STONY BROOK
STATE UNIVERSITY OF NEW YORK

CSE 509

Course Summary

Cryptography Basics

- **Algorithm Vs Key**
- **Symmetric key ciphers (DES, AES, …)**
  - Block vs stream ciphers
- **Public key techniques (RSA, …)**
  - Encryption Vs Signing
- **When to use public vs symmetric crypto**
  - speed of encryption vs ease of key distribution
- **Hash functions (MD5, SHA, …)**
- **Random number generation**
- **Applications**
  - Encryption (Block vs Stream Ciphers)
  - Key generation
  - Authentication
  - Digital signatures
  - Certificates

User Authentication

- **Something you know (secret), have (badge, smartcard) or are (biometrics)**
- **Password-based authentication**
  - History and weaknesses
  - Offline/online attacks: Differences in methods and defenses
  - Brute force vs Dictionary attacks
  - Ease of remembering Vs guessing
  - Password theft, Phishing and trusted path
  - Variants and Improvements
  - Master password and password managers (ssh, browsers, …)
  - Multi-factor authentication
- **Biometrics**
- **Network authentication**
  - Challenge/response protocols
  - SSL, SSH, OTPs, ...

Processor and Virtual Machine Security

- **Principles behind processor and OS security**
  - privileged mode and privileged instructions; kernel vs user code
  - memory protection
  - interrupts and system calls
  - virtualized resources and access control
- **Efficient virtualization**
  - Privileged vs sensitive instructions
- **Process Vs Namespace Vs System virtualization**
  - Docker security
- **Type I and Type II VMMs**
- **Paravirtualization Vs full virtualization**
- **Implementation techniques**
  - Binary translation, paravirtualization, hardware-assisted virtualization
- **Memory virtualization**
- **Security applications**
  - Honeypots, sandboxes, malware analysis, high-assurance VMs
  - Protection from compromised OS

OS Security and Access Control

- **Terminology:** Principal, subject, object, RM, Security kernel, TCB
- **Discretionary Access Control**
  - Access control matrix
  - Groups and RBAC
  - ACLs
    - UNIX permission model
    - effective, real and saved userid, primary and supplementary groups
    - setuid and setgid
  - Capabilities
- **Trojan Horse and Mandatory Access Control**
  - MLS: Bell-La Padula, Biba models; Benefits and drawbacks of information flow
  - Domain and Type Enforcement: SELinux; Benefits and drawbacks
- **POSIX Capabilities**
  - Model, differences with classic capabilities
- **Policies and mechanisms for containing untrusted code**
  - chroot jails, seccomp: basic, BPF and eBPF
  - one-way isolation, information flow policies
- **Other types of policies:** Clark-Wilson policy, Chinese wall policy

Principles of Secure System Design

- Least privilege
- Fail-safe defaults (default deny)
- Economy of mechanism (simplicity => assurance)
- Complete mediation (look out for ways in which an access control mechanism may be bypassed)
- Open design (no security by obscurity)
- Separation of privilege (similar to separation of duty)
- Least common mechanism (avoid unnecessary sharing)
- Psychological acceptability (onerous security requirements will be actively subverted by users)

Software Vulnerabilities: Memory Errors

- **Memory corruption exploits**
  - Stack-smashing
  - Heap overflows
  - Format-string bugs
  - Integer overflows
- **Exploit defenses**
  - DEP/NX
  - Canaries
  - Separating control data from other data
  - Randomization
    - Address-space (absolute or relative address)
    - Data-space
    - Instruction-space
  - CFI
- **Advanced exploits:**
  - ROP
  - double pointer attacks
  - partial overflows
  - information leakage
  - heapspray
- **Preventing memory errors**
  - Definition of memory error
  - Spatial vs Temporal Errors
  - Defenses

Software Vulnerabilties: Injections

- **Example attacks**
  - SQL injection
  - Command injection, script injection, ...
  - XSS
  - Path traversal
  - Format string bugs
  - Memory corruption/code injection attacks
- **Defenses**
  - Static taint analysis
  - Runtime fine-grained taint-tracking: data dependence, control dependence, implicit flows.
  - Taint-aware policy enforcement

More Software Vulnerabilities …

- **Browser attacks**
  - XSS
  - CSRF
- **CWE-25**
- **File-name based attacks**
  - Symlink attacks
  - TOCTTOU attacks
    - How to succeed in races …

Program Transformations for Security

- **General idea**
  - Maintain additional metadata, check policies using this
- **Source-to-source transformations**
  - Guarding techniques
  - Randomization techniques
  - Full memory error detection
  - Fine-grained taint-tracking
  - Control-flow integrity

Malicious Code

- **Current threat environment: Profit-driven crime**
- **Types**
  - Viruses
  - Worms
  - Spam
  - Phishing
  - Botnets
  - Rootkits
  - Spyware
  - DDoS
  - Extortion
  - Cyberwar

Malicious code: Stealth Techniques

- **Stealth and Obfuscation**
  - Behavioral obfuscation
    - Anti-virtualization and anti-analysis techniques
    - Trigger-driven
  - Code obfuscation
    - Control-flow obfuscation
    - Data obfuscation
    - Encryption and packing
    - Polymorphism
    - Metamorphism

Untrusted code and Web Security

- **Javascript**
  - Vs Java
  - DOM model, BOM model
- **HTTP protocol**
  - GET Vs POST, Responses
  - Maintaining state: cookies; sessions; authentication
  - HTML forms, parameters, server-side processing
- **Same origin policy (SOP) and Frames**
  - Page isolation, cookie isolation, network isolation
  - Ajax and XmlHttpRequests
  - Caveats: Embedded scripts, external requests
  - Reflected and persistent XSS; Defenses
  - CSRF and defenses
- **SSL Stripping and defenses (e.g., HSTS)**
- **Other attacks**
  - Clickjacking
  - Timing attacks
  - Logic vulnerabilities

Untrusted code defense

- **Untrusted code implies strong adversary, requires correspondingly strong defenses**
- **System-call interception**
  - Techniques and trade-offs
- **Inline-reference monitors**
  - Issues, challenges
  - Software-based fault-isolation: RISC Vs CISC; PittSFIeld
  - Control-flow integrity

    Coarse vs fine-grained, implementation strategies

- **Sandboxing (confinement policies)**
  - Policies are hard to write!
  - Indirect attacks!
  - Examples: Native Client, WebAssembly
- Isolation
  - Virtual machines
    - VMware, Xen, KVM, Qemu
  - One-way isolation
    - With copy-on-write
  - Two-way isolation
    - Smart phones
    - Caveats

Program Transformation on Binaries

- **Key challenges compared to source code**
  - disassembly techniques and challenges
  - rewriting challenges
- **Dynamic translation**
  - Dynamo Rio, Valgrind, Qemu, Pin, …
  - How it achieves speed
  - Applications: Program shepherding, Taint-tracking, ...
- **Static instrumentation**
  - Disassembly
  - Lifting to machine-independent intermediate code
  - Pointer fixup
  - Secure instrumentation
- **Issues and limitations**

Intrusion Detection

- **Network intrusions**
  - DDoS
  - Botnets
  - Reflection attacks
  - Worms
- **Attack stages**
  - Probing
  - DoS
  - Privilege escalation

Intrusion Detection

- **False positives and negatives**
- **Observation points:**
  - Host-based Vs Network intrusion detection
    - Benefits and drawbacks
- **Techniques**
  - Anomaly detection
  - Misuse detection
  - Specification-based detection
- **Algorithms**
  - Pattern-matching
  - Machine learning

Host-based Intrusion Detection

- **System call logs**
- **APT Campaigns**
  - Challenges: Stealth, sophistication, scale, duration
  - Solutions
- **Evasion: Mimicry attacks**

Static Analysis for Vulnerability Detection

- **Techniques to identify potential bugs and vulnerabilities**
- **Requires a model of what is good behavior, or bad behavior**
  - "Good behaviors" are typically application specific, and hard to come by
  - "Bad behaviors" can be somewhat more generic
    - Common software vulnerabilities
      - Buffer overflow, SQL injection, …
    - Inconsistencies
      - Access check or locking on some program paths, but not others

Static Analysis

- **Usually require source code**
  - Binary code analysis limited by absence of type/bounds information, as well as higher level control structures
- **Most program properties are undecidable**
  - Static analysis has to approximate in order to terminate. Approximation means that analysis can be sound or complete, but not both.
  - Sound: Guaranteed to find all vulnerabilities
  - Complete: No false positives
  - Practical issues: FPs and FNs, scalability, range of properties that can be supported, ...

Dynamic Analysis

- **Manual testing**
- **Random testing ("fuzz testing")**
  - Vulnerabilities often arise due to insufficient testing and optimistic assumptions about input
  - This means that incorrect inputs will cause unexpected behaviors
  - Random input will typically cause crashes
    - Using a debugger or other means, hackers can find additional information to turn the crash into an exploit
- **Coverage-guided fuzzing**
- **Manually assisted fuzz testing**
  - In many cases, random inputs don't work, as they get discarded very early
    - Most of the code is not exercised
  - Better to ensure that some parts of input are valid, so as to traverse more program paths
    - Remaining parts of input can be fuzzed

Symbolic Execution

- **"Intelligent" approach that chooses inputs to ensure more coverage**
  - Often based on some form of symbolic execution
    - Variables left unbound
    - As conditions are tested, constraints on unbound inputs are gathered, depending on whether "then" or "else" clause is taken
    - When multiple conditions are present on the value of a variables, use a constraint solving procedure to narrow down the range
  - Key challenges
    - Range of constraints that can be handled
    - state-space explosion
    - Many approaches choose to bind variables to concrete values when faced with these problems
- **Penetration testing**
  - Just another name for dynamic vulnerability testing

Side-channel attacks and physical security

- **Covert channels**
  - Intentionally embedded
  - Implicit flows, timing, steganographic techniques, ...
- **Side channel attacks**
  - Timing analysis, power monitoring
  - Differential fault analysis
  - Emanations (keyboard, power, screen/camera, shock sensor)
  - Remanence
- **Physical layer attacks and tamper resistance**
  - transmit info by file name or metadata (e.g., timestamp)
    - Information retrieved by checking file presence or stat
      - No need to read the file (or have read permissions on the file)
  - "Port-knocking"
    - Transmit info by probing network ports in a certain sequence
  - tcp acks or retransmissions, packet fragmentation, …

Side-channel attacks and physical security

- **Covert channels**
  - Timing, implicit flows, DNS requests, ...
- **Side-channels**
  - Execution time