Session 06

Modern Offensive and Defensive Solutions

Security of Information Systems (SIS)

Computer Science and Engineering Department

November 8, 2023

Papers

- HCFI: Hardware-enforced Control-Flow Integrity
- Losing Control: On the Effectiveness of Control-Flow Integrity under Stack Attacks

Attack and Defense

- attack: exploit vulnerabilities
- defense: prevent attacks, make attacks difficult, confine attacks
- attacker needs to find one security hole
- defender has to protect all security holes
- attacker invests time
- defense mechanisms incur overhead

Attacker Perspective and Mindset

- find one vulnerability and build from that
- look for something that is valuable
- do reconnaisance, look for weak spots
- create an attack chain
- use every trick in the book
- start from existing knowledge

Defender Perspective and Mindset

- protect all entry points
- users are vulnerable, as well as technology
- use multiple defensive layers
- monitor, be proactive
- discipline, best practices are worth more than skills
- invest more on valuable targets

Attacker Pros/Cons

- apart from ethical hackers, security researchers, it's a shady business
- you may not need skills, just a weak target and a database of attack vectors
- you may get caught
- you only need to find one spot
- possible great gains
- little time for fame (annonymous)
- the Internet gives you tons of targets
- but many targets give little more than fun

Defender Pros/Cons

- less resources (time) than an attacker
- must think of everything
- is being paid constructively
- you have a purpose: keep the system running
- it never ends

Honeypots

- baits
- a system appearing as vulnerable but closely monitored
- deflect, change attention and collect attacker information

Evolution of Application Security

- buffer overflows
- shellcodes
- memory protection (DEP, WX̂)
- memory randomization
- canaries
- code reuse
- CFI (Control Flow Integrity)
- memory safety, safe programming languages
- static and dynamic analysis
- hardware enhanced security

Fine-grained ASLR

- `https://dl.acm.org/citation.cfm?id=2498135`
- issue with ASLR: memory disclosure / information leak
- one address leaked reveals all information
- do it at page level
- one leak may lead to other leaks that are chained together

SafeStack

- `https://clang.llvm.org/docs/SafeStack.html`
- part of the Code Pointer Integrity project: `http://dslab.epfl.ch/proj/cpi/`
- moves sensitive information (such as return addresses) on a safe stack, leaving problematic ones on the unsafe stack
- reduced overhead, protects against stack buffer overflows
- microStache: `https://www.springerprofessional.de/en/microstache-a-lightweight-execution-context-for-in-pro` `16103742`

Address Sanitizer

- ASan
- `https://research.google.com/pubs/archive/37752.pdf`
- `https://github.com/google/sanitizers/`
- instruments code
- only useful in development
- detects out-of-bounds bugs, memory leaks

CFI/CPI

- https://dl.acm.org/citation.cfm?id=1102165
- https://www.usenix.org/node/186160
- http://dslab.epfl.ch/proj/cpi/
- coarse-grained CFI vs fine-grained CFI
- Control Flow Integrity, Code Pointer Integrity
- protect against control flow hijack attacks
- CPI is weaker than CFI but more practical (reduced overhead)
- CPI protects all code pointers, data based attacks may still happen
- CPS (Code Pointer Separation) is a weaker yet more practical for of CPI

Shellcodes

- difficult to inject due to DEP, small buffers and input validation
- preliminary parts of the attack may remap memory region
- shellcode may do stack pivoting and then load another shellcode
- alphanumeric shellcodes: still need a binary address to bootstrap

Code Reuse

- bypass DEP by using existing pieces of code
- code gadgets
- used in ROP (*Return-Oriented Programming*) and JOP (*Jump-Oriented programming*)

Return-Oriented Programming

- gadgets ending in `ret`
- may be chained together to form an attack
- Turing-complete langauge
- `http://www.suse.de/~krahmer/no-nx.pdf`
- `https://dl.acm.org/citation.cfm?id=2133377`
- most common way of creating runtime attack vectors
- JOP: `https://dl.acm.org/citation.cfm?id=1966919`
  - gadgets end up in an indirect branch not a `ret`

Anti-ROP Defense

- prevent atacks
  - SafeStack
  - CFI/CPI, ASan
  - Microsoft CFG, RFG
- detect attacks
  - Microsoft EMET (*Enhanced Mitigation Experience Toolkit*), ProcessMitigations module

Data-Oriented Attacks

- `https://www.usenix.org/node/190963`
- `https://huhong-nus.github.io/advanced-DOP/`
- overwrites data, not code pointers
- bypasses CFI

Evolution of OS Security

- traditional main goals: functionality and reduced overhead
- recent focus on OS security: plethora of devices and use cases
- malware may easily take place among legitimate applications
- kernel exploits become more common
- OS virtualization, reduce TCB to hypervisor
- include hardware-enforced security features

Mandatory Access Control

- opposed to Discretionary Access Control, where owner controls permissions
- system-imposed settings
- increased, centralized security
- difficult to configure and maintain
- rigid, non-elastic
- Bell-LaPadula Model: `http://csrc.nist.gov/publications/history/bell76.pdf`
- SELinux, TrustedBSD, Mandatory Integrity Control

Role-Based Access Control

- https://ieeexplore.ieee.org/abstract/document/485845
- https://dl.acm.org/citation.cfm?id=266751
- aggregate permissions into roles
- role assignment, role authorization, permission authorization
- useful in organizations

Sandboxing

- assume application may be malware
- reduce potential damage
- confine access to a minimal set of allowed actions
- typically implemented at sandbox level (kernel enforced)
- iOS sandboxing, Linux seccomp

Application Signing

- ensure application is validated
- used by application stores and repositories: GooglePlay, Apple AppStore
- device may not run non-signed apps

iOS Jekyll Apps

- `https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/wang_tielei`
- apparently legimitate iOS app
- bypasses Apple vetting
- obfuscates calls to private libraries (part of the same address space, fixed from iOS 7)
- once installed turns out to be malware
- exfiltrates private data, exploits vulnerabilities

Jailreaking/Rooting

- `https://dl.acm.org/citation.cfm?id=3196527`
- get root access on a device
- close to full control
- requires a critical vulnerability that gets root access
- tethered (requires re-jailbreaking after reboot) cs non-tethered
- essential for security researchers

Hardware-centric Attacks

- side channels
- undocumented hardware features
- imperfect hardware features that leak information
- proprietary features that get exploited
- hardware is part of TCB, reveals kernel memory

Sidechannel Attacks

- do not exploit vulnerabilities in applications or kernel code
- mostly use features such as

x86 Instruction Fuzzing

- https://www.blackhat.com/docs/us-17/thursday/us-17-Domas-Breaking-The-x86-Instruction-Set-wp.pdf
- https://github.com/xoreaxeaxeax/sandsifter
- https://i.blackhat.com/us-18/Thu-August-9/us-18-Domas-God-Mode-Unlocked-Hardware-Backdoors-In-x8
  pdf
- instruction of length N is placed at the end of the page
- creates a fuzzer for the x86 instruction set
- found glitches, hidden instructions

IME

- *Intel Management Engine*
- AMD Secure Techonology
- hardware features and highly proprietary firmware
- `https://www.theverge.com/2018/1/3/16844630/intel-processor-security-flaw-bug-kernel-windows-linux`
- user space app could access kernel space memory access
- accused of being a backdoor to the system

rowhammer Attack

- https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf
- https://www.vusec.net/projects/drammer/
- https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html
- https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gai pdf
- hardware fault in DRAM chips
- constant bit flip pattern in certain rows can cause a flip in another row (not belonging to the current process)
- may be exploited to get root access

Spectre and Meltdown

- `https://meltdownattack.com`
- `https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-lipp.pdf`
- application may access data from another application
- Meltdown exploits a hardware race condition allowing an unprivileged process to read privileged data
- Spectre does a side channel attack on speculative execution features of modern CPUs
- hardware fixes by Intel, software solutions

KPTI

- *Kernel Page Table Isolation*
- `https://lwn.net/Articles/741878/`
- place kernel in separate address space
- mitigation against hardware-centric attacks

Hypervisor Attacks

- `https://dl.acm.org/citation.cfm?id=2484406`
- attack/compromise hypervisor, get control of VMs
- may exploit a vulnerability in the hypercall interface or may exploit a hardware bug
- hyperjacking

Evolution of Web Security

- path traversals, misconfigurations
- injections
- XSS
- misconfiguration
- unsafe communication
- application/language bugs

Secure Communication

- provide secure communication between client and server
- HTTPS everywhere
- Secure Cookie
- strong encryption, strong protocols

Attacks on Security Protocols

- `https://tools.ietf.org/html/rfc7457`
- `https://www.mitls.org/pages/attacks`
- flaws in protocol logic
- cryptographic design flaws
- implementation flaws

Connection Downgrade

- part of man-in-the-middle attack
- negociate a connection with weaker protocol features than the current one
- ideally drop HTTPS alltogether
- POODLE (*Padding Oracle On Downgraded Legacy Encryption*)
- `https://www.openssl.org/~bodo/ssl-poodle.pdf`

Advanced Injection Attacks

- LDAP, XPath injection
- blind SQL injection: content-based and time-based
- `https://www.owasp.org/images/7/74/Advanced_SQL_Injection.ppt`
- `https://nvisium.com/blog/2015/06/17/advanced-sql-injection.html`

Language Bugs

- bugs/vulnerabilities in frameworks
- bugs/vulnerabilities in web modules or languate interpreter

Modern Offensive and Defensive Techniques

- attacks focus on low-level aspects of a system: hide features, exploit hardware, side channels, protocol design
- assume better/improved applications but imperfect system/protocol/configuration design
- defense takes more time and incurs significant overhead
- battle rages on

Keywords

- honeypot
- fine-grained ASLR
- SafeStack
- AddressSanitizer
- CFI/CPI
- code reuse
- ROP, JOP
- data-oriented attacks
- MAC, RBAC
- sandboxing
- Jekyll apps
- jailbreak, rooting
- side channel attacks
- IME attack
- Meltdown, Spectre
- KPTI
- rowhammer
- connection downgrade
- POODLE
- blind SQL injection

Resources

- see URLs accross slides