Malicious Code
Lesson Introduction

- Reasons attackers use malware: automation, scalability, and deniability.
- Attackers release malicious programs on the Internet and let them spread
- Overview of malware

We covered a number of topics so far. But you'll see that all of them have focused on problems that we address in the context of a single computer. Unfortunately, a single computer is not very interesting by itself. It has to be connected to a network. And when we connect computer to our networks that brings in a whole bunch of new security problems. That's where we're going to go on to the second part of the course. We have a world-class network security researcher who's actually going to take you there. >> For part of this course, we're going to cover a number of major topics in network security. We're going to start off with malware and network defenses. Then we're going to move on topography, security protocols, web security and mobile security.

There are several reasons why attackers would want to carry out their attacks through malware or malicious code. They can achieve automation, scalability and deniability. For example, they can do these malwares on the Internet, let the malware spread and carry out the attacks on their behalf. In this lesson, we going to give overview of malware. And cover several kinds of malware. In the next lesson, we going to discuss the more advanced malware.

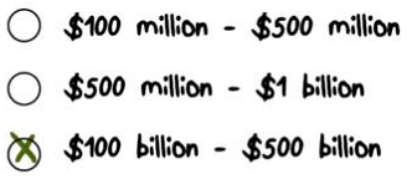
What is Malware? Quiz

What are the estimated yearly losses due to cybercrime worldwide?

- ◯ $100 million - $500 million
- ◯ $500 million - $1billion
- ◯ $100 billion - $500 billion

Instructor Notes:
Cyber Crime Costs Global Economy $500B a Year

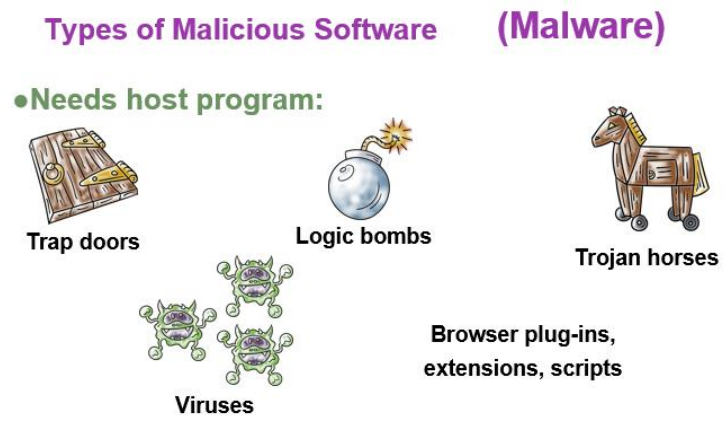
To get you to start thinking about malware, let's do a quick quiz. What are the estimated yearly losses due to cybercrime worldwide? Is it $100 to $500 million? Or $500 million to $1 billion? Or $100 billion to $500 billion?



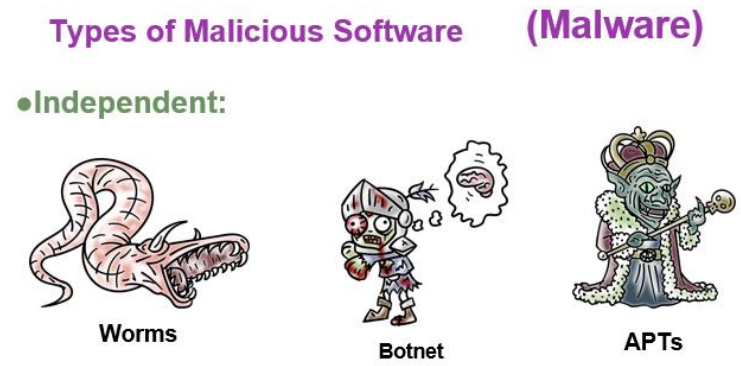
According to the Center for Strategic and International Studies and reports from various antivirus and security companies, the correct answer is $100 to $500 billion. This is a huge amount of money. For comparison, drug trafficking results in about $600 billion a year. So cybercrime is in par with drug trafficking. And you may wonder why such a huge number. Now of course, this number includes the direct financial losses, such as when a credit card number is stolen, or bank account is compromised. But it also includes the cost due to productivity loss, such as the need to compare the computer and systems after a cyberattack. And it also includes losses due to intellectual property when valuable information is stolen.

There are two major types of malware. The first kind of malware needs host program, meaning that they have to be embedded in the host program in order to run and spread. The second type of malware is independent, meaning that they themself are independent programs that can run by themself. We will study some of them in this lecture and cover the rest in the next lecture.

When we say a malware needs a host program, we mean that the malware is embedded in the existing program so that you can enter program, runs on the system and then spread from there.



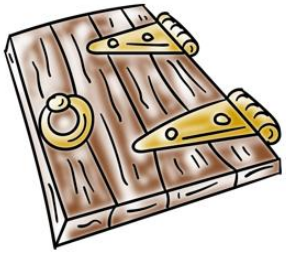
There are several ways for malware to imbed itself into a program. For example, trap doors, logic bombs, Trojan horses, viruses, and malicious browser plugins and extensions and scripts and so on.



And the independent malware does not need a host program, because this malware are complete programs by themself. Examples of these malware include worms, botnets, and advanced persistent threats, or APTs. We will discuss botnets and APTs in a later lesson.

Trap Doors

- A secret entry point to a program or system.
- Typically works by recognizing some special sequence of inputs or special user ID.



Instructor Notes:

Flight Simulator Easter Egg

Now let's discuss some more details of the various types of malware. The first is trap doors. Trap doors is also known as back doors.

It is a sequence of instructions in the host program or system that has been embedded by a programmer and can be activated by the attacker. Essentially, a trap door provides a secret entry point to a program or system, and this secret entry point is typically known only to the programmer and the attacker.

A backdoor in a program typically works by recognizing some special input command, such as a sequence of input specifically crafted, or a special user ID. For example, an attacker can gain access to a system through the back door without providing the proper user authentication. A famous benign version of a trap door, sometimes called an Easter egg, is the fly simulator in the 1997 version of the Microsoft Excel program. The user when entering undocumented series of commands, can gain access to a flight simulator program embedded within Microsoft Excel.

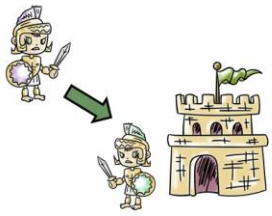
Logic Bombs

- Embedded in some legitimate program
- "Explode" or perform malicious activities when certain conditions are met.

A Logic Bomb is, essentially, a trigger planted in a program. When the triggering condition is met, the planted code then execute. In such a way, malicious activities can be activated whenever a condition is right.

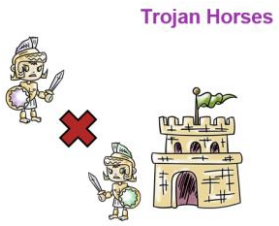
For example, a branch of the program will launch, denounce service attacks to whitehouse.gov only when the current time is the specified time and date. And that's an example of a logic bomb.

Trojan Horses



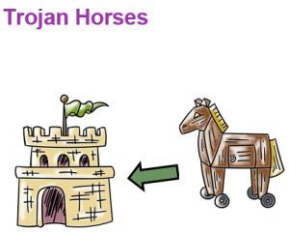
Trojan Horses get their name from a tale from the Trojan Wars. It is said that the Greeks wanted to enter the well fortified city of Troy.

Rather than launching a direct assault at the city and suffering huge losses, they devised a wooden horse and they hid their soldiers inside the horse.



Then the left the horse outside the gates of Troy as a gift.

The Trojans thought the horse as a gift to acknowledge that the Greeks had been defeated. And so they brought the horse into the city of Troy.



At night, the Greek soldiers hidden in the horse came out. And they let their fellow soldiers waiting outside the city come in as well.

As a result, the Greek soldiers passed all of Troy's defenses, and destroyed the city, and won the war.

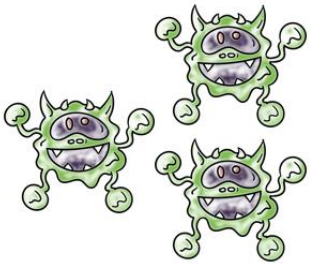
Trojan Horses

- Hidden in an apparently useful host program
- Performs some unwanted/harmful function when the host program is executed

In the context of malware, a Trojan horse is a piece malicious code embedded in a utility program that a user will run frequently. That is, when a user runs this useful program, the malicious code, or the Trojan horse is also executing.

An example of a Trojan Horse is a login program that performs key logging meaning stealing user login and password and pass along such confidential information to an internet server. The login program will still allow the user to log in by calling the real login subroutine because otherwise the user would notice. Many malicious browser extensions or also perform key logging and phishing, in addition to some useful functions and these are the latest examples of Trojan Horses.

Viruses

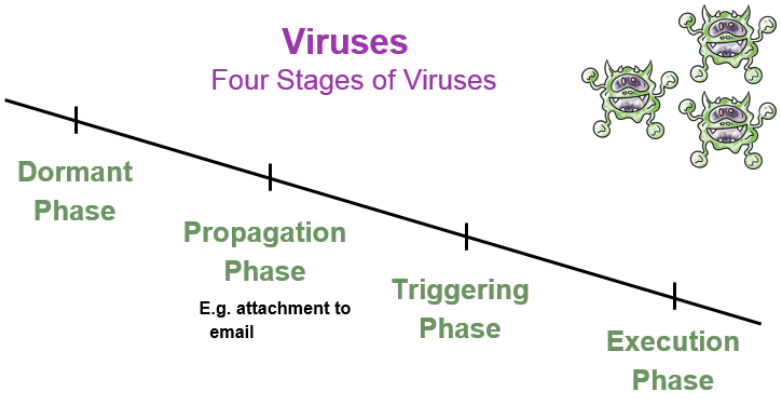


- Infect a program by modifying it
- Self-copy into the program to spread

Viruses is perhaps the best known type of malware.

A virus infects a program, by modifying the program code so that when a program runs, the virus code also runs. It then self-copy into other programs, and thus, spreads itself.

There are many four stages in the life cycle of a virus.



The first is a dormant phase. This is the phase when a program has just been infected by a virus but program has not run yet so the virus has not been triggered or spread.

The second stage is propagation. This is when the malware's being sent around or spread. For example, the malware can come as an email attachment and the email attachment is being sent to many users.

The third phase is the triggering phase. This is when the host program is being run. And as a result, the virus is also triggered to run. For example, when a user clicks an email attachment that contains a virus, it's triggering the virus to run as well.

The fourth stage is the execution phase. This is when the virus code runs, performs some malicious activities, and most importantly, it looks for targets to infect so that it can spread. For example, in a case of email attachment, when the virus runs it can search for users in the address book, and then send email attachment with the virus to users in the address book. And that's how he can be propagated, triggered, executed, and again propagated, triggered, and executed. And this is how virus spreads.

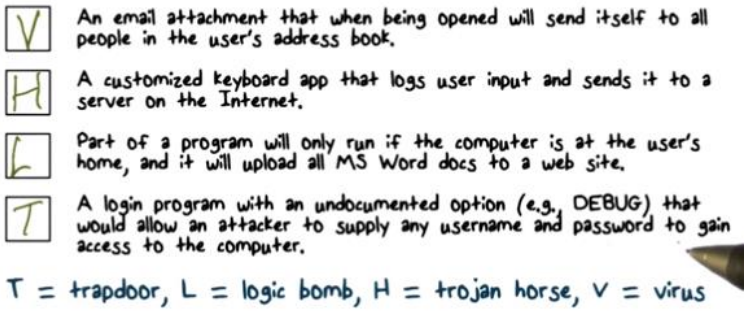
Host-Required Malware Quiz #1

Determine which category each of these belongs to:

- ☐ **An email attachment that when being opened will send itself to all people in the user's address book.**
- ☐ **A customized keyboard app that logs user input and sends it to a server on the Internet.**
- ☐ **Part of a program will only run if the computer is at the user's home, and it will upload all MS Word docs to a web site.**
- ☐ **A login program with an undocumented option (e.g., DEBUG) that would allow an attacker to supply any username and password to gain access to the computer.**

T = trapdoor, L = logic bomb, H = trojan horse, V = virus

For this quiz you write in the box the type of malware. And the choices are T for trapdoor, L for logic bomb, H for Trojan horses, and V for virus.



So the first malware, an email attachment that when being opened will send itself to all people in the user's address book. As we have discussed a little bit earlier, this is a virus. The second malware, a customized keyboard app that logs user input and sends it to a server on the Internet. This is a Trojan Horse, because while it performs some useful function, it also performs some malicious activities. The third malware, part of a program that will only run if the computer is at the user's home and it will upload Microsoft Word documents to a website. This is a logic bomb, because the triggering condition is the place or geolocation or IP address of the computer. Which is at the user's home. And when the triggering condition is met, it performs malicious activities. The fourth malware, login program with an undocumented option, for example DEBUG, that would allow an attacker to supply any username and password to gain access to the computer. This is a trap door, because it allows an attacker to gain access to system without going through the proper security check.

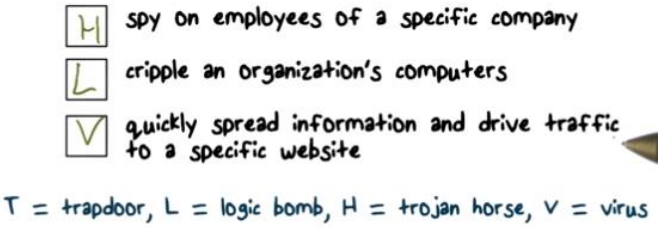
Host-Required Malware Quiz #2

Which type of malware would be best for each of the given tasks?

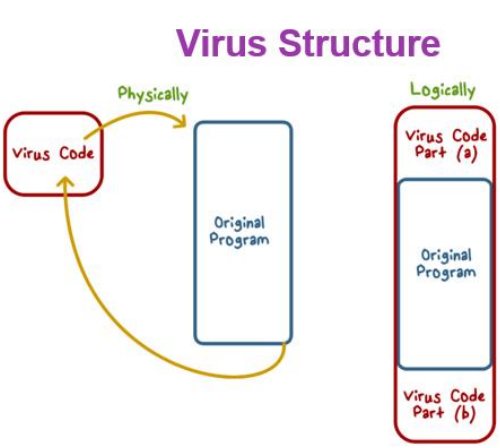
- ☐ **spy on employees of a specific company**
- ☐ **cripple an organization's computers**
- ☐ **quickly spread information and drive traffic to a specific website**

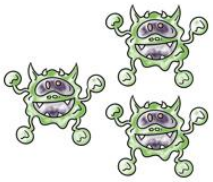
T = trapdoor, L = logic bomb, H = trojan horse, V = virus

Let's do another quiz of malware, that requires host programs. Here in the box, you specify the type of malware that would be best for the given task.



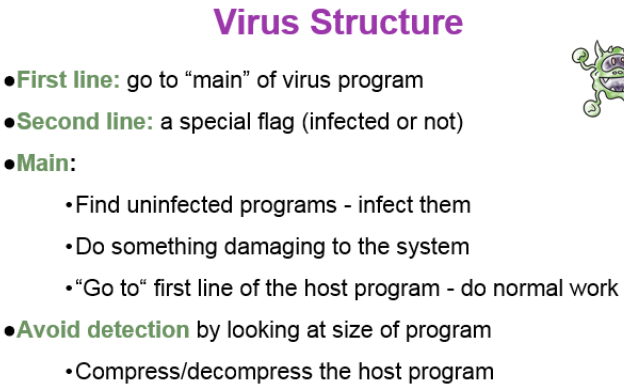
So first, spy on employees of a specific company. You can do this with a Trojan Horse. For example, the Trojan Horse can come in the form of a utility program, such as a company calendar, that also spies on the employees. The second task, cripple an organization's computers. This is a logic bomb. For example, a logic bomb can be inserted into the company's computer servers so that when the time is right, the server will shut down. The third task, quickly spread information and drive traffic to a specific website. This is a virus. As we know, virus can spread quickly, for example, through email attachment. And, when triggered, it can perform a number of malicious activities, such as driving traffic to a website.

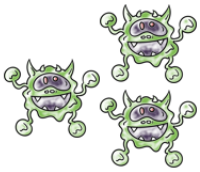




Now, let's discuss some of the details of viruses.

First, let's take a look at the structure of virus. A virus infects a program by modifying the program code. That is, the virus code has to be physically inserted into the program file. Logically, when the infected program runs, the virus' codes run first, then the original program will run, so that the user will not suspect that the program has been infected. And then at the end, there could some virus code that does clean up to avoid detection.





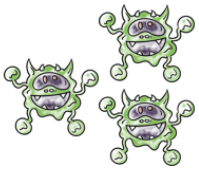
If you look at the infected program, the first line controls that the virus program will always run first. It is critically important to put this control in the first line of the infected program, because this is the only way to guarantee that the virus code will always be run whenever the program executes.

It is also important to put a marker in the infected program. Such as putting a special flag in the second line of the infected program to indicate whether the program has been infected by the virus or not. Otherwise a program can be repeatedly infected.

When the virus code is run, it typically first finds other programs to infect. Of course, it will check whether a program has been infected already by looking at the special flag. In addition to infecting other programs, the virus code can also perform other malicious activities on the system, such as stealing valuable documents. After performing the malicious actions the virus will then transfer the control to the original program so that the normal work can be performed in such a way the user would not notice.

The virus code can also perform other actions in order to avoid detection. For example, because the virus code is physically inserted into the original program file the file size of the original program obviously increases. And this can be a tell tale sign that a program has been infected. Therefore, in order to avoid detection, the virus code can compress the infected program so that the file size is the same as the size of the program before it is infected.

Types of Viruses



- Parasitic virus: scan/infect programs
- Memory-resident virus: infect running programs
- Macro virus: embedded in documents, run/spread when opened
- Boot sector virus: run/spread whenever the system is booted
- Polymorphic virus: encrypt part of the virus program using a randomly generated key

Now let's look at the different types of viruses.

The first is a parasitic virus. They typically scan programs on a system, for example on the hard drive, and then infect these programs.

The second is the memory-resident virus. They're typically of an operating system, and when the system runs, the opening system is loaded into the memory. So as long as a system is running, the virus, resize the memory. Then it can infect any running program on the system.

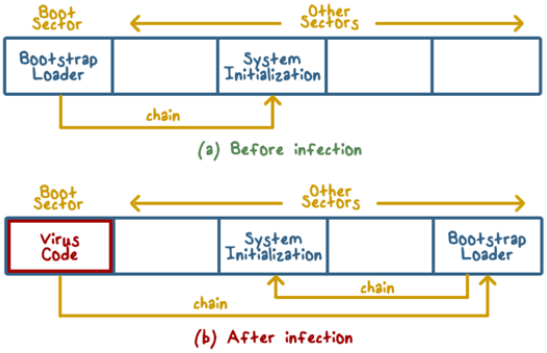
The third is the macro virus. They're typically embedded in a document. And when a document is opened, the virus also runs and spreads.

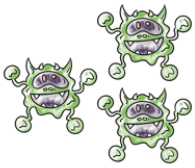
The fourth, is the Boot sector virus. They reside in the boot sector of a hard drive, and whenever a system is booted the Boot sector virus will run and spread.

A very important type of virus is called a Polymorphic virus. For a Polymorphic virus, each instance, or each infection, can look different because part of the virus program is encrypted by a randomly generated key at each infection. The purpose of using polymorphic virus is to avoid detection by easy signature matching. We will discuss this a little bit later.

We should note that any of these type of viruses can by polymorphic.

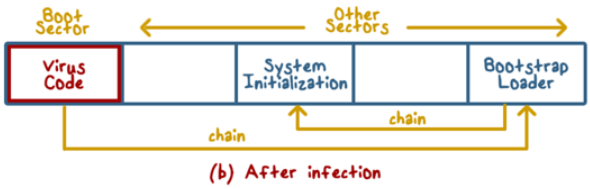
Boot Sector Virus





Now let's discuss boot sector virus in more details.

First let's look at how boot sector works. A boot sector is a special sector on the hard drive of a system. When a system is booted the code in a boot sector will always run first. In a code, it's called bootstrap loader. The bootstrap loader is typically responsible for loading the operating system. For example, it may ask a user to choose a list of operating systems to boot from. For example, the bootstrap loader typically may ask a user to choose an operating system from a list and then loads that operating system. And this is how system boots from a hard drive. Again, it starts with code in the boot sector. And then when a bootstrap loader runs, it loads the operating system.



When a boot-strapper virus infects the system, the virus code is inserted in the boot sector. And the reason is, boot sector again, is a special place in the hard drive and the code there will always be executed first when the system boots. So, by putting virus code there, whenever system boots the boot sector virus will run. Then, of course, the boot sector virus can perform a number of malicious functions such as infecting other programs on the system, spreading to other systems and stealing useful documents from the system. After the virus code runs, the boot sector virus should transfer the control to the original bootstrap loader so that the system can boot normally, at least appear to the user that the system boots normally.

Macro Viruses

Macro:

- An executable program (e.g. instructions opening a file, starting an application) embedded in a word processing document, e.g. MS Word

Now let's take a look at macro viruses.

First, what is a macro? A macro is actually a program embedded in a document, such as a Microsoft Word Document. It typically contains instructions for some useful functions, such as opening a file or starting a new application.

Macro Viruses

A common technique for Spreading:

- A virus macro is attached to a Word Document
- Document is loaded and opened in the host system
- When the macro executes, it copies itself to the global macro file
- The global macro can be activated/spread when new documents are opened

And because a macro is an executable program, it can be infected by viruses just like any other executable programs.

What's unique about macro viruses is that users typically don't suspect that a document will contain a virus. Here's how a macro virus can typically spread.

First, the attacker creates a macro that contains a virus and then attach it to a Word Document. And then this document can be sent around, for example, through e-mail attachment.

And then, when an unsuspecting user clicks on the e-mail attachment and opens the document, the document is opened on the user's computer.

When the document is opened, the macro executes and as a result, the macro virus also runs. The virus then copies itself to the global macro file. When the document opens, the macro executes and the macro virus also runs. When the macro virus runs, it can perform a number of malicious activities, such as sending the same Word Document to a number of users in the user's address book as an attachment. And the spreading itself.

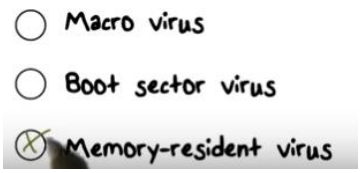
What's more interesting is that the macro virus can copy itself to the global macro file. As a result whenever the user opens a new document or creates a new document, the global macro will be copied into the document, and that's another way that the macro virus can spread.

Types of Viruses Quiz

Which type of virus begins on the **operating system level**?

- ◯ **Macro virus**
- ◯ **Boot sector virus**
- ◯ **Memory-resident virus**

As we discussed, macro virus is embedded in a document. So it's not really at the operating system level. Boot sector virus, as we discussed, boot sector virus resides in the boot sector of the hard drive. And it runs before the operating system is loaded. So it's not really at the operating system level. Memory-resident virus, as we discussed it is embedded in the operating system, so that whenever a system runs, the virus stays in the memory and it can infect any running program. So therefore, memory-resident virus begins at the OS level.



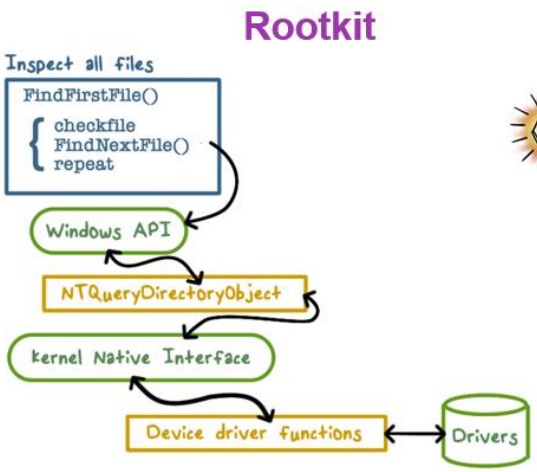
Rootkit

- **Resides in operating systems**
  - Modifies OS code and data structure
- **Helps user-level malware**
  - E.g., hide it from user (not listed in "ls" or "ps" command)

Now, lets discuss a special kind of memory resident virus called Rootkit.

A Rootkit is embedded in an operating system. It typically modifies some of the code and even data structures of the operating system in order to perform some malicious activities.

For example, a Rootkit can be used to hide a malware from the user. For example, when the user uses the ls command to list the contents of a directory, the Rootkit can change the output of the ls command so that the user will not see the malware file. Similarly, when the user uses the ps command to see what programs are running on a system, the Rootkit can modify the output of the ps command to hide the running of the malware.

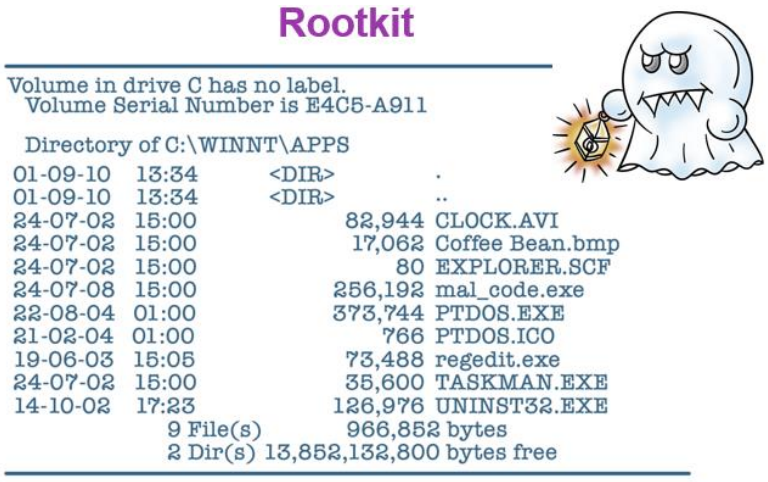


Let's study an example of how Rootkit can modify the operating system in order to perform malicious activities. For example, the rootkit is trying to hide the malware file from the user when he lists the contents of a directory.

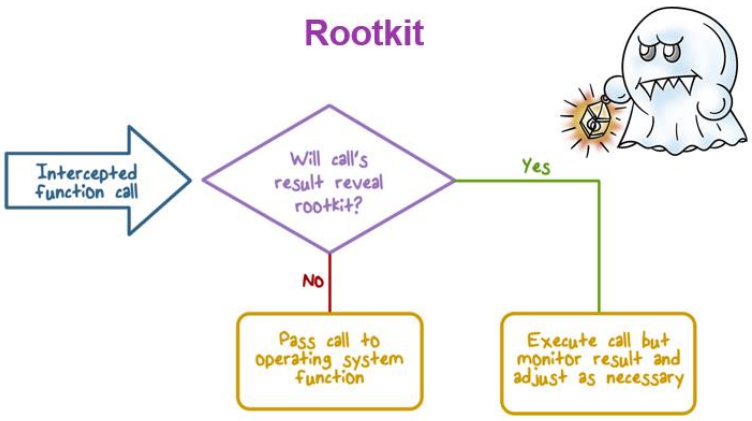
First, let's examine what happens when a user looks at the files in a directory. Suppose on Windows, the user use the command D-I-R, DIR, for looking at files in a directory. As we show here this command can be implemented by a loop that keeps looking at the next file in the directory.

Now, let's look at how a Rootkit can hide a malware from the user when he looks at the files in a directory. We know that files and directories, they reside on hard drive, which is controlled by the operating system. Meaning that, any access to the hard drive has to go through the operating system. Therefore, in order to get information about files in the directory we have to go through operating system functions in order to get such information.

In other words, operating system functions are being called to look at information about files and directories on hard drive and return the results back to the user.

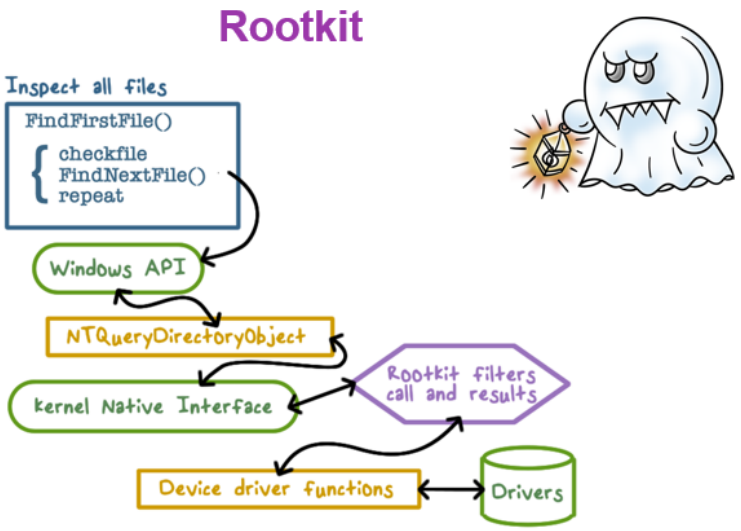


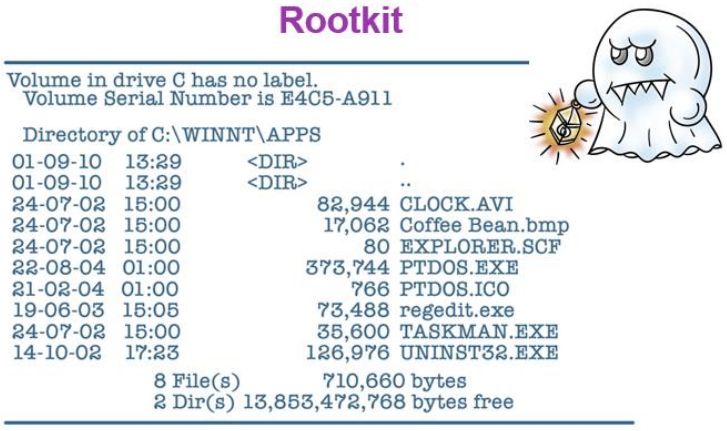
So here is what an operating system will typically return when the user looks at the files in a directory. That is, suppose the Rootkit is not embedded in the OS yet, then the OS will return all the files in the directory including the malware file, say mal_code.exe. And this is the file that the Rootkit would try to hide from the user.



In order to hide the malware from the user, what the Rootkit can do is to intercept any function call to the operating system. And then reason whether the call will end up revealing the malware. And if no, it will just pass the call to the appropriate operating system function. And if yes, it's going to execute the call, but intercept the result so that it can filter out the result as necessary, in order to hide the malware.

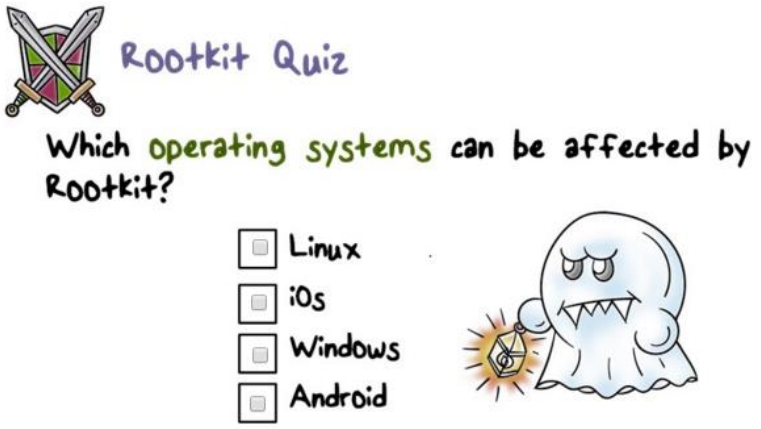
That is the Rootkit intercepts the function call to the operating system and knows that the operating system function call is looking at files in a directory, and then it knows that the return results may contain the malcode.exe. So the Rootkit will filter out the file name so that the user will not see this file name in the result.





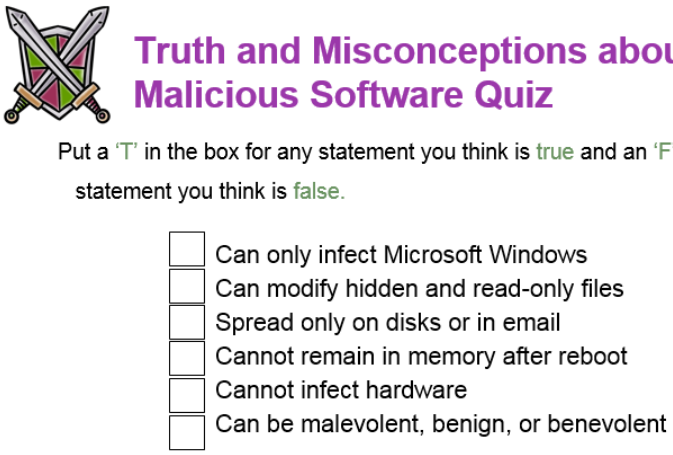
So here you go, when the Rootkit is embedded in the operating system, you can filter out the malware file. So when a user looks at the files in a directory, he will not be able to see the malware and this is how a root kit in the OS can hide a malware from the user.

Again, the root kit is able to accomplish this by modifying the operating system. In particular, it intercepts a function called to the operating system, and future the result of the function calls.

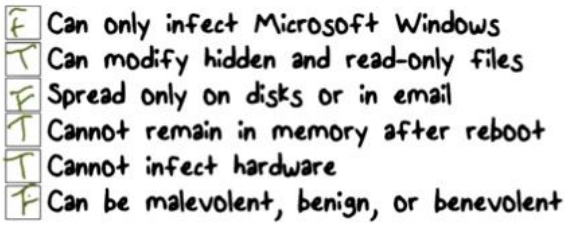


The question is, which operating systems can be affected by Rootkit? Is it Linux, iOS, Windows, Android, or all of them?

The correct answer is that all of them can be affected by Rootkit. Again the reason is that a Rootkit is a piece of malcode that can be inserted into any operating system.



Here I'm going to ask you to tell me which of the following statements are true and which are false.

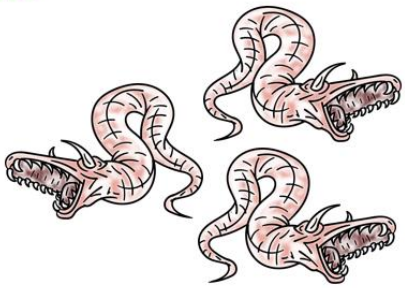


So the first statement, can only infect Microsoft Windows. This is clearly false. The second statement, can modify hidden and read-only files. This is true. You may ask, how can a malware find a hidden file or modify a read-only files? If you think about it, a malware can be inserted into the operating system, which has the highest privilege. It can find any hidden file on a system, and it can override any permissions, such as read-only permission. The third statement, spread only on disks or in email. This is clearly false. The fourth statement, cannot remain in memory after reboot. This is a somewhat tricky question. If this refers to the particular instance of the virus, then this statement is true. Because after reboot, the memory is wiped clean, and start over. On the other hand, although we say that the particular malware instance is gone, after the system reboots, we should know that if the virus is part of a operating system, then when the system reboots, the virus will be loaded into the memory again. So we really have to think about this statement in a context. The fifth statement, cannot infect hardware. This is true. But we have to note that by hardware, we mean, that hardware that does not have any software component, such as firmware. The last statement, can be malicious or benign. Although this may appear to be subjective, I will say the answer is false. The reason

is that, any malware that gets on your system without your knowledge or authorization, already violates security policy, and therefore, it is already malicious.

Worms

- Use **network connections** to spread from system to system



Now let's discuss malware that does not require host programs. The first type that we're going to discuss is Worms. Worms are independent malicious programs and they typically use network connections to spread from one system to the other. Worms represented a major advance in malware in the 1990s coinciding with the rapid expansion of the internet. Worms later evolved into botnets around 2005, which is still the dominant form of malware today. We will cover botnets in a later lesson.

The Internet Worm

What it did:

- Determine where it could spread
- Spread its infection
- Remain undiscovered and undiscoverable

Instructor Notes

CERT

Now let's discuss the first major Internet worm, also called the Morris worm after its creator Robert Morris. Here's how the worm worked.

When the worm runs on a system, it looks for other systems on the Internet that it can spread to. For example, these systems have some security vulnerabilities that the worm can exploit.

And then by exploiting these vulnerabilities, the worm can infect these systems. In other words, it can lure itself to these systems and that's how it spread.

Once the worm got on a system, it also employed a number of tricks in order to keep himself from detectable.

The Internet Worm

Effect:

Resource exhaustion - repeated infection due to programming bug

- Servers are disconnected from the Internet by system admin to stop the infection

According to each creator, the Internet worm was made as part of an experiment to measure the size of the internet. For example, by measuring how many computers are connected together. However, there was a programming error in the code. That is the worm would infect a computer regardless, whether the computer had been infected already one out of seven

times. This proved to be too aggressive. And as a result, many computers get infected repeatedly. That is on these computers, there are many instances of the same worm running resulting in resource exhaustions. And that's really how the Internet worm was discovered, because the system admins find out that their servers were overloaded. As a response, many system admins disconnect their servers from the internet in order to stop the spread of the Internet worm. But since many servers were disconnected, the Internet was disrupted as well.

The Internet Worm

- **Exploit security flaws**
  - Guess password (encrypted passwd file readable)
  - fingerd: buffer overflow
  - sendmail: trapdoor (accepts shell commands)
- **Spread**
  - Bootstrap loader to target machine, then fetch
  - rest of code (password authenticated)

When the Internet worm identified the next target to infect, it looked for several security flaws that it knew how to exploit. These include systems with guessable passwords, systems running the fingerd program that had a buffer overflow vulnerability. Systems running a sendmail program that has a trapdoor, which means that by supplying some special input commands, one can gain access to these systems.

When the security flaws were exploited, the Internet worm can gain access to a target system. It then will load a small piece of code called the bootstrap loader on to a target machine and this loader will then fetch the rest of the worm code. It even used password based authentication to make sure that only the bootstrap loader of the worm can load the rest of the code of the worm.

The Internet Worm

- **Remain un-discoverable**
  - Load code in memory, encrypt, remove file
  - Periodically changed name and process ID
- **What we learned:**
  - Security scanning and patching
  - Computer Emergency Response Team

The Internet worm also employed a number of tricks to hide itself. For example, the worm code is loaded into the memory. It is encrypted and decrypted when necessary and the original file is removed from the hard drive, so that the user will not be able to see the worm program.

The worm even periodically change its process name and process ID, so that even when a system admin looks at what programs are running on the server, he cannot easily discover the Internet worm. The Internet worm resulted in major deception to the Internet, because many servers were infected and had to be disconnected from the Internet.

So what lessons did we learn? The first lesson we learned was that we need to perform security scanning and patching. The Internet worm was able to infect so many servers, because these servers had security flaws. Further, most of these flaws were not only well-known, but also had security patches or fixes available. Therefore, if we scan and patch computers on the Internet that have security flaws, then we can reduce the chances that they will be infected by malware.

The second lesson is that we need to have a fast and coordinated response to a major security incident such as the Internet worm. And because of the Internet worm, the US government established the

computer emergency response team or CERT for short. Nowadays, CERT is usually responsible for issuing alerts about security for flaws and recommendations about patches.

Worm Quiz

Which of the following methods can be used to spread a worm? Check all that apply:

- [ ] email
- [ ] instant messaging
- [ ] downloading files
- [ ] watching a video on netflix
- [ ] clicking on a popup
- [ ] using facebook

Which of the following methods can be used to spread a worm? Here, we list a number of methods, email, instant messaging, downloading files, watch a video on Netflix, clicking on a popup, or using Facebook.

All of these methods can be used to spread a worm. Some of these are obvious, for example email, instant messaging, and downloading files, and others may require some analysis, for example, watching a video on Netflix. A video may contain instructions or executables where the worm can be embedded in. Likewise, clicking a popup. A popup may contain malicious scripts that itself can be a worm. When you use Facebook there are a number of active contents, meaning those are executable programs or scripts that can be worm.

- [x] email
- [x] instant messaging
- [x] downloading files
- [x] watching a video on netflix
- [x] clicking on a popup
- [x] using facebook

Malware Prevention & Detection Approaches

- **Prevention:** Limit contact to outside world
- **Detection and Identification**
- **Removal**

Now, let's look at the counter measures.

The first is prevention. For example, we can limit the content of a computer to the untrusted outside world. Meaning that it would not accept documents or programs, or any active contents from other computers. And of course, this will impose major inconvenience to a computer user.

The second approach is detection. This means that we use a monitor to watch out for telltale signs of malware infection.

The third approach is removal, meaning that once we detect that there's malware infection, we will remove the malware and perhaps we should also patch the system.

Malware Prevention & Detection Approaches

4 Generations of antivirus software:

- Simple scanners: Use "signatures" of known viruses
- Heuristic scanners: Integrity checking: checksum, encrypted has
- Activity traps
- Full-featured analysis: Host-based, network-based, sandboxing-based

Given that prevention severely hampers productivity, detections really the main counter measure that we can use, and there are four generations of antivirus software, or malware detection software.

The first is malware scanners. These scanners use signatures or patterns of known viruses to scan program files to find matches. And if there's a match, that means that this program file has been infected by a known virus. One of the examples of signatures of viruses. A signature of a virus is typically the unique sequence of instructions of the virus code or the unique infection marker that the virus would use. These simple scanners are not effective against polymorphic viruses. And the reason is that for polymorphic virus each instance is encrypted with randomly generated key, such that there's no unique signature across all instances of the same virus.

The second is the heuristic scanners, they are based on possible effects of infection. For example, if a program file has been infected with a virus code the checksum of the original file will have changed because new contents has been added to the file. However, this approach can be defeated if the malware deliberately makes sure that the checksum after infection remains the same. For example, the malware can include some additional bytes at the end of the file to make sure that the checksum remains the same as the file before its infection.

The third is activity traps. These detectors look for particular kind of activities that malware will typically perform on a system. Such as modifying the Windows registry file, or reading the password file and sending it to the Internet, and so on. These detectors are based on our knowledge of malware activities, therefore, these detectors are not effective against malware that performs new kinds of malicious activities.

The fourth is the so called, full feature analysis, which is the state of the art. It typically involves multiple approaches. For example, it typically includes host-based monitoring. That in turn includes activity traps and scanners. It also includes network-based monitors that analyze traffic to the Internet. For example, if there's a traffic that contains password file to Internet server, that can be a telltale sign that a malware has infected an host, and is attempting to steal the password file. Similarly, if there's a connection to a website that is well-known for a malware download, there's also a telltale sign that the end host has been infected. And the malware on that host is attempting to download an update. And you can also include a sandboxing-based analysis approach. A sandbox is typically used to run a piece of executable, for example, an attachment from an e-mail, to see whether this executable would exhibit any malicious activities. By executing this executable in a sandbox, we can make sure that there's no permanent damage to our system and network. And we can observe the behaviors of the executable from outside a sandbox. So that we can be certain whether this executable is a malware or not.

Malware Prevention & Detection Quiz

Given that **signature-based anti-virus solutions** are not always effective, **why do we still use them**?
Check all that apply:

- ☐ **they are very efficient**
- ☐ **effective against known malware good**
- ☐ **"first-line" defense**

Given what we've said about the limitation of signature scanners, why do we still use them?

Is it because they are very efficient? Yes. They are efficient because signature matching or matching can be very efficient. Is it because they are effective against known malware? The answer is yes. Known malware means that a malware that matches a signature. And of course, signature based approach is effective against these malware that have signatures already. Is a good first line defense, yes, because with this approach at least we can detect all the known malware already. Therefore it is a good first line defense.

- [T] they are very efficient
- [T] effective against known malware
- [T] good "first-line" defense

The Most Expensive Worm Quiz

Which of the worms described below caused the **greatest financial damage**?

- ○ ILOVEYOU: Sent by email with the subject "ILOVEYOU". It had an attachment that, when executed, deleted all files on the host computer.
- ○ CODE RED: a worm that took advantage of a buffer overflow vulnerability in Microsoft servers. Infected machines would launch 'denial of service' on IP addresses.
- ○ Morris Worm: 99 lines of code that Robert Morris, a Cornell student launched to find out the size of the internet.

Instructor Notes

Top 10 Worst Computer Worms

Now let's do a fun quiz on worms. Which of the following has caused the greatest financial damage? Is it the ILOVEYOU worm? That as an email attachment gets sent around, and when it is triggered and executed, it will delete all the files on the host computer. Or CODE RED which is a worm that infects many Microsoft servers. Or the Morris Worm or the Internet worm that we discussed.

The correct answer as I hinted, is the ILOVEYOU worm. The estimate cost is above $2 Billion. CODE RED is somewhere around $1 Billion and Morris Worm is around $100 Million.

- [⊗] ILOVEYOU: Sent by email with the subject "ILOVEYOU". It had an attachment that, when executed, deleted all files on the host computer.
- ○ CODE RED: a worm that took advantage of a buffer overflow vulnerability in Microsoft servers. Infected machines would launch 'denial of service' on IP addresses.
- ○ Morris Worm: 99 lines of code that Robert Morris, a Cornell student launched to find out the size of the internet.

Malicious Code
Lesson Summary

Host-dependent malware:

- trap doors
- logic bombs
- trojan horses and
- viruses

Host-independent malware:

- Worms

Some malware required host programs. These include Trojan Horses, trap doors, logic bombs, and viruses. Other malware can run as independent programs. For example, the Internet worms.