Never Sleep Again: Introduction to Rootkit Design

Michael Grube
@michaelgrube
github.com/mgrube

Rootkits

- Code designed to persist for as long as possible
- Main task is to stay hidden
- Can live in the kernel
  - Injection
  - Kernel Modules
- Can live in Userland:
  - Windows Registry
  - Binary Infection
- Can live in BIOS/Firmware
  - UEFI
  - Other

Main Tasks

1. Get Root
2. Get Persistence
3. Hook functionality
4. Stay hidden
5. Profit!

Userland vs Kernel Mode

- Userland is typically easier to infect and detect
- Kernel mode is desirable but difficult to code for
  - Kernel panic is kind of a giveaway!
    - Except on Windows ;)
  - Mac OS Microkernel makes things more of a challenge
    - Pretty much need a kernel 0day
    - “Shellcode” is harder when working in the kernel
    - Code signing
    - Live in EFI?

Kernel Modules

- You win!
- Usually requires an exploit
- Need to be stable
- Simpler is better
- Good to occupy userland as well

Hooking

- Yo Dawg, I heard you like functions
- Main method of interception by a kernel module
- Hide all the things!
- Communicate with userland processes

File Infection

- Fun and easy
- PE, ELF, MachO
- Backdoor Factory (https://github.com/secretsquirrel/the-backdoor-factory)
- Retaliation (http://vxheaven.org/lib/vrn01.html)
- Hard on newest OS X
- Easy on Linux ;)

UEFI

- How often do you upgrade your BIOS?
- Necessary for macOS
- Ultimate persistence!
- SMM Editing, Super dangerous

Firmware Rootkits(Click me!)

Demo

Getting started

- Vagrant Box: vagrant init ubuntu/yakkety64;
- Vagrant up
- Fun Fact: Works on most recent version of Ubuntu
- git clone https://github.com/leixiangwu/CSE509-Rootkit
- Assume we can get root!

Hooking a system call

- Must brute force through kernel memory
- Replace pointer to system function with pointer to our function

Exercise: Manipulating execve

- Called when executing a process
- Lots of possibilities
- Pretty easy!

Working on your own

Strace is your friend

strace -o systemcalls.o programname > /dev/null

Find your list of syscalls:

/usr/include/x86_64-linux-gnu/bits/syscall.h

Manpages are your friends

Countermeasures

- Signatures
- Getting there first
- Logging like crazy
- ????

Resources

- ALL:
  - Shellcoder's Manual
  - The Art of Exploitation
  - The Rootkit Arsenal
- Linux Userland Persistence:
  - Retaliation: http://vxheaven.org/src.php?info=retal10.zip
  - LD_PRELOAD
  - Linux Binary Analysis
  - Jynx/Azazel: https://github.com/chokepoint/azazel
  - https://github.com/michalmalik/linux-re-101
- Windows Persistence
  - Registry editing

Resources 2

- OS X:
  - https://github.com/michalmalik/osx-re-101
  - https://github.com/AndrewDryga/vagrant-box-osx
  - https://bugs.chromium.org/p/project-zero/issues/detail?id=1004
  - https://www.blackhat.com/presentations/bh-usa-08/D'Auganno/BH_US_08_DAuganno_iRK_OS_X_Rootkits.pdf
  - https://papers.put.as/macosx/macosx/
- Firmware:
  - Hacking Team EFI Bootkit
  - Cr4sh: https://blog.cr4.sh
- Community:
  - VXHeaven
  - Kernelmode.info

Questions?
thesnark@protonmail.ch

Getting started

- Vagrant Box: https://github.com/AndrewDryga/vagrant-box-osx
- Fun fact: Box is vulnerable to recent kernel exploit
- Clover for development
- Walkthrough BIOS Setup
- Assume we can get root!