

Operating Systems and Program Security

Kc Udonsi

Application Security & Threat Modelling

Common Threats Against Software

- ➡ Presence of security bugs “Vulnerabilities”
- ➡ Unauthorized modification e.g Backdoors
- ➡ Supply chain bugs - Vulnerabilities in dependencies and/or tooling, partners

Why do Vulnerabilities exist ?

- ➡ Fundamental oversights in software design. Designed to do the wrong thing a.k.a Design Flaws
- ➡ Implementation flaws/bugs relevant to security a.k.a Technical Flaws
- ➡ Faulty inter-operation with executing environment a.k.a Operational Flaws
- ◎ **Arbitrarily trusting input data, misplaced trust**

Threat Modelling

- ➡ Description of system
- ➡ Potential threats to the system (threats against CIA)
- ➡ Actions that can be taken to mitigate each threat
- ➡ Validation of model
- ➡ Threat Modelling Manifesto: <https://www.threatmodelingmanifesto.org/>
- ➡ Think about “abuse cases” and what can be done to mitigate those

Secure Programming

- ➡ Familiarity with relevant vulnerability classes
- ➡ Modularity - separate modules for separate functionalities
- ➡ Sanitize, validate, restrict input data even between modules or components (mutual suspicion)
- ➡ Be “fault tolerant” by having a consistent policy to handle failure
- ➡ Use reputable, security conscious and well maintained libraries
- ➡ Adopt good programming practices, be security aware

Software Security Assessment

- ➡ Manual, guided or automated audit and security testing
- ➡ Security test cases may validate threat mitigation strategies
- ➡ Internal or external auditors methodologically review code for design, implementation or operational flaws
 - ➡ Vulnerability Rewards Program, Bug Bounties etc
- ➡ Fuzz testing can be combined with manual audits to discover vulnerable code paths
- ➡ Can be carried out at various stages of the SDLC

Secure Software Development Life Cycle

- ➡ Description of subject
- ➡ Potential threats to the system
- ➡ Actions that can be taken to mitigate each threat
- ➡ Validation of model
- ➡ Continuous security testing throughout the SDLC “DevSec Ops”
- ➡ Think about “abuse cases” and what can be done to mitigate those

Formal Methods of Verification

Mathematical description of the problem

*Refinement
steps*



Proof of correctness



Executable code
or hardware design

Formal Methods of Verification

➡ Examples:

Hardware design (VHDL, Verilog)

✓ Used by semi-conductor companies such as Intel

Critical embedded software (B/Z, Lustre/Esterel)

- ✓ Urban Transportation
(METEOR Metro Line 14 in Paris by Alstom)
- ✓ Rail transportation (Eurostar)
- ✓ Aeronautic (Airbus, Eurocopter, Dassault)
- ✓ Nuclear plants (Schneider Electric)

Pros and cons of using formal methods

- ✓ Nothing better than a mathematical proof
 - ➔ A code “proven safe” is safe
- ⦿ Development is time and effort (and so money) consuming
 - ➔ Should be motivated by the risk analysis
- ⦿ Do not prevent from specification bugs
 - ➔ Example of network protocols

Operating System Security

Exploit mitigation, Endpoint Detection and Response (EDRs), Security Policies

Exploit Mitigation

Exploit Mitigation Contd.

- ➡ Fortify Source Functions
- ➡ Stack Canaries
- ➡ Data Execution Prevention / Non-Executable Stack
- ➡ Address Space Layout Randomization (ASLR)

Exploit Mitigation Contd.

- ➡ Position Independent Executables
- ➡ Control Flow Guard
- ➡ Application sandboxing
- ➡ Non-exhaustive. Often implemented at OS or Compiler

Fortify Source Functions

- ➔ GCC macro `FORTIFY_SOURCE` provides buffer overflow checks for unsafe C libraries

`memcpy, mempcpy, memmove, memset, strcpy, stpcpy, strncpy, strcat, strncat, sprintf, vsprintf, snprintf, vsnprintf, gets`

Checks are performed

- some at compile time (compiler warnings)
- other at run time (code dynamically added to binary)

Canaries

- The compiler modifies every function's prologue and epilogue regions to place and check a value (a.k.a a canary) on the stack
- When a buffer overflows, the canary is overwritten. The programs detects it before the function returns and an exception is raised
- Different types:
 - random canaries
 - xor canaries
- Disabling Canary protection on Linux
`$ gcc ... -fno-stack-protector`
- Bypassing canary protection : *Structured Exception Handling (SEH)* exploit overwrite the existing exception handler structure in the stack to point to your own code

DEP/NX - Non Executable Stack

- The program marks important structures in memory as non-executable
- The program generates an hardware-level exception if you try to execute those memory regions
- This makes normal stack buffer overflows where you set `eip` to `esp+offset` and immediately run your shellcode impossible
- Disabling NX protection on Linux
`$ gcc ...-z execstack`
- Bypassing NX protection : *Return-to-lib-c* exploit
return to a subroutine of the lib C that is already present in the process' executable memory

ASLR - Address Space Layout Randomization

- The OS randomize the location (random offset) where the standard libraries and other elements are stored in memory
- Harder for the attacker to guess the address of a lib-c subroutine
- Disabling ASLR protection on Linux
`$ sysctl kernel.randomize_va_space=0`
- Bypassing ASLR protection : Brute-force attack to guess the ASLR offset
- Bypassing ASLR protection : *Return-Oriented-Programming (ROP)* exploit use instruction pieces of the existing program (called "gadgets") and chain them together to weave the exploit

PIC/PIE - Position Independent Code/Executables

- **Without PIC/PIE**

code is compiled with absolute addresses and must be loaded at a specific location to function correctly

- **With PIC/PIE**

code is compiled with relative addressing that are resolved dynamically when executed by calling a function to obtain the return value on stack

Confined execution environment - Sandbox

A sandbox is tightly-controlled set of resources for untrusted programs to run in

- ➔ Sandboxing servers - virtual machines
- ➔ Sandboxing programs
 - Chroot, Seccomp, AppArmor in Linux
 - Sandbox in MacOS
 - Application Guard Windows
 - Windows Sandbox
- ➔ Sandboxing applets - Java and Flash in web browsers

Security Policies

Baselining System Security

- ➡ OSes strive for secure out-of-the-box
- ➡ Granular controls may be required to customize security posture
- ➡ Often pushed down as configurations or profiles in enterprise environment
- ➡ May include firewall settings, password strength requirements, application installations, removal drive controls, suspicious site access, file download policies etc.

Vulnerability Management

To Patch or Not to Patch ...

- ➡ Patches often need to be validated
- ➡ Risk-based discovery, prioritization and remediation
- ➡ Kernel Data Protection (Windows)
- ➡ System Coprocessor / Kernel Integrity Protection (MacOS)
- ➡ Pointer Authentication Codes (MacOS)
- ➡ Code integrity and signing
- ➡ Non-exhaustive. Often implemented at OS or hypervisor level (Virtualization Based Security)

Securing the Kernel

Kernel Patch and Exploit Mitigations

- ➡ Kernel Self-Protection (Linux)
- ➡ Kernel Patch Guard / Patch Protection (KPP) (Windows)
- ➡ Kernel Data Protection (Windows)
- ➡ System Coprocessor / Kernel Integrity Protection (MacOS)
- ➡ Pointer Authentication Codes (MacOS)
- ➡ Code integrity and signing
- ➡ Non-exhaustive. Often implemented at OS or hypervisor level (Virtualization Based Security)

Endpoint Detection and Response

Endpoint Protection

- ➡ Historic anti-virus - signature based detection
- ➡ Heuristics and behavioural based detection
- ➡ Implemented as an extension to the kernel often with user-space components
- ➡ Passive or Active mode, event logging and streaming
- ➡ Often featuring a cloud component for incident investigation and security overview
- ➡ Still software hence can be contain vulnerabilities

Endpoint Protection

➔ Mitre Attack Matrix



Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 13 techniques	Persistence 19 techniques	Privilege Escalation 13 techniques	Defense Evasion 42 techniques	Credential Access 17 techniques	Discovery 30 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Active Scanning (3) Gather Victim Host Information (4) Gather Victim Identity Information (3) Gather Victim Network Information (6) Gather Victim Org Information (4) Phishing for Information (3) Search Closed Sources (2) Search Open Technical Databases (5) Search Open Websites/Domains (3) Search Victim-Owned Websites	Acquire Infrastructure (7) Compromise Accounts (3) Compromise Infrastructure (7) Develop Capabilities (4) Establish Accounts (3) Obtain Capabilities (6) Stage Capabilities (6)	Drive-by Compromise Exploit Public-Facing Application External Remote Services Hardware Additions Phishing (3) Replication Through Removable Media Supply Chain Compromise (3) Trusted Relationship Valid Accounts (4)	Command and Scripting Interpreter (8) Container Administration Command Deploy Container Exploitation for Client Execution Inter-Process Communication (3) Native API Scheduled Task/Job (5) Serverless Execution Shared Modules Software Deployment Tools System Services (2) User Execution (3) Windows Management Instrumentation	Account Manipulation (5) BITS Jobs Boot or Logon Autostart Execution (14) Boot or Logon Initialization Scripts (5) Browser Extensions Compromise Client Software Binary Create Account (3) Create or Modify System Process (4) Event Triggered Execution (16) External Remote Services Hijack Execution Flow (12) Implant Internal Image Modify Authentication Process (7) Office Application Startup (6) Pre-OS Boot (5) Scheduled Task/Job (5) Server Software Component (5) Traffic Signaling (2) Valid Accounts (4)	Abuse Elevation Control Mechanism (4) Access Token Manipulation (5) BITS Jobs Build Image on Host Debugger Evasion Deobfuscate/Decode Files or Information Deploy Container Direct Volume Access Domain Policy Modification (2) Execution Guardrails (1) Exploitation for Defense Evasion File and Directory Permissions Modification (2) Hide Artifacts (10) Hijack Execution Flow (12) Impair Defenses (9) Indicator Removal (9) Indirect Command Execution Masquerading (7) Modify Authentication Process (7) Modify Cloud Compute Infrastructure (4) Modify Registry Modify System Image (2) Network Boundary Bridging (1) Obfuscated Files or Information (9) Plist File Modification Pre-OS Boot (5) Process Injection (12) Reflective Code Loading Rogue Domain Controller Rootkit Subvert Trust Controls (6) System Binary Proxy Execution (13) System Script Proxy Execution (1) Template Injection Traffic Signaling (2) Trusted Developer Utilities Proxy Execution (1) Unused/Unsupported Cloud Regions Use Alternate Authentication Material (4) Valid Accounts (4) Virtualization/Sandbox Evasion (3) Weaken Encryption (2) XSL Script Processing	Abuse Elevation Control Mechanism (4) Access Token Manipulation (5) BITS Jobs Build Image on Host Debugger Evasion Deobfuscate/Decode Files or Information Deploy Container Direct Volume Access Domain Policy Modification (2) Execution Guardrails (1) Exploitation for Defense Evasion File and Directory Permissions Modification (2) Hide Artifacts (10) Hijack Execution Flow (12) Impair Defenses (9) Indicator Removal (9) Indirect Command Execution Masquerading (7) Modify Authentication Process (7) Modify Cloud Compute Infrastructure (4) Modify Registry Modify System Image (2) Network Boundary Bridging (1) Obfuscated Files or Information (9) Plist File Modification Pre-OS Boot (5) Process Injection (12) Reflective Code Loading Rogue Domain Controller Rootkit Subvert Trust Controls (6) System Binary Proxy Execution (13) System Script Proxy Execution (1) Template Injection Traffic Signaling (2) Trusted Developer Utilities Proxy Execution (1) Unused/Unsupported Cloud Regions Use Alternate Authentication Material (4) Valid Accounts (4) Virtualization/Sandbox Evasion (3) Weaken Encryption (2) XSL Script Processing	Adversary-in-the-Middle (3) Brute Force (4) Credentials from Password Stores (5) Exploitation for Credential Access Forced Authentication Forge Web Credentials (2) Input Capture (4) Modify Authentication Process (7) Multi-Factor Authentication Interception Multi-Factor Authentication Request Generation Network Sniffing OS Credential Dumping (8) Steal Application Access Token Steal or Forge Authentication Certificates Steal or Forge Kerberos Tickets (4) Steal Web Session Cookie Unsecured Credentials (7)	Account Discovery (4) Application Window Discovery Browser Bookmark Discovery Cloud Infrastructure Discovery Cloud Service Dashboard Cloud Service Discovery Cloud Storage Object Discovery Container and Resource Discovery Debugger Evasion Domain Trust Discovery File and Directory Discovery Group Policy Discovery Network Service Discovery Network Share Discovery Network Sniffing Password Policy Discovery Peripheral Device Discovery Permission Groups Discovery (3) Process Discovery Query Registry Remote System Discovery Software Discovery (1) System Information Discovery System Location Discovery (1) System Network Configuration Discovery (1) System Network Connections Discovery System Owner/User Discovery System Service Discovery System Time Discovery Virtualization/Sandbox Evasion (3)	Exploitation of Remote Services Internal Spearphishing Lateral Tool Transfer Remote Service Session Hijacking (2) Remote Services (6) Replication through Removable Media Software Deployment Tools Taint Shared Content Use Alternate Authentication Material (4)	Adversary-in-the-Middle (3) Archive Collected Data (3) Audio Capture Automated Collection Browser Session Hijacking Clipboard Data Data from Cloud Storage Data from Configuration Repository (2) Data from Information Repositories (3) Data from Local System Data from Network Shared Drive Data from Removable Media Data Staged (2) Email Collection (3) Input Capture (4) Screen Capture Video Capture	Application Layer Protocol (4) Communication Through Removable Media Data Encoding (2) Data Obfuscation (3) Dynamic Resolution (3) Encrypted Channel (2) Fallback Channels Ingress Tool Transfer Multi-Stage Channels Non-Application Layer Protocol Non-Standard Port Protocol Tunneling Proxy (4) Remote Access Software Traffic Signaling (2) Web Service (3)	Automated Exfiltration (1) Data Transfer Size Limits Exfiltration Over Alternative Protocol (3) Exfiltration Over C2 Channel Exfiltration Over Other Network Medium (1) Exfiltration Over Physical Medium (1) Exfiltration Over Web Service (2) Scheduled Transfer Transfer Data to Cloud Account	Account Access Removal Data Destruction Data Encrypted for Impact Data Manipulation (3) Defacement (2) Disk Wipe (2) Endpoint Denial of Service (4) Firmware Corruption Inhibit System Recovery Network Denial of Service (2) Resource Hijacking Service Stop System Shutdown/Reboot

Last modified: 01 April 2022