Malware – set of instructions that cause a site’s security policy to be violated.

1. **#! /bin/sh**:
   * This is the shebang (or hashbang) line, which is always the first line in a script. It tells the system which interpreter to use to execute the script. In this case, **/bin/sh** is the path to the shell interpreter. It means that this script should be run using the Bourne shell (**sh**), which is a common command interpreter in Unix-like systems.
2. **cp /bin/sh /tmp/.xxsh**:
   * **cp** is the copy command.
   * **/bin/sh** is the source file, which is the shell interpreter.
   * **/tmp/.xxsh** is the destination file. This is where **/bin/sh** is being copied to.
   * The destination file name, **.xxsh**, is somewhat concealed by starting with a dot (**.**), which in Unix-like systems means it will be hidden from a basic directory listing.
   * This command essentially creates a copy of the shell interpreter in the **/tmp** directory, which is a temporary directory that all users typically have access to.
3. **chmod o+s ,w+x /tmp/.xxsh**:
   * **chmod** is the command used to change the file mode bits (permissions) of a file or directory.
   * **o+s** sets the "setuid" bit for others (all users). This means that anyone executing the **/tmp/.xxsh** file will have the permissions of the file owner (which, in this case, is likely to be the root user, since **/bin/sh** is usually owned by root).
   * **w+x** sets the write and execute permissions for others.
   * **/tmp/.xxsh** is the file whose permissions are being changed.
   * In summary, this command changes the permissions of **/tmp/.xxsh** so that any user can write to and execute the file, and when they execute it, they will do so with the permissions of the owner of the file (likely root).

In summary, **ls $\*** in a shell script is a way to pass all the script's arguments to the **ls** command, enabling dynamic listing of directories or files based on user input when the script is executed.

Creating a copy of a shell and setting its UID (User ID) to that of the user executing the program is a technique often used in system administration and cybersecurity, particularly for privilege escalation. Here's what it means:

1. **Shell**: A shell in the context of operating systems is a program that provides an interface for users to interact with the operating system. Common shells in Unix-like systems are **bash**, **sh**, and **zsh**.
2. **Creating a Copy of a Shell**: This step involves making a duplicate of the shell executable file. This is typically done using commands like **cp** (copy) in Unix-like systems.
3. **SetUID (Set User ID)**: SetUID is a special kind of permission in Unix-like operating systems. When a file (in this case, the shell) has SetUID permission set, it runs with the permissions of the file owner, rather than the user who is running it.
4. **Setting to User Executing the Program**: This phrase suggests changing the ownership of the copied shell to the user who is executing the program. However, in typical use of SetUID for privilege escalation, the copied shell's ownership is set to a superuser (like root) instead. When a non-privileged user executes this SetUID-enabled shell, it will run with superuser privileges, allowing the user to perform actions they normally couldn't.
5. **Context**: This method is often used in situations where a user needs to temporarily gain higher privileges than they are normally granted. However, it's important to note that using SetUID in this way can pose significant security risks if not managed properly, as it can be exploited by unauthorized users to gain elevated privileges.

In summary, creating a copy of a shell and setting its UID to the user executing the program generally refers to a process where a duplicate of the shell is made, and its permissions are altered to allow it to run with the privileges of a different user, often for the purpose of temporarily elevating privileges within the system. This should be done with caution due to the associated security implications.

System can’t determine if instructions executed by a process are known to user or are set of instructions user does not intend to execute.

Trojan horse – program with overt(documented or known) purpose and covert

( undocumented) purpose.

In preceding ex overt purpose – list files in directory.

Covert purpose – create shell setuid to user executing script

Anyone executing file will have permission of file owner.

Rootkit – Is a pernicious(highly harmful or destructive) trojan horse .

Hides itself on a system so it can carry out its actions without detection.

Earliest rootkits installed backdoors and other traps at various places in the system

Rootkit then changes system programs that reported on status of system and its components.

Program that list contents of directory altered to not report presence of certain files.

Approach to counter rootkits run nonstandard programs that obtained same information as system programs.

These programs bypassed system programs ,using system calls and information from kernel to obtain required information.

Later rootkits became more sophisticated and altered parts of the kernel changing modules loaded onto it so any program accessing kernel would get information filtered by the rootkit.

Propagating (replicating) trojan horse – trojan horse that creates a copy of itself

Some trojan horse can make copies of themselves. One of the earliest trojan horse was a version of the game Animal. When the game was played, It created a copy of itself.

These copies spread, taking up much room.

The program was modified to delete one copy of the earlier version and create 2 copies of the modified program.

Virus

When the trojan horse can propagate freely and insert a copy of itself into another file , it becomes a computer virus.

A computer virus is a program that inserts a possibly transformed version of itself into one or more files and then performs some(possibly null) action.

First phase (insertion phase)– virus inserts itself into a file

Second phase(it performs some action) – called execution phase

Viruses can infect systems in 3 ways, through a boot sector ,executables or data.

Boot sector virus is a virus that inserts itself into boot sector of disk.

When system boots, any virus in sector executed.

Executable infector – virus that infects executable programs.

Multipartite virus – infects both boot sectors and applications.

Macro virus – Composed of sequence of instructions that is interpreted ,rather than executed directly. They can execute on any system that can interpret the instructions.

Ex - A spreadsheet virus executes when spreadsheet interprets the instructions

A macro virus can infect executables or data files

Goal of virus remain undiscovered until executed, possibly even after that.

As detection methods became more sophisticated, so did methods of concealment.

Stealth viruses are viruses that conceal infection of files.

The virus intercepts calls to the OS that access files to present appearance of non-infected file.

Polymorphic virus - virus that changes form of its decryption routine each time it inserts itself into another program.

Metamorphic virus – virus that changes its internal structure but performs same actions each time it is executed.

Worm – program that copies itself from one computer to another

Worm has 3 phases

First – target selection. Occurs when worm determines what systems to attempt to spread to.

Then propagation begins ,worm attempts to infect set of chosen targets.

After this worm enters execution phase once it is resident on target.

Last execution phase may be empty(worm is simply spreading)

Target selection and propagation phase – infection stage of virus

Sometimes attackers co-ordinate actions among malware on different systems.

The malware can preset specific actions to occur with specific triggers.

Attacker can have malware take action based on messages that the attacker sends.

Bot – Malware that carries out some action in co-ordination to other bots.

The attacker called botmaster controls bots from one or more systems called command and control(C and C) ,servers and motherships .They communicate over paths called C and C channels. A collection of bots is a botnet.

Distinguishing characteristic of bot is its using of C and C channel.

The bot can be updated or triggered through this channel.

Each bot has 4 stages in its life cycle.

1.Bot infects a system. Can be done in any number of ways, for example as a computer worm or trojan horse resident in a program an unsuspecting user installs and executes.

2.Bot then checks for a network connection and looks for either a C and C server or another node it can communicate with.

3.Bot then is given commands to execute by C and C server or other node. This may also involve downloading additional components to the bot to add to its capabilities.

4.The bot then executes the commands. If appropriate ,it sends the results to another site.

Botnet can be organized in 3 basic ways.

Botnet may be centralized ->each bot communicates directly with C and C server. However, the C and C server will become a bottleneck for large botnets.

Thus many botnets use a hierarchical control scheme in which C and C server communicates with a set of bots that are C and C servers for other bots.

C and C server -> set of bots (C and C servers)-> set of other bots

Allows control over large botnet.

Peer to peer botnets use a C and C structure in which there is no single C and C server. A peer to peer network is constructed with bots acting as peers.

If some portion of botnet is deleted ,remainder of botnet can function.

Third organizational scheme has high latency. When a bot wants to communicate with another node, it scans addresses at random till it finds another bot. It then forwards message to that bot.

Such organization minimize damage if bot discovered, as bot can only lead to another bot.

One problem with botnets is address of C and C servers must be available to bots, if redundant other C and C servers.

A bacterium – program that absorbs all of some class of resource.

Logic bomb – Performs action that violates security policy when some external event occurs.

Adware

Displays advertisements on systems. It may be benign, if user consents to it being present ,and understands exactly what it does.

This is often not the case, so adware considered as malware.

Adware - Trojan horse that gathers information for marketing purposes and displays advertisements often based on gathered information.

Covert purpose – gather information about user

Spyware

Gathers information about a user, system or other entity and transmits it or stores it for later retrieval.

Unlike adware ,its presence is invisible to the user. Hence it is truly covert .Hence it is malware.

Spyware is a trojan horse that records information about the use of computer ,typically confidential information like keystrokes, passwords, credit card numbers and visits to websites.

Information may be transmitted to a third party, stored for later transmission.

Enters system through vulnerabilities ,programs ,applications or websites user visits.

Ransomware – malware that inhibits use of resources until a ransom ,usually monetary is paid.

Phishing – act of impersonating a legitimate entity ,typically a website associated with a business , in order to obtain information such as passwords ,credit card numbers and other private information

without authorization.

Ransomware – malware that inhibits use of resources until ransom is paid.

Malware and virus are different

Virus has a program as its host.

Worm has a computer system as its host.

Virus has to start with initiation of a program

Worm actively looking for new hosts. Worms more effective at spreading quickly before we have a chance to discover and remedy problems.

Trapdoor – Means of subverting normal access control and authentication of a system. Means

of bypassing complete mediation.

A rootkit is a pernicious Trojan horse. It hides itself on a system so it can carry out its actions without detection. The earliest rootkits installed backdoors and traps at various places in the system.

The rootkits changes various system programs that report on the status of system and its components.

Ex – a program that lists contents of directory altered to not show specific files.

A network status program altered so it will not show network connections from specific hosts.

Approach to counter rootkits – run nonstandard programs that obtain same information as system programs. For example, a program might access the directory directly and read its contents, which would be a list of files and other information.

Later rootkits altered parts of kernel so any program accessing kernel to retrieve information would get information filtered by rootkit.

Should be called anti- malware not anti-virus.

They are addressing all types of malware.

Stealth virus – virus that conceals infection of files.

Polymorphic malware – avoid being detected by systems that know what malware looks like.

One stealth strategy – change what things look like.

Program doesn’t have to look same to do same thing.

Polymorphic viruses – program themselves contain strategies to rearrange instructions

Can do a simple encryption.

Covert channels

Covert channels refer to methods used to communicate information secretly, typically within a computer system, in a manner that violates security policies or breaches security mechanisms. These channels are not designed for communication but are exploited to transmit data surreptitiously. They are particularly relevant in the context of computer security, where they can be used to bypass security controls, leak sensitive information, or signal control commands to malicious software. Here are some key aspects of covert channels

1. **Types of Covert Channels**:
   * **Storage Covert Channels**: These involve the indirect communication of information through a storage location. For example, one process writes data to a disk file, and another process reads it, even though direct communication between these processes is blocked by security controls.
   * **Timing Covert Channels**: These channels rely on the timing of events to transmit information. One process might signal information to another by modulating its use of system resources, like CPU time, which the second process can observe and decode.
2. **Use in Malware and Attacks**: Covert channels are often used by malware to exfiltrate data or receive commands from an attacker. This can be done without triggering network security mechanisms like firewalls or intrusion detection systems.
3. **Detection Challenges**: Detecting covert channels is challenging because they often use legitimate system functions and processes. Since they don’t rely on typical communication methods, they can bypass standard monitoring and detection tools.
4. **Example Scenarios**:
   * A malware program that communicates with its command and control center using a covert channel, such as by modulating the timing of legitimate network traffic.
   * An insider threat where an employee uses a covert channel to leak sensitive data from a secure environment to an external recipient.
5. **Mitigation**: Mitigating the risk of covert channels often involves:
   * Rigorous system design and architecture that minimizes unnecessary shared resources and ensures strong separation of processes.
   * Monitoring and analyzing patterns of system usage to detect anomalies that could indicate covert communications.
   * Implementing strict access controls and minimizing the privileges of each system component to reduce the potential for unauthorized information flow.

In summary, covert channels are a sophisticated means of circumventing security mechanisms to communicate information secretly. They are a concern in high-security environments and require advanced methods for detection and prevention.

Stealth