Software Security

Attack Vectors modules

Dr. Zeyad Safaa Younus Saffawi

Attack Vectors

- An **attack vector**, or **threat vector**, is a method that attackers use to gain unauthorized access to a network or system.
- Common attack vectors include **social engineering attacks, credential theft, vulnerability exploits, and insufficient protection against insider threats**.
- **To illustrate this concept, consider the following example:**
  - Suppose a security firm is tasked with guarding a rare painting displayed in a museum. There are several ways a thief could enter or exit the museum, such as through the **front doors, back doors, elevators, or windows**. A thief could also gain access by **posing as a member of the museum staff**.
  - All of these methods represent **attack vectors**. To prevent theft, the security firm might place guards at the doors, install locks on windows, and regularly verify the identities of museum staff.

- Similarly, **digital systems also have multiple entry points that attackers may attempt to exploit**. Because modern computing systems and application environments are highly complex, it is usually impossible to eliminate all attack vectors completely. However, **strong security practices and safeguards can reduce most attack vectors**, making it much more difficult for attackers to exploit them.

Attack Vectors

1. A Denial of Service (DoS) attack occurs when legitimate users are unable to access information systems, devices, or other network resources due to the actions of a malicious attacker.

- Affected services may include:
  - Email systems
  - Websites
  - Online accounts (such as banking services)
  - Other services that rely on the affected computer or network
- A DoS attack is typically accomplished by **flooding the targeted host or network with excessive traffic** until the **target system becomes unable to respond or crashes completely preventing access for legitimate users.**
- **DoS attacks** can **cost** an organization both **time and money** while their **resources and services** are **inaccessible**.

Attacker
A cyber attacker infects the devices of compromised internet users, creating a network of bots.

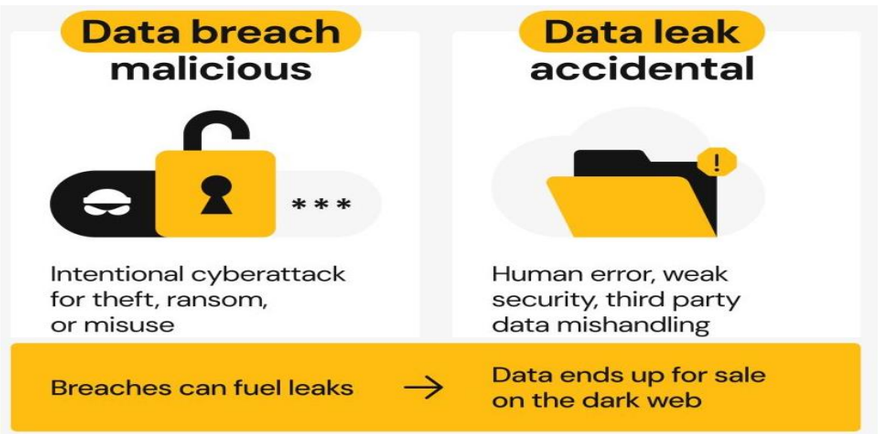
Network of bots
A large number of infected bot devices flood the targeted system with multiple requests.

Targeted system
The attacked server, website, or application is overwhelmed with traffic, resulting in a system slowdown or crash.

Attack Vectors

2. Information leakage refers to the exposure or sharing of sensitive information with unauthorized parties.

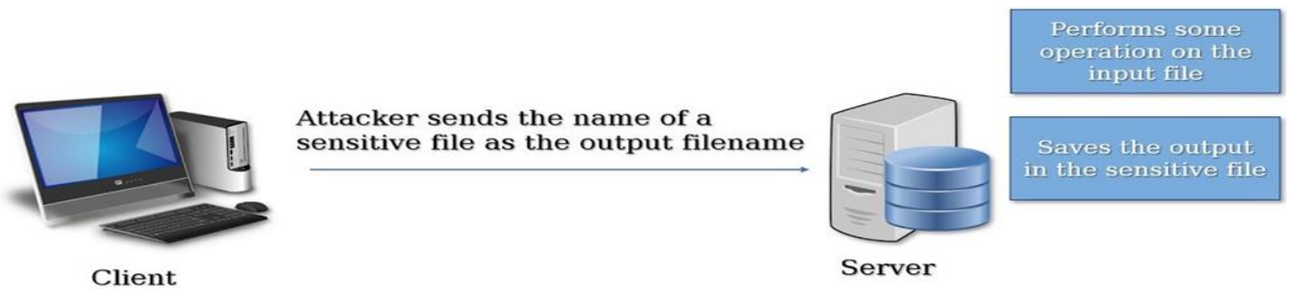
- Information leakage can occur in two ways:
  1. **Accidental leakage**, such as an **employee sharing confidential information** with an **external party via email**.
  2. **Malicious leakage**, such as the **exfiltration of data through phishing scams**.
- Regardless of the intent, the **information shared is valuable** to hackers and can be used to **execute attacks** on your organization's **infrastructure, services or applications**.
- While **information leaks originate** from **within an organization**, **data breaches** are a result of **actions** that **take place** from **unauthorized users** from **outside of the organization**.
- Organizations use several **techniques** to **prevent data leakage**, including:
  - **Data encryption**
  - **Implementing strong security controls**
  - **Classifying sensitive data**



Attack Vectors

3. Confused Deputy problem is a security issue that occurs when a program (the **deputy**) is **tricked into performing actions on behalf of an attacker with more privileges than the attacker should have**.

- This problem often occurs in **systems where one program (the deputy) performs tasks** on **behalf of another program or user, and the deputy is tricked into executing actions it would not normally perform**.
- Here's **a simplified example** of **how the Confused Deputy problem** might a rise:
  1. **Scenario:** Consider a web application that allows users to upload files. The application processes these files on behalf of users, using a server side script with permissions to read and write files in a specific directory.
  2. **Exploitation:** An attacker may upload a file with a name that includes a path traversal attack (e.g., ../sensitive_file.txt). The application's script, operating with the server's higher privileges, might process this file and inadvertently read or write to sensitive files outside the intended directory.
  3. **Result:** The attacker successfully accesses or modifies sensitive files by exploiting the higher privileges of the server-side script.



Attack Vectors

4. Privilege escalation is a type of security **vulnerability that allow an attacker to gain expanded access to resources or permissions beyond what is normally authorized, leading** to **unauthorized access to sensitive data, system control, or administrative functions**.

- Privilege escalation can be **classified** into two main types:

1. **Vertical privilege escalation** occurs when a **user with lower-level permissions** gains **access to higher-level privileges or administrative rights**.

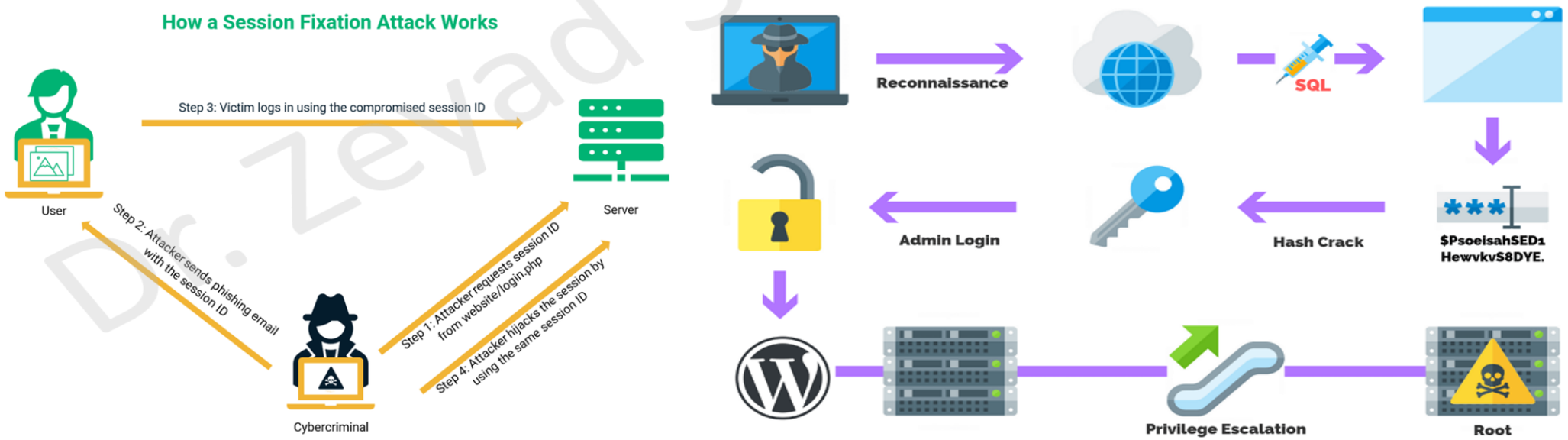
- For **example**:
  - **Exploiting Vulnerabilities**: Attackers might exploit software vulnerabilities to gain administrative access. For instance, vulnerability in an application might allow a user to execute commands with root privileges.
  - **Weak Authentication**: Weak or misconfigured authentication mechanisms can allow an attacker to impersonate an administrative user and gain elevated access.
  - **Misconfigured Permissions:** Incorrectly configured file or directory permissions can allow a user to access or modify files without authorization.

Privilege Escalation

2. Horizontal privilege escalation happens when a **user gains access to resources or permissions of another user with the same level of privilege**.

For **example**:

- **Session Fixation**: An attacker **might hijack a user's session to access resources** that the **user can access**, even though the **attacker shouldn't have access to get those resources**.
- **Insecure APIs**: APIs that do not enforce proper access controls may allow an attacker to perform actions intended for other users with similar permissions.



Attack Vectors

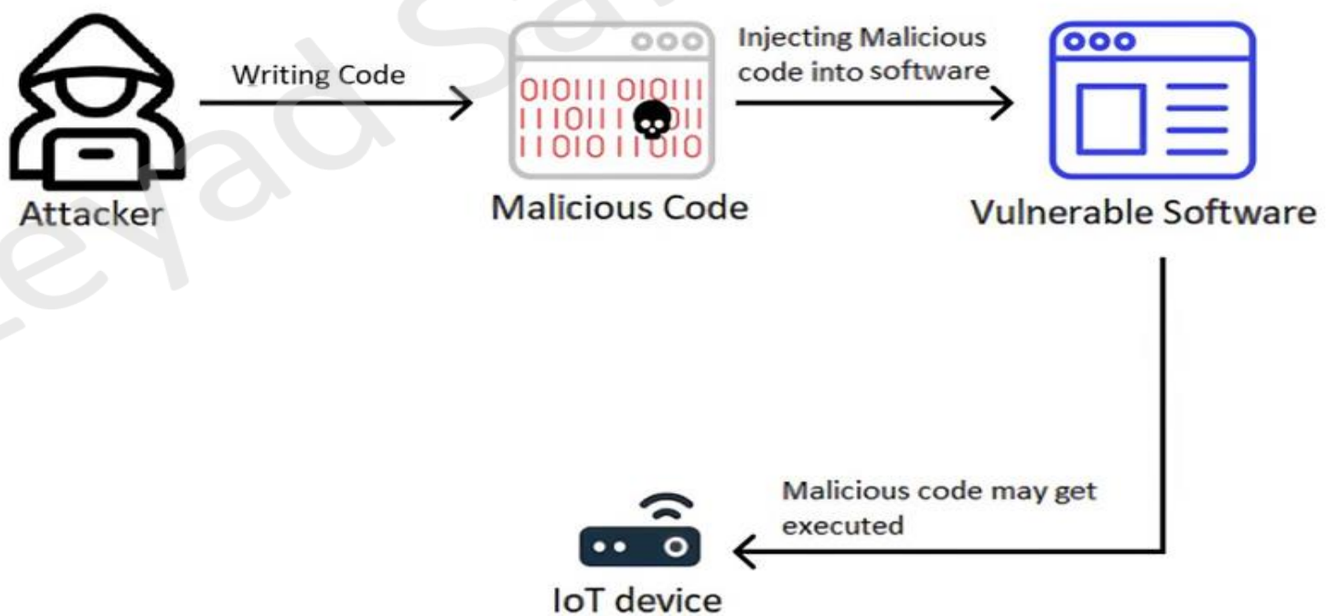
5. Control-flow hijacking is a type of security vulnerability that **allows an attacker to alter the flow of execution in a program**.

- 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** (tools) 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**.

Attack Vectors

6. 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**.



Attack Vectors

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**.

Attack Vectors

7. 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 existing 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**.

Attack Vectors

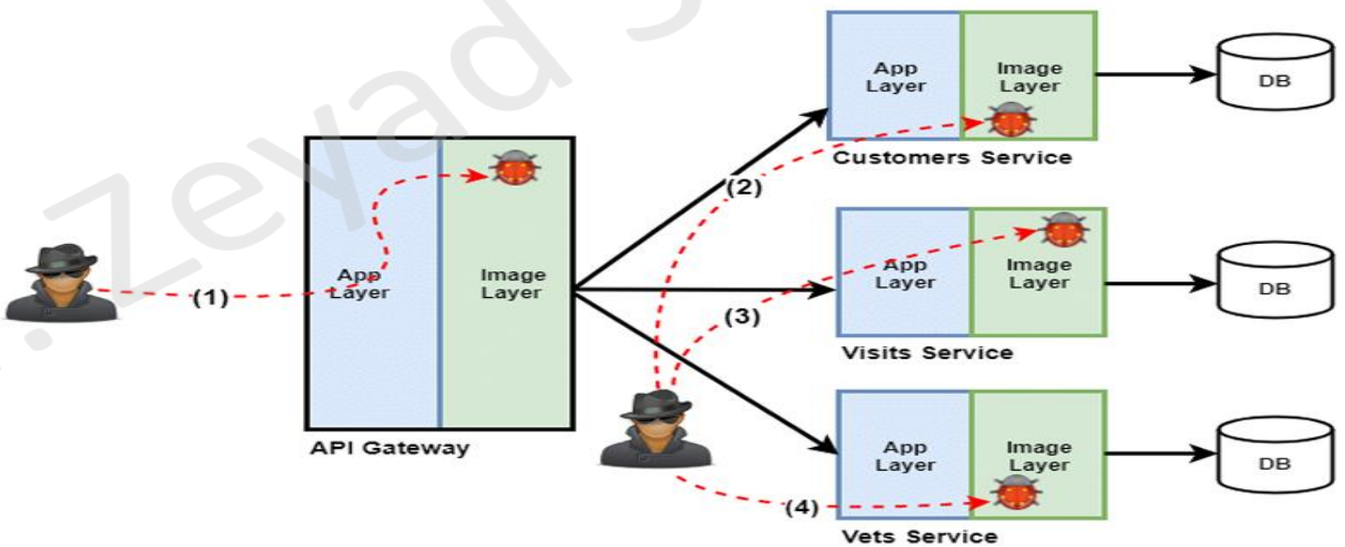
Common Types of Code Reuse Attacks:

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.

Attack Vectors

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**.



References

- Payer, M. (2021). *Software security: Principles, policies, and protection* (Version 0.37).