50.020 Security
Lecture 10: Operating System Security III

Introduction

- Start preparing your CTF
- Lecture today: OS Security III

Your CTF: Labs 7-9, Week 9-11

- 3 out of labs in Security will be related to your CTF
- Please form groups of four students each (until Week 9 Thursday)
- Each group selects a topic related to security, and prepares a challenge for the other students
  - Topic of your choice, or from list of suggested topics
  - Challenge should be solve-able by a group of four students in $\approx$ 2 hours
- Lab 7 and Lab 8 are related to drafting and completing the challenge
  - You will need to provide a challenge manual, and required files
  - Each challenge should yield a *flag* when solved
- In Lab 9 you will try to solve each other's challenges. All challenges are open for roughly 1 Week

Your CTF: Labs 7-9, Week 9-11

- Tentative Timeline (all details to be confirmed this Thursday):
  - You can start preparing now (this Thursday has one Lab 8: Block Cipher)
  - By Week 10 Thursday 9pm, submit first draft. Comments to be provided by instructors
  - By Week 11 Thursday 9pm, all teams submit their finalized challenges. All teams' challenges will be open Week 11 Friday 9am - Week 12 Thursday 9am.
  - By Week 12 Thursday 9am (this is hard deadline), submit your solutions to other teams' challenges, and do peer assessment (rating).
  - Week 12 Thursday 1-3pm, Lab 10 (Network Security)

Topic Ideas

- Stack overflow
  - 32-bit stack overflow
  - More advanced Return-to-Libc
- Heap overflow
- Web exploits
  - More serious command injection
  - More serious SQL injection
- Crypto challenges
  - Asymmetric crypto
- Reverse Engineering
- Forensics
- Visit `https://ctftime.org/writeups` to get inspired

CTF: Prizes

- Projects will be graded like normal exercises
  - Up to 8 points for challenge you prepared
  - Up to 4 points for challenges that you solved
- In addition, we will vote on the *best challenge*, and a jury will award some technical score
- Together with the points gained from solving other challenges, an overall winner team will be selected
- The top 3 teams will get a book prize donated by Citibank

Another important announcement

Attendance is compulsory for Week 9 guest lecture from ST Engineering:
Time: 27 March 2019, 1pm - 3pm (1 hour Guest Lecture follow by a Tea reception).
Venue: Lecture Theater 4
Topic: Information Security or Cyber Security? ...and Why?
Details to be confirmed and announced by ISTD pillar soon.

Lecture today: OS Security III

Focus of this lecture

Last lecture:

- Buffer overflow attacks
- Malware

This lecture:

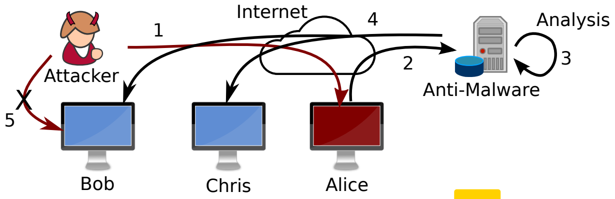
- Common OS defense mechanisms
  - Anti-malware tools
  - User accounts
  - Privilege rings
- Host attestation
- Device encryption

Anti-Malware Tools

Anti-Malware Tools

- Anti-Malware tools aim to detect malware, prevent it from compromising the host, and removing it
- Problems faced:
  - There are many malwares out there, how to recognize all?
    - Potentially millions of malware variants
  - Detection should ideally be in real-time for individual files

Detection by Anti-Malware



- Detection is done at two locations:
  - Training at Anti-Malware company
    - Based on samples collected by company
  - Customer PCs
- Detection can use *static* and *dynamic* features
- These checks are ideally executed before each execution, download
- Fundamental trade-off: Accuracy vs. Efficiency

Static Analysis

- Generally, hashes of files are not used to classify
  - hash computation requires reading whole file - slow
  - hashes change with single bit changed in malware
- Detect small *signature* sections
  - E.g. a certain sequence of machine code operations
  - String matching is a known hard problem (even exact)
    - Naive implementations: $O(mn)$ for single m length pattern, n length binary
- Newer anti-malware uses essentially machine learning approaches
  - Extract features, classify
  - Optimize for fast execution of classifier
- Many tricks are used to be faster
  - Scan only certain file types, small sizes, known hash

Defeating Static Analysis

- Polymorphic malware
  - Keep main logic, but change instructions
- Packers
  - Packers are designed to compress, encrypt, and/or modify a malicious file's format
  - Saves storage space, increases runtime
  - Also makes static analysis very hard/impossible
  - Packers are often used by malware to hide true payload

Runtime Packers

- Runtime packers will allow executables to be compressed
  - Actual unpacked code will be the same
  - Small unpacking delay when starting executable
- This saves disk space

One example packer: `http://upx.sourceforge.net/`

- This also makes analysis harder and causes effort
- More complex packers are decrypting content
  - A common solution: black-listing some unpackers

In-Memory Malware

- To avoid file-based detection, malware can also hide in memory
- If fully in-memory, malware will not be persistent
  - Might hide little hook in system startup config
- To detect, we now need to continuously monitor all RAM of host. . .

Dynamic Analysis

- In dynamic analysis, the suspected malware is supervised during execution
- This will ideally be done in secure environments, e.g. *sandboxes*
- Sandboxing and API call interruptions can slow down the application a lot
- Suspicious behaviour can be detected, and the file be flagged as malware
- Potential behaviour to be detected:
  - Anomaly Detection (e.g. unusual API calls etc.)
  - Heuristics based on API calls,
  - Suspicius self-modifying behaviour
- Defense mechanisms: detect that you are runnning in a virtualized environments, behave nicely in that case. .

Manual Malware Analysis

- How do the patterns get generated?
  - Automated machine learning
  - Manual definition by experts
- *Reverse Engineering* tries to re-construct functionality of binaries
  - *Disassembler* will translate opcode to assembler
  - *De-compiler* attempt to translate assembler to C sourcecode

```
[----------------------------------code-----------------------------------]
   0x80486d3 <main+323>:        mov    DWORD PTR [ebp-0x44],eax
   0x80486d6 <main+326>:        call   0x8048430 <fgets@plt>
   0x80486db <main+331>:        mov    ecx,DWORD PTR [ebp-0x1d]
=> 0x80486de <main+334>:        cmp    ecx,DWORD PTR [ebp-0x18]
   0x80486e1 <main+337>:        mov    DWORD PTR [ebp-0x48],eax
   0x80486e4 <main+340>:        jne    0x8048700 <main+368>
   0x80486ea <main+346>:        lea    eax,ds:0x8048856
   0x80486f0 <main+352>:        mov    DWORD PTR [esp],eax
[----------------------------------stack----------------------------------]
```

disassembled binary in gdb/peda

Advanced Malware Defence Measures

- Making reverse analysis harder by obfuscation
- Preventing analysis through encryption
  - Key has to be stored somewhere, or loaded later
  - Decrypter could also be constructed to look inconspicuous (i.e., invisible)
- Making detection harder by *polymorphism*
  - One example: altering code sequence, but keep side-effect/results the same

Exploiting Anti-Malware

- Effectively, anti-malware will interprets scanned files
- If there is a security vulnerability in the anti-malware, it can be exploited
- Anti-Malware also has to protect itself from being stopped
- In the end, the anti-malware tool will use techniques as powerful as malware to do so

Online Scanner

- A multi-engine online scanner to test suspicious files:
- For example: VirusTotal.

Browser-based sandboxing

- As seen earlier, browser is a main source of malware
- Code run to render the website is usually also sandboxed
  - Javascript, flash (many problems, finally dying), Java
- Browser becomes first line of defense against web-based malware
- Browser also start to incorporate defenses against XSS etc.

User accounts

File system-based access control

Today, the main file systems all use some sort of *Access Control*

- Several user accounts with passwords
- Access to applications and hardware based on user ID
- Default users have restricted rights to reconfigure system
- Dynamic change to root/admin role
- In Linux, access controls are limited to file access (everything is a file)

```
not@XiMac:~$ ls -al /etc/passwd
-rw-r--r-- 1 root root 1878 Aug 11 13:22 /etc/passwd

not@XiMac:~$ ls -al /dev/ttyS0
crw-rw---- 1 root dialout 4, 64 Aug 15 09:38 /dev/tty
```

Access control to memory areas + hardware

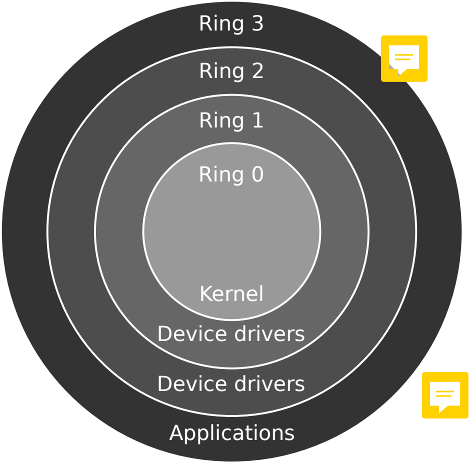
- Memory and hardware access also needs to be secured
  - Usually not based on user accounts, but on *user/kernel mode*
- Abstract concept to separate
  - *priviledged* processes (kernel)
  - User processes
- While code is executed, a flag in processer stores current mode
- Special instructions are used to change between the two modes

Privilege rings

Kernel mode and User mode

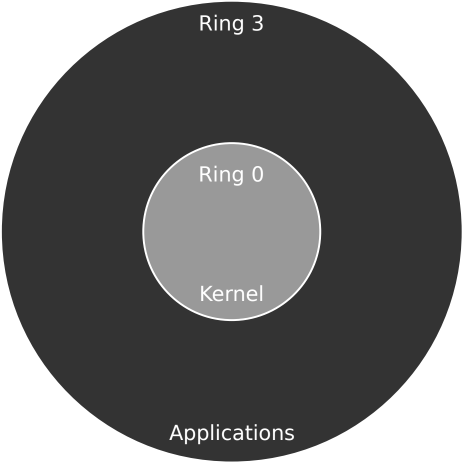
- x86 and AMD64/i64 have 4 privilege levels
- Current level is stored in special registers
- Based on current level,
  - some operations can be restricted
  - some memory access can be restricted
- This is *independent* of the user accounts
- Level 0 (highest privilege) is called *kernel mode*
- Level 3 (lower privilege) is called *user mode*
- Used by Linux, Windows, Unix, Mac OS...

Hierachical Protection Domains



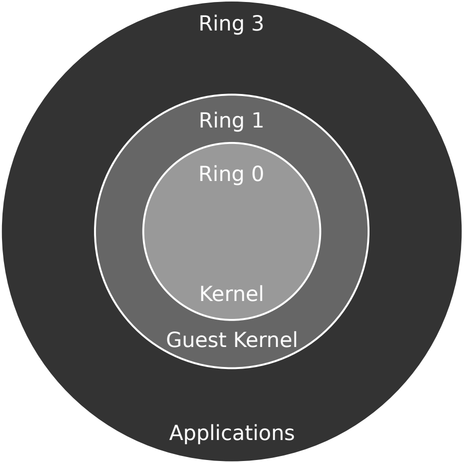
Intel x86 allows 4 protection domains/states of CPU

Hierachical Protection Domains



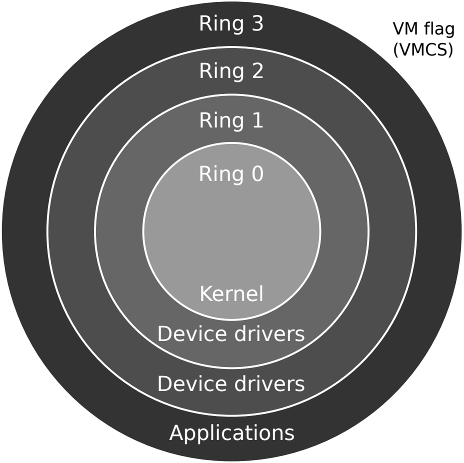
Most OS actually only use 0 and 3 (kernel/usermode)

Hierachical Protection Domains



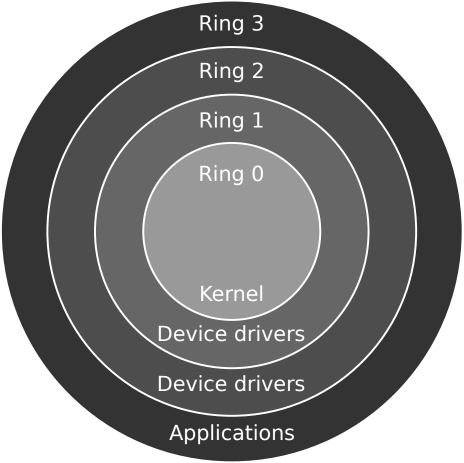
Some virtualization software uses ring 1 for guests

Hierachical Protection Domains



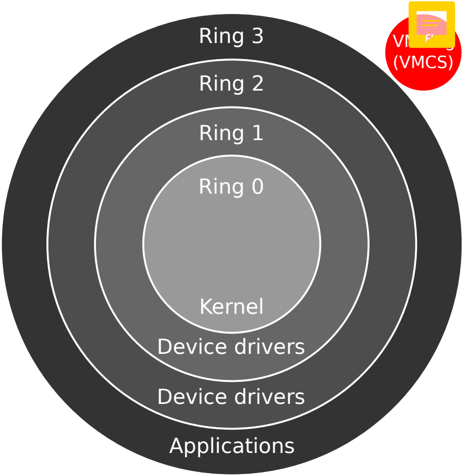
Intel VT-x and AMD-V add a "Virtualization flag" for guests

"Blue Pill" Attacks



In a blue pill attack, OS runs in ring 0

"Blue Pill" Attacks



Attacker puts OS into a VM via VM flag, preventing detection

Attacks with physical access

- So far, we discussed only remote attacks over the network
  - This provided a well-defined interface to the attacker
- With local access, the attacker has many attack vectors:
  - Media (DVDs), HDDs
  - Physical access to RAM
- All these can be exploited for attacks
- Attacker can also add devices (e.g. USB key loggers)

Access with live system

- By rebooting into a live system, the attacker can:
  - Read/write all data of the OS
  - Reset passwords or copy shared keys
  - Inject Viruses or install other malware
- Example password reset via Linux live system:

`https://bit.ly/2PDqSDV`

Keyloggers

The attacker can place physical Man-In-the-Middle devices

- USB Keyloggers: Intercept all keystrokes, send via WLAN
- DVI screengrabbers: Take screenshots every few seconds
- Basically impossible to detect/protect against from victim PC

Active compromise via USB

- The USB bus (and others) essentially allow direct memory access
- This can be used to read out memory of victim
- Bus can also be used to install malicious firmware on USB controllers
- -> An attacker can compromise a running victim by connecting USB key!

Rubber Ducky from hakShop

Windows 10 Security Mechanisms

- Since Windows 8, Windows has its own anti-Malware suite, Windows Defender
- Windows 10 requires drivers to be *signed* (indirectly, by MS)
- Windows also supports *secure boot* by default
  - Only signed bootloaders will be loaded by BIOS
  - Signed bootloader can then check integrity of loaded OS
  - Support for "measured boot"
- Windows 10 also has full OS ASLR
- Windows 10 also has NX stack protection by default

Host Attestation

Secure Boot

- Ideally, the OS loads only code that is *secure* (signed)
- But who is checking the code that is checking the signatures?
- Trust into code has to start somewhere, the *root of trust*
- The trusted platform module is a common solution to this
- Hardware module in every modern PC
  - Access control with own password
  - Contains a hash function, signature generator, registers
  - Contains hash values of secure Bootloader
  - Before booting into bootloader, signature is validated
- Only a subset of software is checked (too complex)
  - Hash values have to be updated/precomputed all the time

Remote Attestation

- Instead of starting only signed code
  - Any code is executed
  - On Boot, hash values are taken and stored in TPM
  - Can be sent to third party for *remote attestation*
- The third party then validates whether the OS is compromised
- Only then, acccess to critical resources could be allowed
- Windows calls this "Measured boot"

Device encryption

Full-disk encryption

- With physical access, attackers can
  - Boot into other OS, access all data of original OS
  - Remove physical harddrives, read out later
  - Make direct low-level copies of harddisk
- Full-disk encryption ensures that all data on HDD is encrypted
- Bootloader starts from unencrypted partition
  - Then needs to decrypt main partitions
- Keys, decrypted data is not stored on HDD, but in memory
- Running system should be safe even against physical attacker
  - All attacks described require shutdown and reboot
- Can you imagine attacks?

"Cold Boot" Attacks

- Attacker has physical access to running system
- Shuts down the system, quickly freezes RAM
- Reboots into custom OS, read out remaining content of RAM

Conclusion

To prevent malware:

- OS defence mechanisms
- Host attestation (TPM, etc)
- Full disk encryption