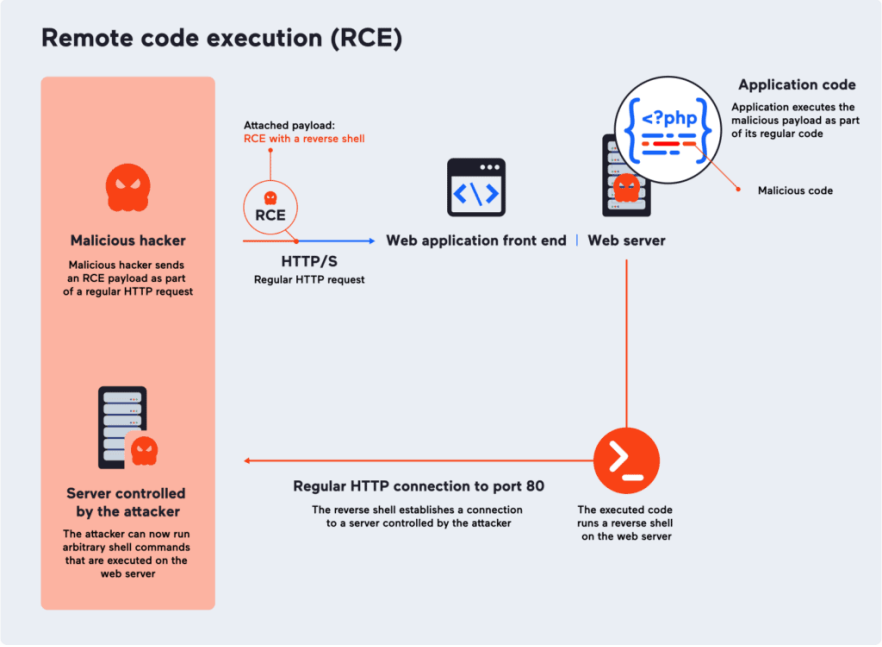
# RCE

## Remote Code Execution

Remote Code Execution (RCE) represents a critical vulnerability wherein user input infiltrates a file or string, subsequently executed by the programming language's parser. Such behavior is typically unintended by web application developers. RCE exploits can unleash full-scale attacks, jeopardizing both the web application and server security. Moreover, RCE can facilitate privilege escalation, network pivoting, and persistence establishment, amplifying its severity to a high or critical level. It's crucial to recognize that nearly all programming languages incorporate distinct code evaluation mechanisms.

Allowing user inputs to access functions evaluating code within the same programming language can also trigger code evaluation. This may occur intentionally to access mathematical functions or inadvertently when developer-designed user inputs reside within these functions. Engaging in such actions is ill-advised, as many perceive code evaluation as malicious and untrustworthy.



Various methods exist for executing remote code, each targeting different layers of a server. A prevalent approach involves injecting code to seize control over the instruction pointer. This enables attackers to direct the execution flow towards specific instructions or processes. Code injection can occur through various means and locations, but attackers must effectively direct the execution towards the injected code to activate it. The injected code may take the form of a command, a script, or other executable content.

- **Injection Attack**: Many applications accept user input to execute commands. Attackers exploit this vulnerability by providing deliberately malformed input data to execute arbitrary code.
- **Deserialization Attack**: Applications often utilize serialization for data organization in communication. Deserialization processes may mistakenly interpret user-supplied serialized data as executable code.
- **Out-of-Bounds Write**: Applications frequently allocate fixed memory chunks for data storage. Flaws in memory allocation can enable attackers to provide inputs that overwrite outside the buffer, potentially including executable or malicious code.
- **Remote Code Evaluation (RCE)**: Certain functions that evaluate code accept user input, making them vulnerable to exploitation. For example, some applications permit users to create variable names using their usernames, allowing for the inclusion of malicious code.
- **Stored Code Evaluation**: This method differs from typical RCE as it relies on the interpreter parsing files instead of specific language functions. Interpreters should not execute files with user input. However, web applications with upload functionality may fail to sufficiently validate uploaded files, leading to potential exploitation.

## Some Examples of RCE Vulnerabilities:

1. **Apache Struts (Equifax Breach)**: The Apache Struts framework was involved in a massive data breach at Equifax. Attackers exploited a vulnerability in the framework's Jakarta Multipart parser, allowing for remote code execution. This incident resulted in the exposure of sensitive personal information of millions of individuals.

2. **Shellshock (Bash)**: The Shellshock vulnerability emerged in the Bash shell. Attackers exploited a flaw in the way Bash processed environment variables, enabling arbitrary code execution. This vulnerability impacted various Unix-based operating systems and Internet-facing servers, potentially affecting a wide range of systems.
3. **Eternal Blue (WannaCry Ransomware)**: Eternal Blue is an exploit developed by the NSA and later leaked by the Shadow Brokers hacking group. It targeted a vulnerability in Microsoft's Server Message Block (SMB) protocol implementation. Eternal Blue facilitated the rapid spread of the WannaCry ransomware, infecting hundreds of thousands of computers worldwide and causing widespread disruption.
4. **Heartbleed (OpenSSL)**: Heartbleed was a critical vulnerability in the OpenSSL cryptographic library. It allowed attackers to exploit a flaw in the implementation of the Transport Layer Security (TLS) protocol's heartbeat extension, potentially leaking sensitive information from the server's memory, including private keys. This vulnerability affected a large portion of web servers on the Internet, highlighting the widespread impact of RCE vulnerabilities in foundational software libraries.
5. **Drupalgeddon (Drupal)**: Drupalgeddon refers to a series of critical vulnerabilities discovered in the Drupal content management system. One of the most notable incidents occurred with the Drupalgeddon 2 vulnerability, allowing remote attackers to execute arbitrary code on vulnerable Drupal installations. This vulnerability affected millions of websites powered by Drupal, emphasizing the importance of promptly patching and securing content management systems.
6. **Adobe ColdFusion**: In the cybersecurity landscape of 2023, one prominent vulnerability that sent shockwaves across organizations was CVE-2023-26360 within Adobe ColdFusion. This vulnerability stemmed from a deserialization flaw in handling untrusted data, opening a gateway for remote code execution.
7. **Microsoft Windows (CVE-2023-36884):** Microsoft Windows is marked as a zero-day vulnerability, initially disclosed in the July 2023 Patch Tuesday by Microsoft. Originally perceived as a remote code execution vulnerability,

the situation took an intriguing turn when it was addressed during Microsoft's August 2023 Patch Tuesday, and the flaw was reclassified as a security feature bypass. The intricate nature of this vulnerability involved the exploitation of Microsoft Office and its defense mechanisms. This high-severity vulnerability specifically targeted Microsoft Windows Search, rooting from an unspecified flaw that, when exploited, allowed an attacker to bypass Mark of the Web (MOTW) defenses using a specially crafted Microsoft Office document. The consequence was a pathway to remote code execution, presenting a serious threat to system integrity.

## Escalating vulnerability to perform RCE (Remote Code Execution):

1. RCE via Dependency Confusion
2. RCE via File Upload
3. RCE via SQL Injection
4. RCE via LFI
5. RCE via SSRF
6. RCE via XXE
7. RCE via SSTI
8. RCE via Command Injection
9. RCE via Insecure Deserialization

## Impact of Remote Code Execution:

**Data Breach Risk:** Remote Code Execution (RCE) poses a significant risk of unauthorized access to sensitive data, potentially leading to data breaches where personal, financial, or proprietary information is compromised.

**System Compromise**: RCE vulnerabilities enable attackers to take control of systems, allowing them to execute malicious activities such as spreading malware, launching further attacks, or seizing resources.

**Supply Chain Risks**: RCE vulnerabilities in third-party software or components pose risks to the entire supply chain, allowing attackers to gain access to systems and data across interconnected organizations.

**Reputation Damage**: Falling victim to RCE attacks damages an organization's reputation, eroding trust among customers, partners, and stakeholders, leading to loss of business opportunities and credibility.

**Operational Disruption**: RCE attacks disrupt normal business operations, causing downtime and productivity losses, which can be particularly severe in critical infrastructure sectors.

## Mitigation:

- Regularly update software and systems with security patches released by vendors to address known vulnerabilities, including those that could lead to RCE.

- Conduct regular code reviews and use static analysis tools to identify potential vulnerabilities in software code, including those that could lead to RCE, and address them before deployment.

- RCE attackers can also exploit issues with memory management, such as buffer overflows. Applications should undergo vulnerability scanning to detect buffer overflow and other vulnerabilities to detect and remediate these errors

- Implement network segmentation to restrict access to critical systems and resources, reducing the potential impact of RCE attacks by limiting the attacker's lateral movement within the network.

- Deploy WAFs to filter and monitor HTTP traffic to web applications, helping to detect and block attempts to exploit vulnerabilities that could lead to RCE.

## References:


https://www.bugcrowd.com/glossary/remote-code-execution-rce/

https://www.cloudflare.com/learning/security/what-is-remote-code-execution/

https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

https://hackerone.com/reports/678727

https://airman604.medium.com/from-xxe-to-rce-with-php-expect-the-missing-link-a18c265ea4c7

https://medium.com/r3d-buck3t/rce-with-server-side-template-injection-b9c5959ad31e

https://secure-cookie.io/attacks/insecuredeserialization/