جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأمن السيبراني

Lecture: 8

Control-Flow Hijacking

Subject: software security

second Stage

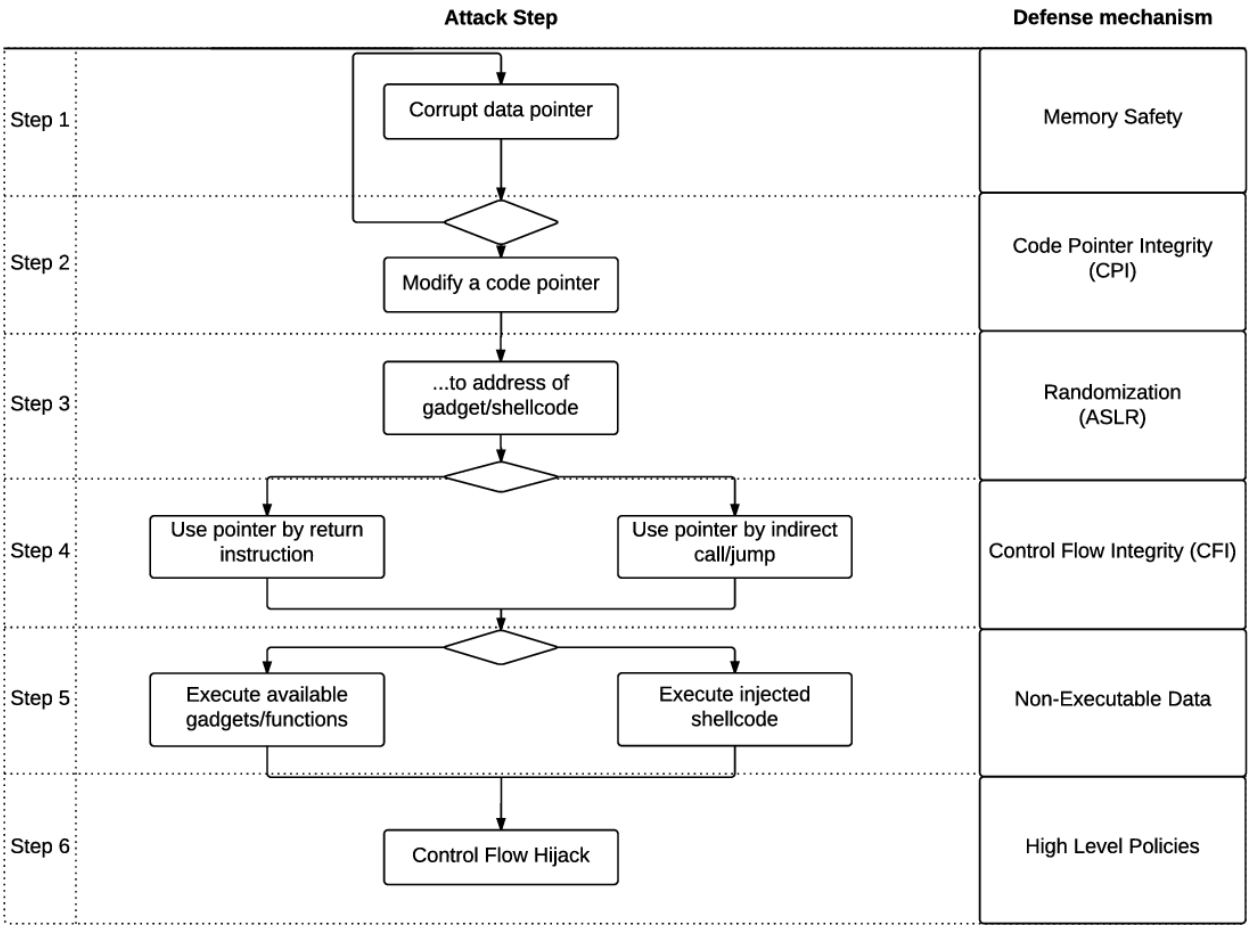
Lecturer: Asst. Lecturer. Suha Alhussieny

Control-Flow Hijacking

Control-flow hijacking is a type of security vulnerability that allows an attacker to alter the flow of execution in a program. This usually involves redirecting the program's execution to a location of the attacker's choosing. It's often used to exploit software vulnerabilities such as buffer overflows, where an attacker can overwrite parts of memory and redirect the execution flow to execute malicious code.

Here are some common types of control-flow hijacking techniques:

1. **Buffer Overflow Attacks**: By overwriting a buffer, an attacker can alter the return address on the stack, redirecting the program's execution to their own code.
2. **Return-Oriented Programming (ROP)**: An attacker uses existing code snippets (gadgets) that end in return instructions to execute malicious code without injecting new code into the process.
3. **Jump-Oriented Programming (JOP)**: Similar to ROP but uses jump instructions instead of return instructions to build a chain of gadgets for malicious purposes.
4. **Code Injection Attacks**: Injecting malicious code into the process's memory space and redirecting execution to it.
5. **Function Pointer Overwriting**: Manipulating function pointers to redirect the program's execution to malicious code.



Code Injection

Code injection is a type of security vulnerability where an attacker is able to insert or "inject" malicious code into a program or system. This injected code is then executed by the system, leading to unauthorized actions or compromise. Code injection attacks can affect various types of applications and systems, including web applications, databases, and even local software.

Common Types of Code Injection

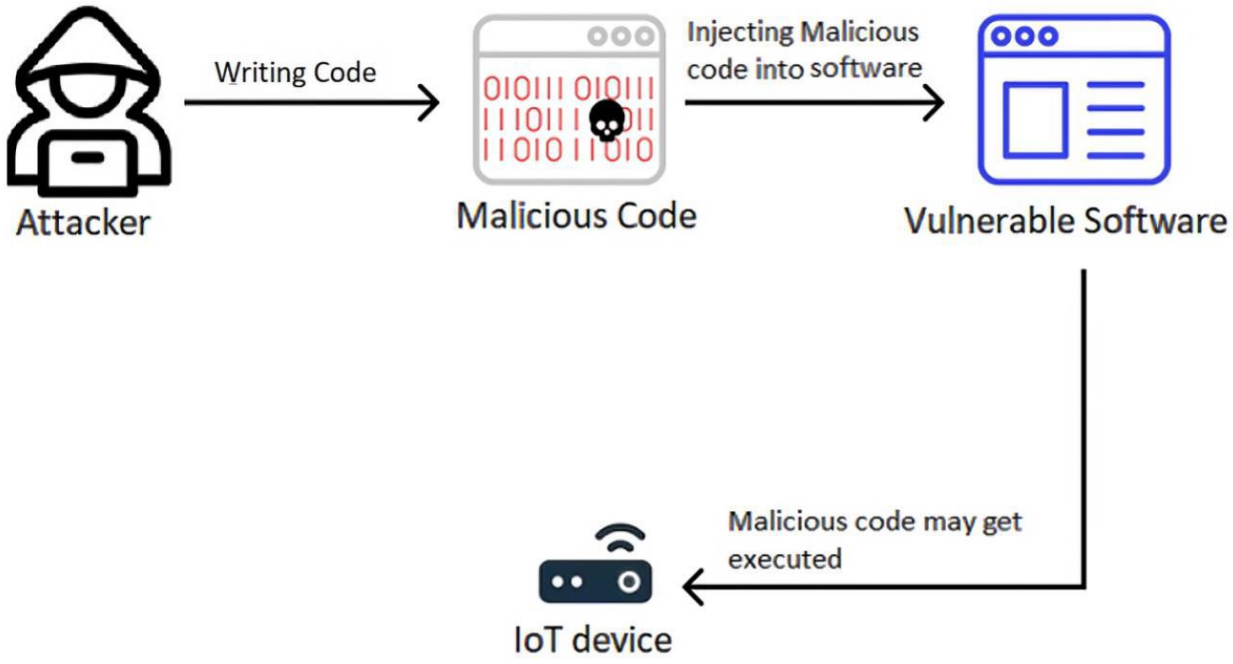
1. **SQL Injection**: An attacker inserts malicious SQL queries into a form input or URL parameter to manipulate or gain unauthorized access to a database.

2. **Cross-Site Scripting (XSS)**: Malicious scripts are injected into web pages viewed by other users, potentially leading to session hijacking, data theft, or other malicious actions.

3. **Command Injection**: Malicious commands are injected into an application's input fields, which are then executed by the server's command-line interface.

4. **Code Injection in Programming Languages**: Attacks that exploit weaknesses in how programming languages handle code execution, such as JavaScript eval(), PHP eval(), or Python exec().

5. **Script Injection**: Similar to XSS but more broadly involves injecting scripts into any environment where the script might be executed.



Code Reuse

A code reuse attack is a type of security exploit where an attacker leverages existing code within a program or system to perform malicious actions. Instead of injecting new code, the attacker reuses legitimate code already present in the program, often to bypass security mechanisms. These attacks are typically sophisticated and require an understanding of the program's structure and available code snippets.

Common Types of Code Reuse Attacks

1.**Return-Oriented Programming (ROP)**: An attacker uses sequences of instructions (gadgets) ending in return instructions to build a chain of operations that perform malicious actions. This technique is used to exploit vulnerabilities like buffer overflows, where the attacker manipulates the program's control flow to execute their chosen sequence of gadgets.

2.**Jump-Oriented Programming (JOP)**: Similar to ROP, but instead of return instructions, JOP relies on jump instructions to control the execution flow. JOP can be used to bypass security defenses by exploiting existing code that uses jump instructions.

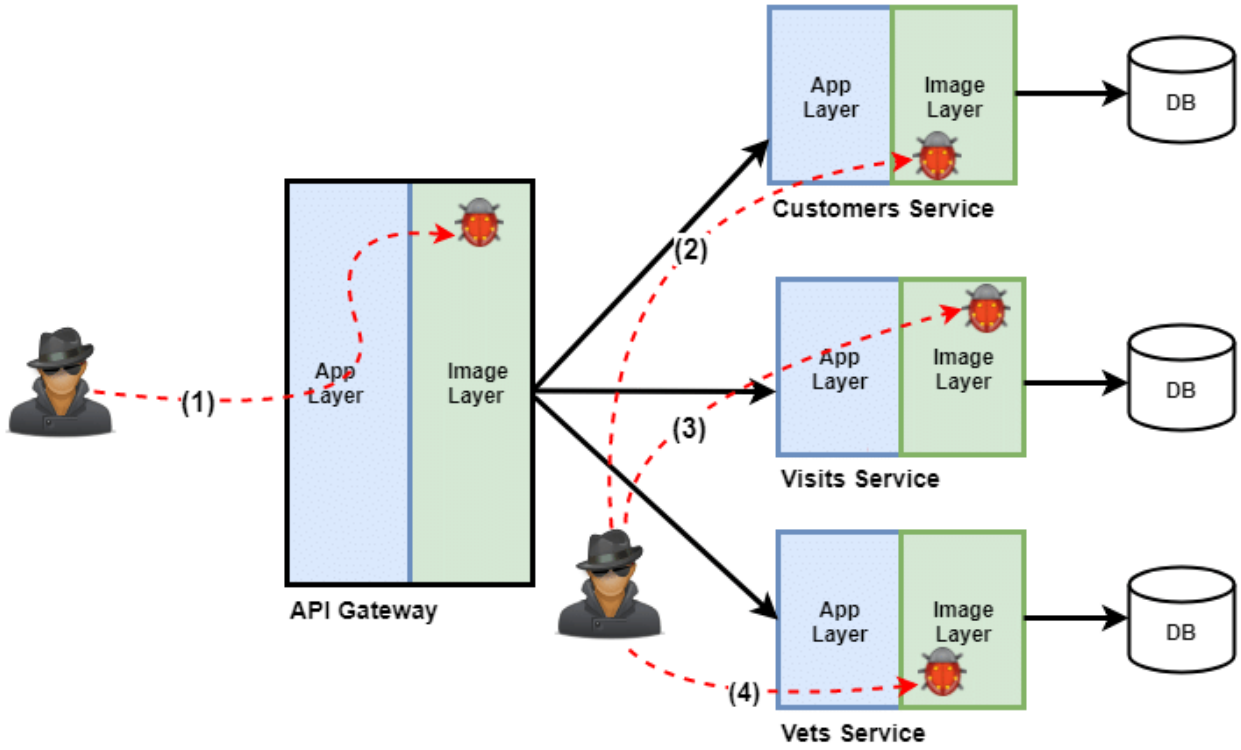
3.**Call-Oriented Programming (COP)**: This technique involves chaining together existing code that contains call instructions. By manipulating these calls, an attacker can execute a series of operations in a controlled manner.

Characteristics of Code Reuse Attacks

• **No Code Injection**: These attacks do not require injecting new code into the memory but instead exploit existing code, which can be harder to detect and mitigate.

• **Bypassing Security Mechanisms**: Code reuse attacks can bypass security measures like Data Execution Prevention (DEP) or Address Space Layout Randomization (ASLR) by avoiding the need to execute injected code.

• **Exploitation of Code Patterns**: Attackers exploit predictable patterns or behaviors in the existing code to achieve their goals.



Q/

Control-Flow Hijacking

1. What is control-flow hijacking?
   a) A technique to maliciously alter a program's execution flow

b) A method to debug software

c) A tool for memory optimization

d) A type of hardware failure

2. Which technique overwrites the return address on the stack?

a) SQL Injection

b) Buffer overflow attacks

c) Cross-Site Scripting (XSS)

d) Firewall bypass

3. In ROP, what are "gadgets"?

a) Hardware components for hacking

b) Debugging tools

c) Existing code snippets ending in return instructions

d) Injected malicious code

4. Jump-Oriented Programming (JOP) relies on:

a) Return instructions

b) Call instructions

c) Encryption keys

d) Jump instructions

5. Function pointer overwriting allows attackers to:

a) Encrypt sensitive data

b) Redirect execution to malicious code

c) Improve program speed

d) Bypass network firewalls

Code Injection

6. Code injection exploits often target:
   a) Input fields in applications
   b) Hardware firmware
   c) Network cables
   d) Screen resolution settings
7. Which attack manipulates databases via malicious queries?
   a) Cross-Site Scripting (XSS)
   b) Buffer Overflow
   c) SQL Injection
   d) ROP
8. XSS attacks primarily endanger:
   a) Server hardware
   b) Database administrators
   c) Network routers
   d) Web application users
9. Command injection exploits:
   a) A server's command-line interface
   b) User passwords
   c) Graphics rendering
   d) CPU clock speed
10. The eval() function in PHP/JavaScript is risky because it:
   a) Encrypts user data
   b) Deletes files
   c) Executes arbitrary input as code
   d) Slows down the system

Code Reuse Attacks

11. Code reuse attacks are dangerous because they:
    a) Corrupt hardware
    b) Bypass DEP/ASLR by reusing existing code
    c) Overwrite BIOS settings
    d) Disable monitors
12. ROP chains are built using:
    a) Injected shellcode
    b) Firewall rules
    c) Return instructions from existing code
    d) Password hashes
13. JOP differs from ROP by using:
    a) Call instructions
    b) Jump instructions
    c) Debugging symbols
    d) Memory compression
14. COP manipulates:
    a) Database schemas
    b) Screen pixels
    c) Mouse movements
    d) Function call sequences
15. Which defense does ROP/JOP bypass?
    a) Antivirus scans
    b) Password policies

c) Screen locks

d) Data Execution Prevention (DEP)