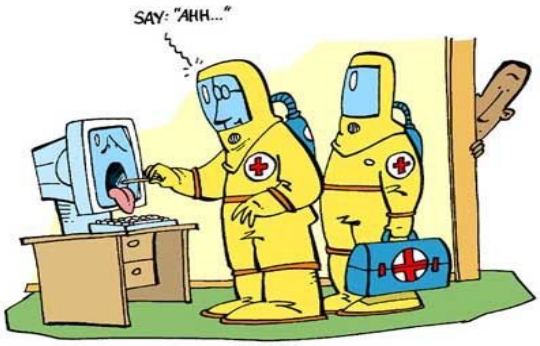
CS458/CS558 Introduction to Computer Security

Chapter 19 Malicious Software



Malicious Programs

- Malicious software: software that is intentionally included or inserted in a system for a harmful purpose.

Malicious Programs

- Malicious software: software that is intentionally included or inserted in a system for a harmful purpose.
  - Virus: a piece of software that can infect other programs by modifying them
    - The modification includes a copy of the virus program, which can then go on infect other programs

Malicious Programs

- Malicious software: software that is intentionally included or inserted in a system for a harmful purpose.
  - Virus: a piece of software that can infect other programs by modifying them
    - The modification includes a copy of the virus program, which can then go on infect other programs
  - Worm: a program that can replicate itself and send copies from computer to computer across network connections.

Malicious Programs

- Malicious software: software that is intentionally included or inserted in a system for a harmful purpose.
  - Virus: a piece of software that can infect other programs by modifying them
    - The modification includes a copy of the virus program, which can then infect other programs
  - Worm: a program that can replicate itself and send copies from computer to computer across network connections.
  - Trojan horse: program that contains unexpected additional functionality

Malicious Programs

- Malicious software: software that is intentionally included or inserted in a system for a harmful purpose.
  - Backdoor: bypass normal authentication without being detected.
  - Flooder: perform denial of service attack.
  - Rootkit:
    - Designed to take control of a computer system, without authorization by the system's owners.
    - Used by attackers to hide their actions from system administrators.

Malicious Software: Two Categories

- Software that need a host program: fragments of programs that cannot exist independently of some actual application program, system program.
- Software that are independent: self-contained programs that can be scheduled and run by the operating systems.

Malicious Software: Two Categories

- Software that need a host program: fragments of programs that cannot exist independently of some actual application program, system program.
  - E.g. Trojan horses, Virus
- Software that are independent: self-contained programs that can be scheduled and run by the operating systems.
  - E.g. Worms

Viruses

- A piece of software that can infect other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.
- Carries code to make copies of itself as well as code to perform some covert task

Virus Operation

- Virus phases:
  - Dormant – waiting on triggering event, such as a date, the presence of another program.
    - E.g. Friday the 13th
  - Propagation – places an identical copy of itself into other programs. Each infected program will now contain a clone of the virus
  - Triggering – the virus is activated to perform the function for which it was intended.
  - Execution – the function is performed. The function may be harmless or damaging.
- Usually machine/OS specific – designed to take advantage of the weaknesses of particular systems.

Virus Structure

- The infected program, when invoked, will first execute the virus code and then execute the original code of the program

Virus Structure

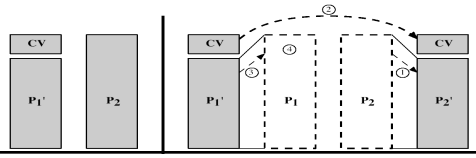
```
program V :=
    {goto main;      //jump to the main virus program
    1234567;         // a special marker that is used to determine whether
                     // or not a potential victim program has already been
                     // infected with this virus
    subroutine infect-executable :=      {loop:
                  file := get-random-executable-file;
                  if (first-line-of-file = 1234567) then goto loop
                  else prepend V to file; }
    subroutine do-damage :=  {whatever damage is to be done}
    subroutine trigger-pulled := {return true if condition holds}
    main: main-program :=      {infect-executable;
                                if trigger-pulled then do-damage;
                                goto next;}  // tranfer control to original
                                             //program
    next:
}
```

Virus Structure (Cont.)

- The virus described in the previous slide is easily detected because an infected version of a program is longer than the uninfected one.

Compression Virus

- The virus described in the previous slide is easily detected because an infected version of a program is longer than the uninfected one.
- Compression virus
  - Compress the program and attach a copy of the virus to the compressed program – make it the same size as the uninfected one.
  - Uncompress the infected program and execute.



Types of Viruses

- Can classify on basis of how they attack
  - Parasitic virus: traditional and most common form of virus.
  - Memory-resident virus: lodges in main memory as part of a resident system program. Infects every program that executes.
  - Polymorphic virus: mutates with every infection, making detection by the signature of the virus impossible
    - Infect files with an encrypted copy of itself, which is decoded by a decryption module.
    - A random key is created for every infection.

Types of Viruses

- Can classify on basis of how they attack
  - Boot sector virus: infects a master boot record or boot record and spreads when a system is booted from the disk containing virus
  - Stealth: explicitly designed to hide itself from detection by antivirus software.
    - E.g. compression virus
  - Metamorphic virus: mutates with every infection.
    - Avoid being detected by emulation
    - Rewrites itself completely at each iteration, increasing the difficult of detection.
    - May change their behavior as well as their appearance.

Macro Virus

- Macro virus
  - Takes advantage of macro (an executable program embedded in a word processing document) in Word and other office application.
  - Platform independent: any hardware platform and operating system that supports word can be infected
  - Infects documents, not executable portions of code.
  - Easily spread – e.g. by electronic mail.
  - Are no longer dominant virus threat – word provides increased protection against macro viruses.

Email Virus

- Spread using email with attachment.
  - E.g. Melissa: containing a macro virus
- Triggered when user opens attachment or worse – triggered by opening an email rather than opening an attachment.
- Propagate very quickly
- Usually targeted at Microsoft Outlook mail agent & Word/Excel documents

Virus Countermeasures

- Best countermeasure is prevention — not possible.
- Hence need to do one or more of:
  - Detection – once the infection has occurred, determine that it has occurred and locate the virus.
  - Identification – Once detection has been achieved, identify the specific virus that has infected a program.
  - Removal – Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state.
  - If detection succeeds but either identification or removal is not possible, then discard the infected program and reload a clean backup version.

Anti-Virus Software - Scanner

- First-generation
  - Uses virus signature to identify virus
  - Maintains a record of the length of programs and looks for change in length
- Second-generation (Integrity checking)
  - A checksum is appended to each program
  - If virus infects the program without changing the checksum, then integrity check can detect
  - Some virus may be able to change the checksum – we can use an encrypted hash function.
  - The encryption key is stored separately from the program. So the virus cannot generate a new hash code and encrypt that.

Advanced Anti-Virus Techniques

- Generic decryption
  - When a file containing a polymorphic virus is executed, the virus must decrypt itself to activate
  - To detect such a structure, executable files are run through a GD scanner
    - Use Virus signature scanner to check program signature & behavior before actually running it
    - Instructions in an executable file are interpreted by the CPU emulator (a software-based virtual computer) rather than executed on the underlying processor.
    - If the code includes a decryption routine, the control module interrupts interpretation to scan the target code for virus signatures.

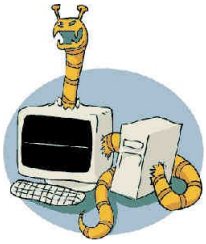
Trojan Horse

- Program with hidden side-effects
- Which is usually superficially attractive
  - E.g., games.
- When run, performs some additional tasks: allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor, or simply to destroy data

Trojan Horse

- Example: A user wants to gain access to the files of another user on a shared system
  - Create a Trojan horse program that, when executed, changed the invoking user's file permissions s.t. the files are readable by any user
  - Induce another user to run the program

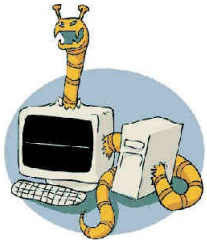
Worms



- A program that can replicate itself and copies from computer to computer across network connections.
- Email virus vs worms

http://www.topnews.in/technology-update/internet-security

Worms



- A program that can replicate itself and copies from computer to computer across network connections.
- Email virus vs worms
  - Email virus: requires a human to move it forward
  - Worms: actively seeks out more machines to infect and each infected serves as an automated launching pad for attacks on other machines.

Worm Operation

- Worm phases like those of viruses:
  - Dormant
  - Propagation
    - Search for other systems to infect
    - Establish connection to target remote system
    - Replicate self onto remote system
  - Triggering
  - Execution

Morris Worm – The Best Known Classic Worm

- Was designed to spread on Unix systems
- For each discovered host, the worm tries a number of methods for gaining access:
  - It attempted to log on to a remote host as a legitimate user – first attempted to crack the local passwords and corresponding users ID.
    - Assume that many users would use the same password on different systems
    - Exploit bug in finger daemon
- If any attack succeeds, the worm replicates itself and executes.

Backdoor

- Secret entry point into a program that allows someone who knows access bypassing usual security procedures
- Programmers have used backdoors legitimately for many years to debug and test programs.
  - Avoid authentication procedure or a long setup
  - If something is wrong with the authentication procedure, then have some method of activating the program
- A threat when left in production programs allowing exploited by attackers

Rootkits

- Typically used in the very first steps an attacker takes once he or she has compromised a system
- Help an attacker keep a previously obtained root access.
- Used by attackers to hide their actions from system administrators.
- Types
  - User mode rootkits
  - Kernel mode rootkits

User Mode Rootkits

- Focus on replacing specific system programs commonly used
- The most primitive user-mode rootkits, are dated in 1989.
  - Manipulate the system logs
    - wtmp - keeps track of all logins and logouts to the system
    - Lastlog - formats and prints the contents of the last login log
  - The attacker cannot be identified by commands: w, who, last

User Mode Rootkits (Cont.)

- A list of the typical files substituted by user-mode rootkits
  - Hide files: ls, df
  - Hide processes: ps, top
  - Hide connections: netstat, tcpd
  - Hide logs: syslogd
  - Hide logins: w, who, last
  - Backdoor: login, rlogin, sshd

Kernel Rootkit

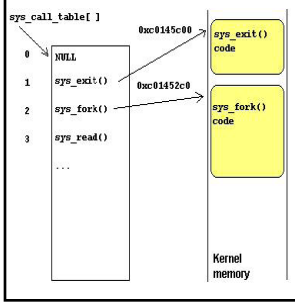
- Alter the kernel
  - Provide all the user-mode rootkit features
  - In addition, enable the redirection of any program execution
- The first rootkit focused on tampering the kernel appeared in 1997

Typical Actions of Kernel Rootkit

- Some of the most typical actions a kernel rootkit can perform.
  - Execution redirection:
    - It intercepts the sys_execve() syscall to execute a different program.
    - E.g. when someone launches "/bin/bash", the standard Linux shell binary, the rootkit intercepts its execution and launches "/bin/evilbash" instead.

Typical Actions of Kernel Rootkit

- Some of actions a kernel rootkit can perform.
  - File/Process hiding
    - Instead of replacing the ls or ps program, it can be implemented manipulating the system calls that access files, e.g. modify the system call table
    - Hide ls: substitute sys_getdents(), which is used to get directory information
    - Hide file content: Intercept sys_open()

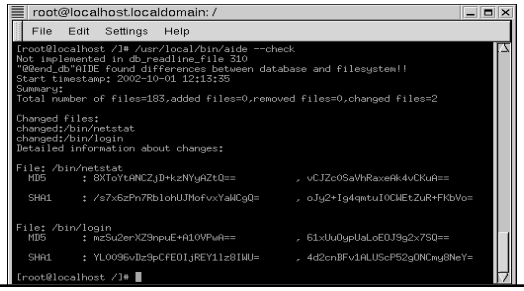


Typical Actions of Kernel Rootkit (Cont)

- Some of the most typical actions a kernel rootkit can perform.
  - File/Process hiding
    - Hide processes: manipulate the sys_getdents() in order not to show some process entries inside the "/proc" pseudo-file system. Through the sys_fork() and sys_clone() syscalls, new spawned child processes can be hidden too.

Rootkit Detection Tools

- AIDE
  - Detect rootkits based on the checksums of binary files
  - Detected that the netstat and login files have been changed by looking at their checksums



Rootkit Detection Tools

- chkrootkit – http://www.chkrootkit.org/
  - A script file that can be used to detect the presence of rootkits based on certain signatures, e.g. strings that are present in the existing rootkits
  - looks for files or file changes created by well-known rootkits