a compromised host in order to assist in their activities. This is still true, but now they have a more defined role. Today, rootkits are considered a package of programs or tools that help the hacker avoid detection and hide information, in addition to assisting the hacker in his or her activities.

Two main types of rootkits are user-mode and kernel-mode. Contrary to their name, rootkits are not designed to gain root access on a system. The hacker already needs that access to install the tools.

User-Mode Rootkits: UNIX

2/11

User-mode rootkits on UNIX systems usually contain trojanized versions of system tools. Since they work at the user privilege level, Endpoint Security Products that work on the kernel level can often find these types of rootkits. Programs like Tripwire and chkrootkit can also find these tools on UNIX systems, which are great for SysAdmins, but useless for you. It is useless because you do not want to upload packages that include more than a single binary, as that could easily get your activities and tools discovered. This type of rootkit does exactly what was discussed earlier, when we talked about classes of malware, specifically, UNIX Trojans.

User-Mode Rootkits: Windows

3/11

User-mode rootkits on Windows systems are different than those on UNIX machines because they do not traditionally use Trojans. Instead, they usually use hooking or injection into a user application, hijacking that application in the process. For example, if the user was to run netstat, the rootkit would intercept the system calls and modify the information returned, hiding the fact that the rootkit had a port listening.

User-mode rootkits on Windows systems only require user privileges. A more in-depth look at how user-mode rootkits work on Windows systems can be found in this module's Resources.

Kernel-Mode Rootkits: UNIX

4/11

As the name implies, kernel-mode rootkits in UNIX systems reside as code in kernel space, often as a driver or kernel library. Because they infect the kernel, the attacker normally requires root privileges to install them. Once they are installed and running, they have the ability to affect the entire system and user base. The job of most of these tools is to hide processes, files, directories, and network connections.

The three main ways to implement kernel-mode rootkits are:

- Altering memory directly,

- Loadable Kernel Module (LKM), and
- Patching a kernel image.

Kernel-Mode Rootkits: Windows

Kernel-mode rootkits on Windows systems are similar to user-mode rootkits on Windows systems, because they involve system hooking or modification of kernel space and require system privileges to install. There are five methods that are currently used by kernel-mode rootkits. They are:

- System Service Descriptor Table (SSDT) hooking,
- I/O Request Packet (IRP) hooking,
- Detour patching,
- Direct Kernel Object Manipulation (DKOM), and
- Interrupt Descriptor Table (IDT) hooking.

Refer to this module's Resources for more information on these methods.

General Tips for Detecting Rootkits

When exploring a target system, there are several ways to find rootkits. You can check for known bad files and drivers, look for open handles, and compare ports in netstat to the fport output. Custom tools and memory forensics are also popular methods to discover installed rootkits, but are out of scope for most remote operations.

Click each technique to learn more.

Drivers: Drivers.exe from Microsoft will list all running drivers. No options are required to run it. You can look for instances of known bad drivers listed and loaded, to possibly detect rootkits.

Handle Trick: Sometimes handle can find certain types of rootkits. Handle shows open file handles, and if you use the -a option, it shows open handles to all objects. Each PID should have a corresponding process with it. For example, PID 656 could be the PID for csrss.exe. When installed, some rootkits hide processes; however, these processes can still show up in the output of handle, but will show up with a PID and no corresponding process name.

Run handle and look for instances of the term Non-existent Process. The first indications show up under open handles for csrss.exe. That provides the PID number, and later on, that PID number will be listed with its open handles.

Detect Presence: It may be possible to detect the presence of malware, if you compare the netstat output to the fport output, which lists connections and the full path to the program that has the socket open. The next thing you can do is to try and create a file or directory with the same name as the rootkit tool. Use the full path from fport of the program and attempt to create a directory in the same path. Because two files cannot have the same name, you will get an error if the program is there, but hidden.

Detecting UNIX Rootkits

There are several tools on the market that help detect known rootkits. The drawback is that, if these tools are not aware of a particular rootkit, they may not detect its presence. There are a few open source tools that are used to look for rootkits, such as chkrootkit and rkhunter.

Although they are not in wide spread use, there are several antivirus programs on the market for UNIX systems. Many of these are rootkit aware. On the other hand, there is nothing to keep a rootkit from being antivirus aware. Vendors such as ClamAV and AVG provide free Linux antivirus products. Kaspersky and Panda are just a couple examples of commercial antivirus tools that provide rootkit awareness.

A simple method for detecting user-mode rootkits is to check programs against known good hashes. A user-mode rootkit usually modifies or wraps a known binary, such as ls, ps, or netstat. These modifications change the md5 hash signature of the file. By comparing the file's hash against known signatures, you can detect programs that should not be trusted. If the output from your copy of netstat and the system's copy of netstat is different, then the system may be compromised.

Detecting Windows Rootkits

8/11

Some Windows rootkits can be discovered using anti-virus software. Endpoint Security Products flag unusual address ranges found on the System Service Descriptor Table, or SSDT, Interrupt Descriptor Table (IDT), and drivers' I/O request packet (IRP) function tables. They can scan the first few bytes of the kernel functions and look for detours, and they look for suspicious or known bad drivers that are installed. Unfortunately, there are not really other simple ways to find Windows rootkits beyond what has been described.

Rootkits

9/11

There are several other types of rootkits in addition to user-mode and kernel-mode. Click the different rootkits to learn about them.

MBR Rootkits: Master Boot Record (MBR) rootkits work by modifying the MBR so that they are loaded on startup. They hook and patch operating system functions while the operating system is being brought up and will remain active in the system memory during the computer's active state. These types of rootkits are sometimes referred to as "bootkits."

BIOS Rootkits: Basic input/output system (BIOS) rootkits infects the motherboard's programmable memory. It deploys on the operating system before the system boots, and targets specific BIOS environments such as AWARD and Phoenix.

This type of rootkit is very risky, because poor coding will brick the target system. On the other hand, the Unified Extensible Firmware Interface (UEFI) reference model makes this attack easier, rather than harder, on modern systems.

HyperVisor Rootkits: HyperVisor rootkits exploit CPUs that support hardware virtualization. It moves the host operating system into a VM, where the rootkit has full control over it.

UEFI: UEFI, or Unified Extensible Firmware Interface, is a specification that defines a software interface between an operating system and platform firmware. UEFI is meant to replace BIOS and since it is a reference model, the days of the proprietary BIOS (hence the requirement for specific BIOS malware customization) are going away, making BIOS-based malware easier to write.

Rootkits: Why do we Care?

So why should we care about rootkits? Because rootkits are not installed by mistake. When they are deployed, it is usually because someone somewhere wanted access to that specific computer. They're designed to hide the presence of a hacker on the machine, while keeping their activities and their files away from prying eyes. Rootkits pose a serious risk to operations, because seasoned hackers, not script kiddies, use them. If you are conducting operations on a box that has a rootkit, your activities could be discovered and our tools could be compromised.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Malware, Botnets, and Rootkits: Section 5 Transcript

Summary

You have completed the Malware, Botnets, and Rootkits module. During this module, we discussed how to detect, identify, and analyze suspicious programs.

You should now be able to:

- Describe the different classes of malware and their key features or abilities,
- Describe the different noise signatures from various malware, and how that can affect your ability to operate remotely,
- Identify malicious programs on a remote host,
- Analyze malware based on its observed fingerprint, and
- Describe the different types of rootkits, their methods of infection, privilege levels, and methods for detection.

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam.

Click the Next Section button to begin the Module Exam.