ROOTKITS

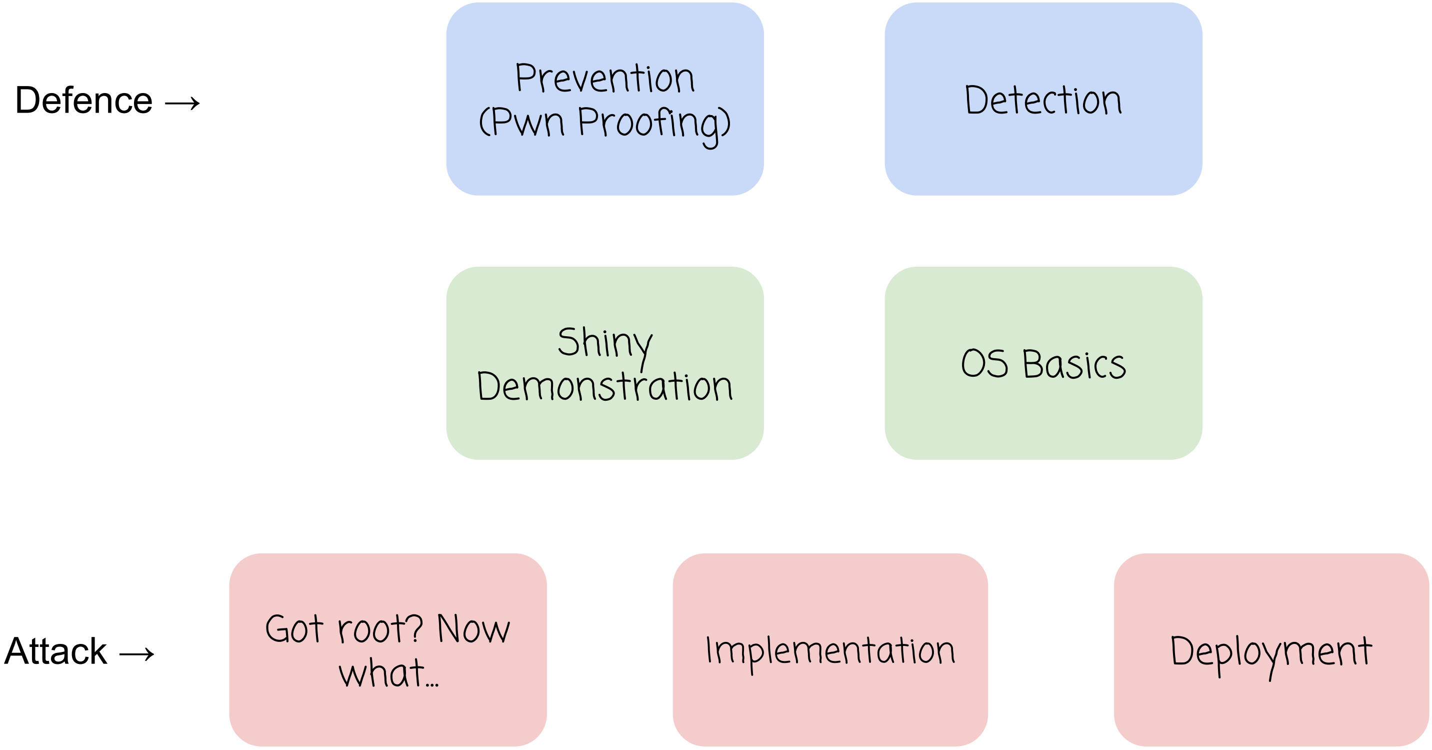
Hiding in Plain Sight

What are Rootkits?

- Software tools that allow unauthorised user to retain control of system
  - Can operate at root level - User Rootkit
  - Boot loader level - Bootkit
  - **Kernel level**
  - Many others... ← Our Focus
- They **actively hide** their own presence and activities
- Rootkits are not in themselves malware
  - Are used to run unauthorised processes without detection from user
  - May or may not be used for malicious purposes

ROOT
KIT
ROOT/ADMIN ACCESS
SET OF TOOLS

Topics Overview



Defence →

Prevention (Pwn Proofing)

Detection

Shiny Demonstration

OS Basics

Attack →

Got root? Now what...

Implementation

Deployment

#include <Demo>

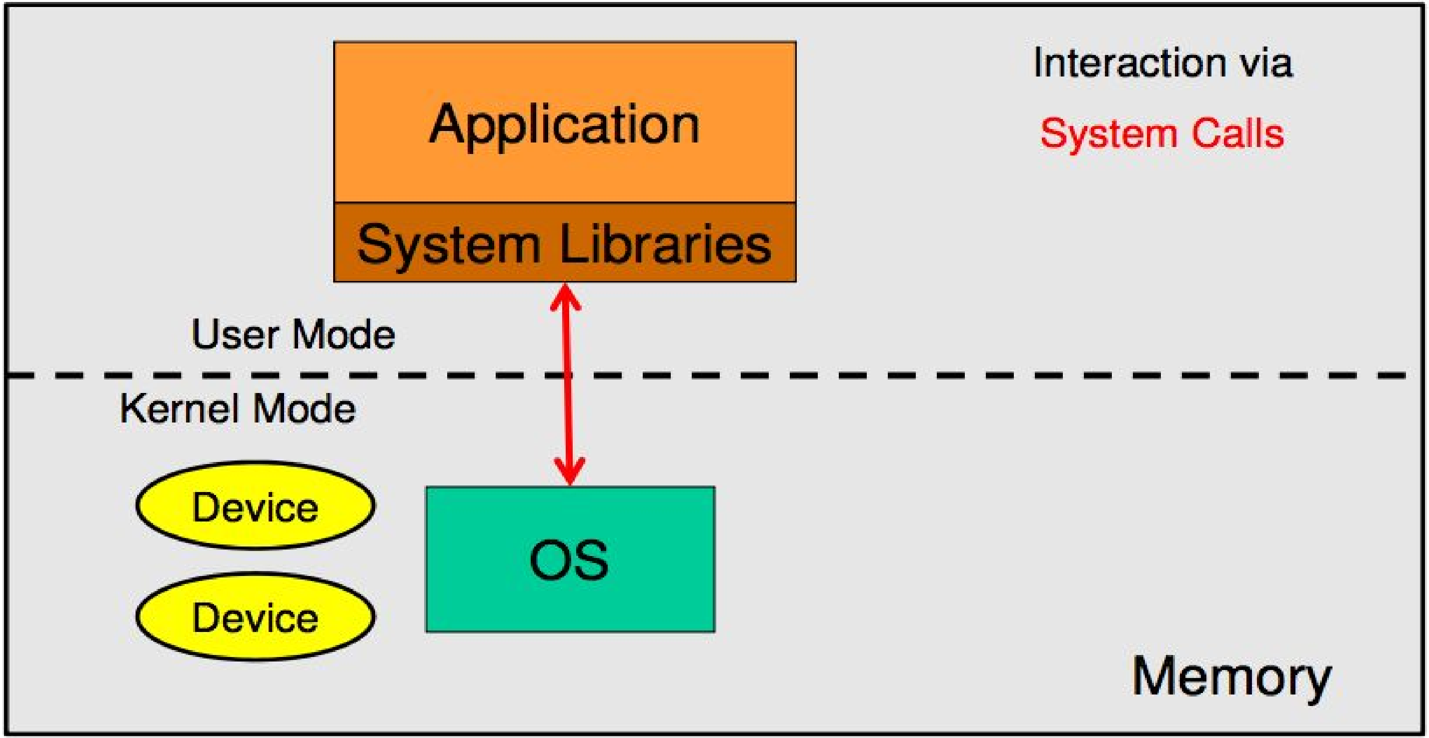
OS Basics

- **General roles of operating system**
  - Extends and hides hardware
  - Manages processes

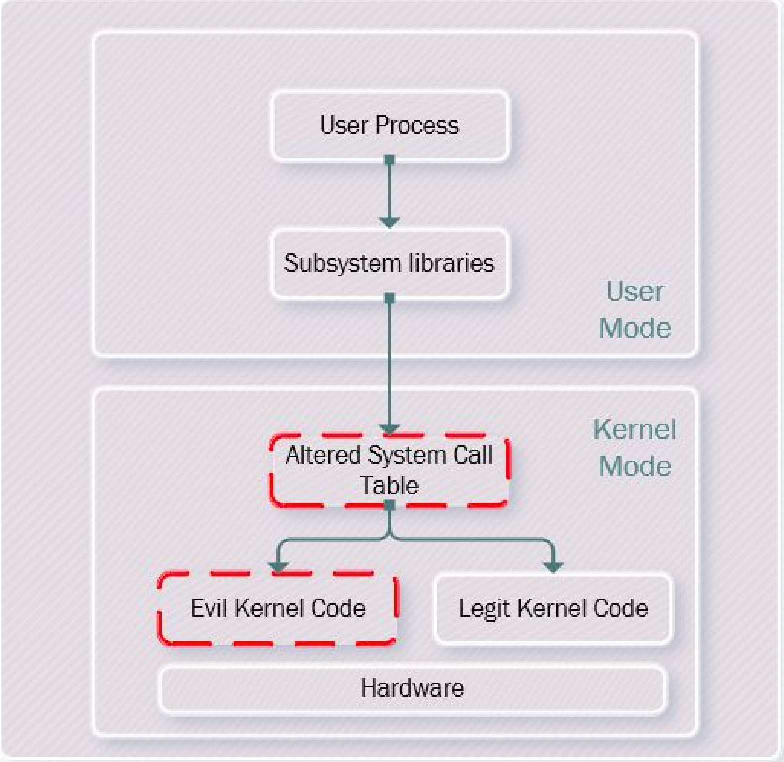
- **Usermode vs Kernelmode**
  - Usermode rootkits will tamper with the application layer
  - Kernelmode rootkits will modify kernel code

- **Syscalls**
  - Application asks the operating system to perform a restricted action for them

A Standard Syscall



A Malicious Syscall



Implementation

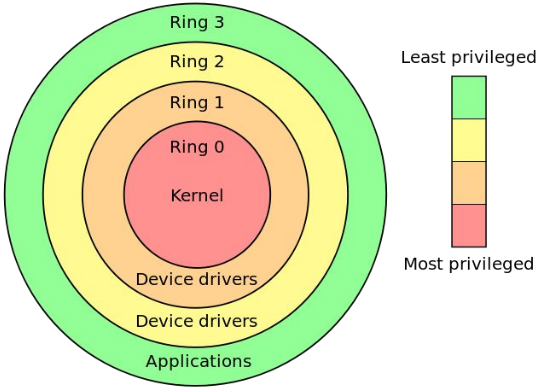
https://github.com/ljcusack/freebsd-rootkit

Rootkit Deployment

- How do rootkits get themselves on a device?
- Attack Vectors
  - Websites with unwanted downloads
  - Suspicious email attachments
  - Legitimate vendor software in rare instances
- Example: Sony’s XCP Rootkit
  - Uncovered in 2005 after spreading for over 1½ years
  - Installed itself through music CDs
  - Spyware to cloak its digital rights management scheme
    - Invasive & makes system more vulnerable to malware
    - Hard to remove, interferes with Registry files
    - ‘Patch’ only makes detectable, doesn’t uninstall

Gaining Root Privileges

- Rootkits first need to gain root or kernel access
- Social engineering
- Privilege Escalation
  - Exploit existing unpatched vulnerabilities
  - Gain progressively higher access
  - Lends itself to large scale attack



Installation

- Once machine is accessed and root privileges acquired…
- ... Rootkit installs self and begins subsystems.
    - Cloaking itself from detection
    - Achieving what the attacker intended to do with the elevated control

Prevention

> *“ In a way… ‘rootkit prevention’ doesn’t make sense… rootkit installation is something that occurs after a system is compromised ”*

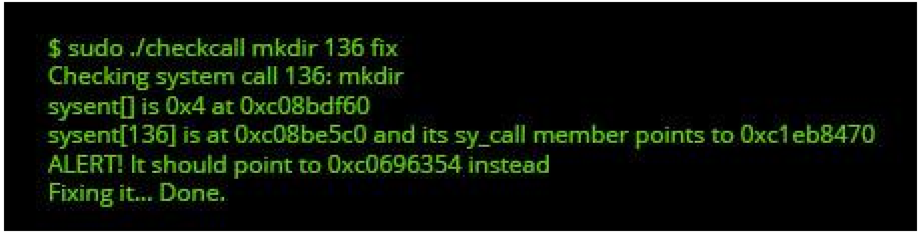
- **Strip Down OS** → minimise attack surface by decluttering applications
- **Keep OS Patches Up to Date** → do any updates ASAP
- **Keep Antivirus Up to Date** → whilst antivirus often struggles to detect rootkits, it’s critical in reducing the attack surface
- **Know Your Device** → be able to identify when things are amiss, even if your computer and virus software is unaware
- **Least Privilege Principle** → grant as little access as absolutely required
- **Firewalls** → some firewalls analyse network traffic at the application layer, helping identify and intercept malicious traffic
- **Never Insert USBs or Run Software of Unknown Origin** → just don’t.
- **Phishing** → be on the lookout, know the signs

Detection and Defense

- Signature Detection
  - Scanning operating system for code which restores original system function pointers
  - A rootkit infection means that an OS can not be trusted to report accurate system state
- Behavioural Detection
  - By comparing reports of system state with actual system behaviour, the OS can be found to be infected
  - e.g. To discover rootkit data exfiltration one could compare all open TCP-based ports with the listed ports in tcbinfo.hashbase and if there is a discrepancy the rootkit is caught

Detection and Defense

- An example of signature detection:
  1. Retrieve the address of the system call table (sysent[]) and the specific system call to be checked (argv[1])
  2. We make a copy of the potentially modified system call for local use
  3. The system call object (sysent[example]) is checked to make sure that the function it invokes (sy_call) matches it's original function
     a. I.e. if the address that sysent[mkdir] points to is different to the actual address of mkdir there is a system call
  4. The system call (sy_call) is adjusted to point to its original function



```
$ sudo ./checkcall mkdir 136 fix
Checking system call 136: mkdir
sysent[] is 0x4 at 0xc08bdf60
sysent[136] is at 0xc08be5c0 and its sy_call member points to 0xc1eb8470
ALERT! It should point to 0xc0696354 instead
Fixing it... Done.
```

Detection and Defense

- How do we remove a rootkit?
  - Essentially requires replacement of all modified driver files that change OS behaviour
  - These can be stored in hidden sections of the file system only visible to the rootkit
  - A rootkit often patches the whole system to make sure that it cannot be discovered and thusly the system can never be trusted until you do a full wipe and overwrite of the hard drive and a subsequent re-install

Applications

- Industrial Espionage
- Botnets
- Banking Information Stealing
- User Control
- Whatever you can think of...

Well-Known Rootkit Examples

- **Lane Davis and Steven Dake** - wrote the earliest known rootkit in the early 1990s.
- **NTRootkit** – one of the first malicious rootkits targeted at Windows OS.
- **HackerDefender** – this early Trojan altered/augmented the OS at a very low level of functions calls.
- **Machiavelli** - the first rootkit targeting Mac OS X appeared in 2009.
- **Greek wiretapping** – in 2004/05, intruders installed a rootkit that targeted Ericsson's AXE PBX.
- **Extended Copy Protection** - in 2005 by **Sony BMG** the software included a free music player, it also contained a rootkit which limited the end-user's ability to access the CD.
- **Zeus** - first identified in July 2007, is a Trojan horse that steals banking information by man-in-the-browser keystroke logging and form grabbing.
- **Stuxnet** - the first known rootkit for industrial control systems
- **Flame** - discovered in 2012 for Windows OS. It can record audio, screenshots, keyboard activity and network traffic.

Questions