50.020 Security
Lecture 8: Operating System Security II

Introduction

Operating System Security

Last Lecture, we discussed:

- How buffer overflow attacks work

This Lecture:

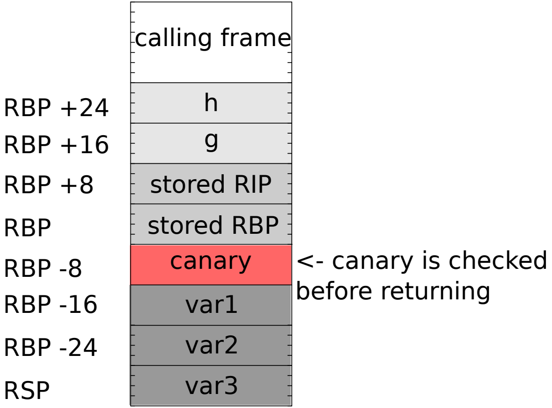
- Buffer overflow attacks (continue)
  - Countermeasures of buffer overflow attacks
    - secure coding (avoid insecure functions, etc.)
    - Canaries
    - Non-executable (NX) bit
    - ASLR (Address Space Layout Randomization)
  - Variants of buffer overflow attacks
    - Jump into existing function (e.g. doSensitiveStuff())
    - Jump into injected code by attacker
    - Jump into LibC (to bypass NX bit countermeasure)
    - Jump into PLT (Procedural Linkage Table) (to bypass ASLR countermeasure)
- More about malware

Countermeasures of buffer overflow attacks

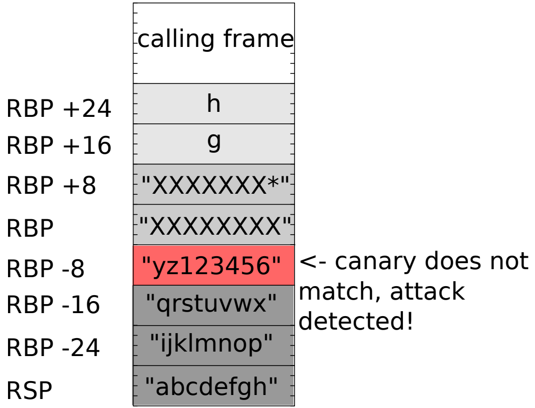
Canaries

- Canary birds were used in mines to detect gas
- Here, they are used to detect overflow attacks
- Canaries are random values saved just below RBP
- Before returning, the OS will check if the canary is "alive"
  - Canary can be random values (saved outside the frame)
  - Alternative: *Terminator* canary with \0 values, hard to overwrite
- GCC uses canaries by default! (ProPolice)
- Visual studio supports canaries as well

Canaries Figure



Canaries Figure



NX Bit

- The NX (non-executable) bit is a technology used in CPUs to **segregate** areas of memory for use by either storage of processor **instructions (code)** or for storage of **data**.
- An operating system with support for the NX bit may mark certain areas of memory as non-executable. The processor will then refuse to execute any code residing in these areas of memory.
- For example, making stack non-executable to prevent stack-based buffer overflow attacks.
- However, Return-to-LibC invariant has been proposed to defeat NX bit technology (Refer to Return-to-LibC slides later)

ASLR

- Buffer overflows require an attacker to know where each part of the program is located in memory.
- Without ASLR, libraries, stack, heap are mapped to constant addresses
- Address space layout randomization (ASLR) is an exploit mitigation technique that randomizes the location where system executables are loaded into memory (including stack address, heap address, shared library address)
- In particular when shared library address is randomized, return-to-LibC wont work since attacker needs to know LibC base address.
- Sometimes has to be enabled manually in the operating system.

Variants of buffer overflow attacks

Variants of buffer overflow attacks

- Jump into existing function (e.g. doSensitiveStuff())
- Jump into injected code by attacker
- Jump into LibC (to defeat countermeasure of NX-bit)
- Jump into PLT (Procedural Linkage Table)

We are going to focus (a bit) on the last two invariants.

Return-to-LibC attacks

- Since 2004, most OS have pages in stack either writeable OR executable...
  - NX bit, first supported by AMD64 architecture
  - So code injection does only work if NX is disabled for some reason!
- So, what can the attacker do to attack?
- NX-bit prevents jumping into injected code on stack.

Return-to-LibC attacks

- Return-to-LibC attacks return address points to LibC (standard C library) functions [1]. LibC is a library of standard functions that can be used by all C programs (and sometimes by programs in other languages)
- Addresses have to be guessed based on similar setup
- Popular functions[2] to jump into: `system()`, `unlink()`,...
- But you have to set up the stack for that function+ arguments in registers!

[1] https://linux.die.net/man/7/libc

[2] https://www.tutorialspoint.com/c_standard_library/c_function_system.htm

Return-To-PLT

- Procedural Linkage Table (PLT)
  - Used to direct executable's calls to LibC to dynamic address.
- Exploiting PLT to defeat ASLR on LibC address.
- Instead of jumping into **dynamic LibC address**, we jump into **static PLT**
- More info on:
  - `https://sploitfun.wordpress.com/2015/05/08/bypassing-aslr-part-i/`

Malware

Types of Malware

The following are popular terms for malware

- Virus
- Worm
- Adware
- Trojans
- Rootkits / Remote Access Tools
- Ransomware

What are the differences?

Spreading classification

- Spreading by replicating code into executables
  - Viruses (uncommon nowadays)
- Spreading by automated exploit over the network
  - Worms (niche cases)
- Downloaded by the user/ browser
  - Adware (as part of *free* applications)
  - Trojans (hiding payload code as part of application)

Payloads

The payload is performing the malicious actions on victim machine

- Ad injector
- Keylogger, screengrabber, etc (Spyware)
- Rootkit
- Botclient
- Ransomware

Malware (in 2013)

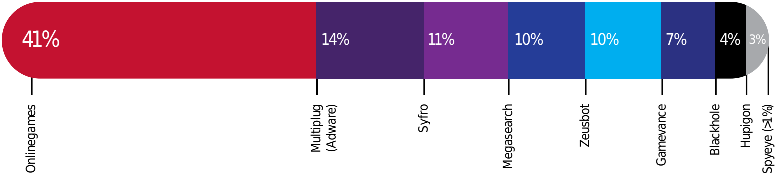
Top Malware Categories

This figure displays the top malware categories. Trojans are the most common malware, followed by adware.
Source: Sourcefire (ClamAV and FireAMP solutions)

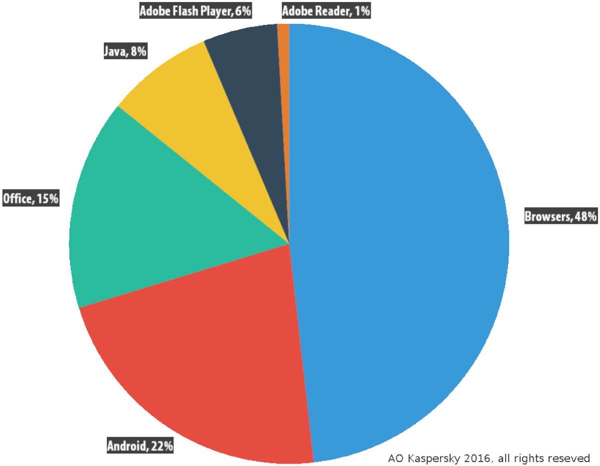
Malware for Windows (in 2013)

Top Windows Malware Families

This figure shows the top malware families for Windows. The largest, Trojan.Onlinegames, mainly comprises password stealers. Source: Sourcefire (ClamAV solution)



Attack Vectors 2016



Source: Kaspersky, all rights reserved

What will attacker's code do?

Lets assume the attacker was able to run code

- This code will run in context of executing user
- E.g. ShellShock: code run as www-data (apache user)
  - Attacker can read and write to website
  - E.g. inject javascript into website to spread malware
- Attacker can also read user's files, and many system files
  - Depending on OS/ access control setup
  - If attacker's code can connect through firewall, data can be exfiltrated
- Attacker can also try to *escalate* his rights
  - `http://www.cvedetails.com/cve/CVE-2014-4618/`
- Once attacker has root rights, he can do *anything*

Example: Locky

- Malware that was spread via pdf and MS office files
- After infection, connects to Command & Control server
  - Obtains public key of new unique RSA-2048 key pair
  - Then, all accessible documents are encrypted with AES-128
  - Original versions deleted
  - After all files are encrypted, a 0.5 Bitcoin ransom is charged
  - After paying ransom, personalized removal tool is provided

Example: WannaCry

- Ransomware from May 2017
- Leveraged *EternalBlue* vulnerability (by NSA)
- Infected hospitals and train information systems

Zero-Day Exploits

- Strong attackers might also have access to unknown exploits
- Those exploits will not be listed in CVE database, and no fixes exist
- Such zero-day exploits are quite valueable
  - They are sold for money on the Internet
- Attackers will try to use them stealthily
  - Being caught/ detected after the attack might reveal the zero-day

How to recover from compromise

- What should you do after you detect compromise?
- Once infected, it is likely that the attacker has full control
- From that point on, no on-board tools can be trusted any more
  - Rootkits are made to resist detection
- Depending on how paranoid your are
  - Run a virus scanner from a live USB key
  - Backup your data. Check for viruses.
  - Completely reinstall.
- If we are talking about work systems
  - Do forensic analysis of system(s)
  - Set up from scratch and get data from known good backups

Conclusion

- This week (OS Security I, II) :
  - A focus on buffer overflow attacks:
    - How buffer overflow attacks work
    - Countermeasures of buffer overflow attacks
    - Variants of buffer overflow attacks
  - Some general things on attack and malware
- Next Lecture:
  - OS Security III: Common OS defense mechanisms.