HW #2 Primer

ECE 650
Systems Programming & Engineering
Duke University, Spring 2016

Rootkits

- Malware that acquires privileged access to the OS
  - Also maintains that access
  - By hiding its presence from normal OS activity
- Goals of a rootkit
  - Run (without restriction) on a target system
    - Use social engineering or vulnerabilities in protection (e.g. ACLs)
  - Remain invisible to security software, OS, users
  - Perform malicious action (called the payload)
    - Steal information or access to resources; install other malware

What's in this Assignment

- You will be writing part of a rootkit
- Assume that part #1 has been accomplished
  - A vulnerability has been exploited (e.g. buffer overflow)
  - In a privileged operation
  - Now you have ability to run malware code w/ root privilege
- You will implement aspects of part #2
  - How do you keep the malware code & process hidden?
    - From suspicious users
    - From security software
  - High level
    - Create a program that will run with root privilege
      - E.g. 'sudo ./my_program.exe
    - Program will load & unload a *kernel module*

High-Level Tasks & Learning

- Create a *loadable kernel module* (LKM)
  - A way to add (temporary) extensions to kernel support
    - E.g. new device drivers, new system calls, new kernel support fcns
  - Object file with code to extend the kernel
  - Makefile will be provided
  - Linux example:
    - sudo insmod my_module.ko
    - lsmod
    - sudo rmmod my_module.ko
- Create a program that will run w/ root privilege
  - E.g. 'sudo ./my_program.exe'
  - Program will load & unload a *kernel module*
  - Practice with fork() & exec*() to create child processes

Your Kernel Module

- Goal
  - Hide the existence of your program & process
  - In several ways that will be described in the assignment
    - E.g. hide program file from view (ls)
    - E.g. hide process ID from view (ps, top, ls /proc)
    - E.g. hide malicious kernel module from view (lsmod)
- How can this be done?
  - What do the above commands have in common?
    - Utilize system calls to inspect system state
    - Try it out!
      - strace <command>
  - Commands use many system calls, only some do the key work
  - Your kernel mod will change system call table to point to your new "sneaky" functions
  - Your "sneaky" functions will need to alter behavior of default system calls

Kernel Module Implementation

- Let's walk through a demo that illustrates key points

Assignment Admin Stuff

- You will need (want!) to use the same Linux image
  - https://vm-manage.oit.duke.edu/vm_manage
  - vcl-ubuntu14-generic
- Gives you a dedicated VM w/ root access
- You can install packages easily on the new image
  - sudo apt-get install <package_name>
  - E.g. sudo apt-get install emacs
- From web interface you can power-off, power-on VM
  - May need to do this occasionally as you develop code
  - Things may get into an unrecoverable (easily) state

Ethical Hacking

- As systems programmers, it is important to…
  - Understand the types of exploits that exist
  - Understand the unintended or malicious ways that kernel facilities can alter the machine and its software
  - Deeply understand how systems calls work
- Purpose is not to teach recipes for conducting malicious exploits and creating malware