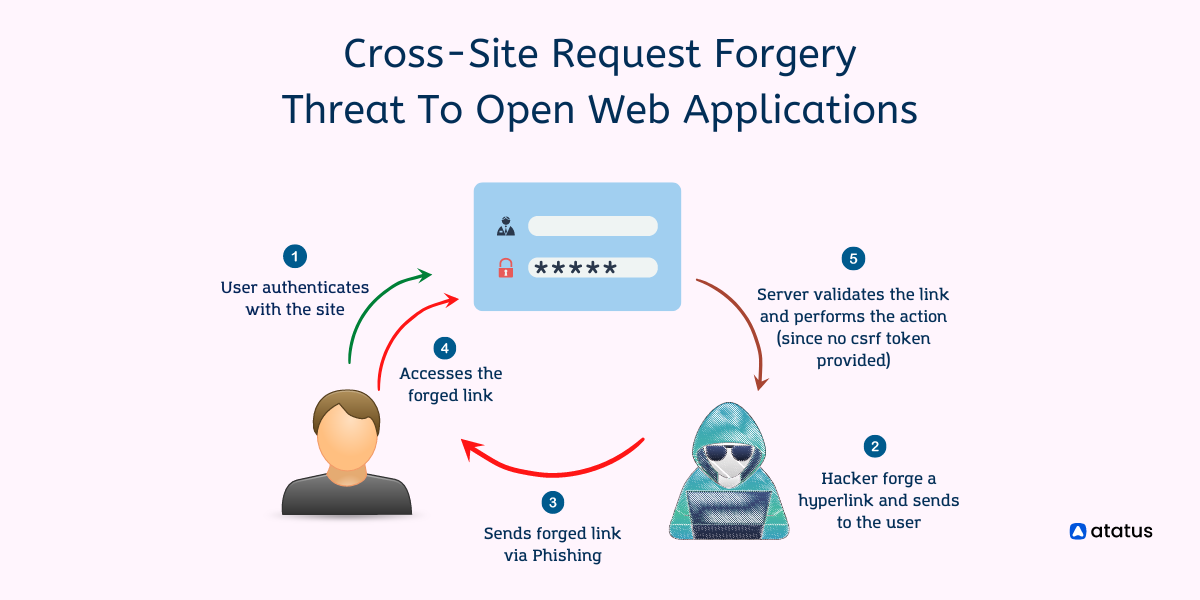
1. **Cross-Site Request Forgery (CSRF)**

Description: CSRF attacks trick users into performing unintended actions on a website when they are logged in. Attackers use social engineering to execute actions without the user's knowledge or consent.

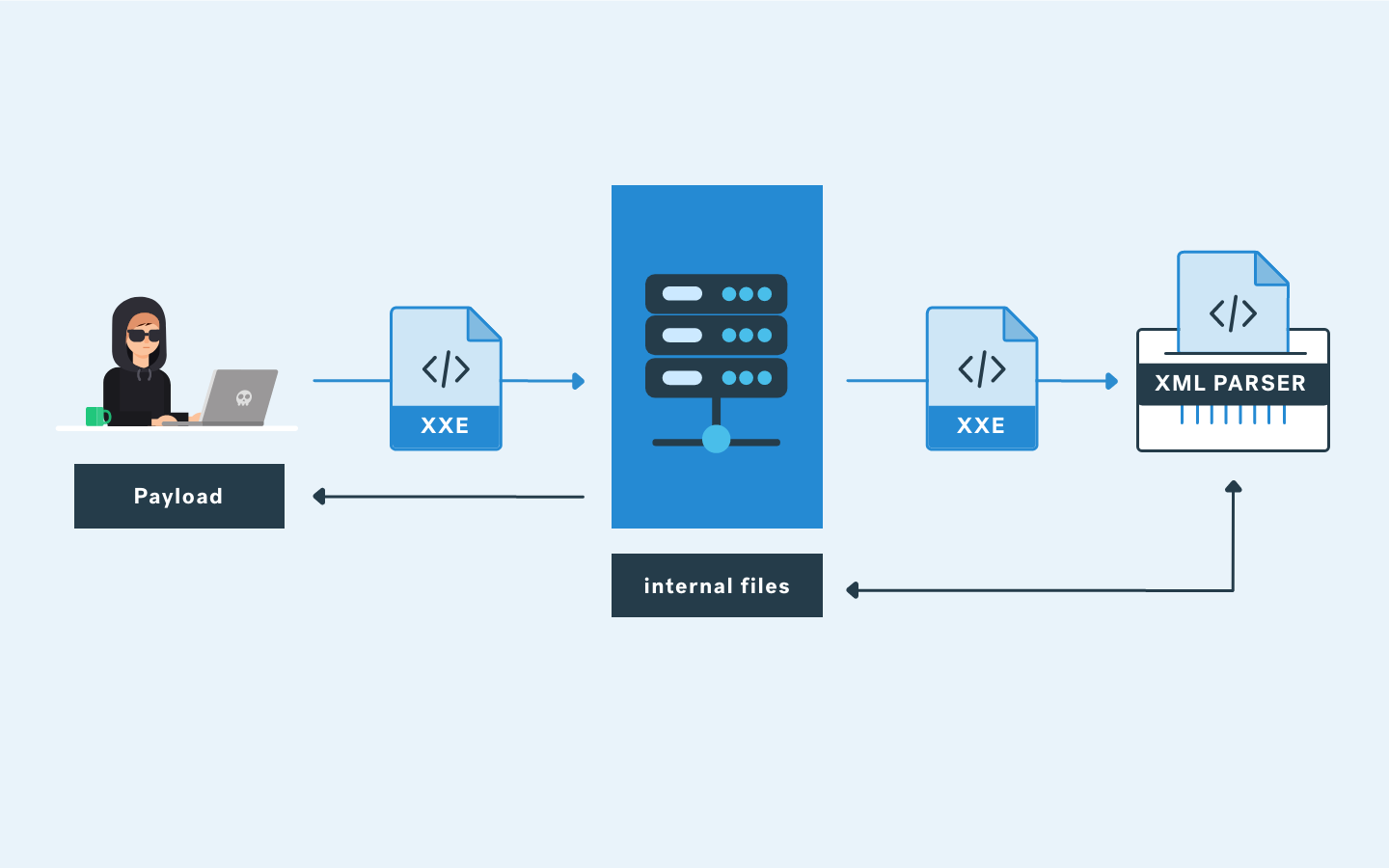
Mitigation: Implement anti-CSRF tokens, validate requests, and enforce same-origin policies.



1. **XML External Entity (XXE)**

Description: XXE vulnerabilities occur when an application processes XML input from untrusted sources. Attackers can exploit this to read internal files, perform DoS attacks, or execute remote code.

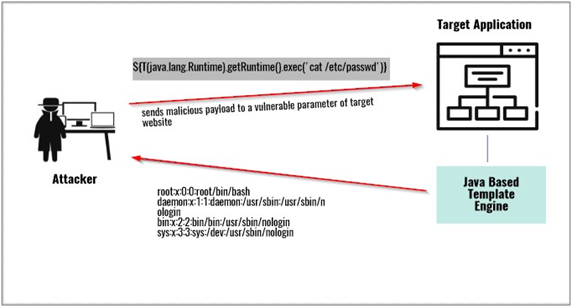
Mitigation: Disable XML entity expansion, use safer XML parsers, and validate XML input.



1. **Server-Side Template Injection (SSTI)**

Description: SSTI occurs when user input is injected into server-side templates. Attackers can manipulate templates to execute arbitrary code on the server.

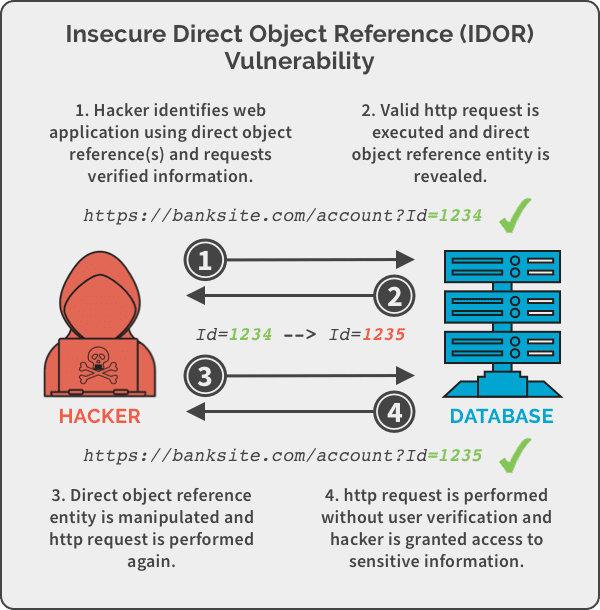
Mitigation: Avoid dynamic template rendering, sanitize user inputs, and validate template expressions.



1. **Insecure Direct Object References (IDOR)**

Description: IDOR vulnerabilities arise when an attacker can manipulate references to access unauthorized resources. This often occurs when developers rely on client-provided data for resource access.

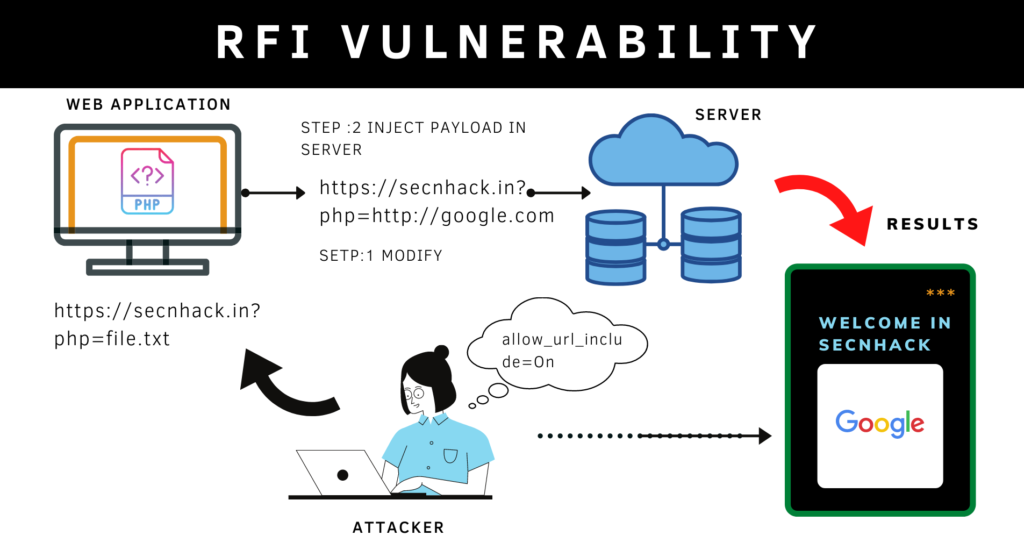
Mitigation: Implement proper authorization checks, use unique identifiers for resources, and validate user access rights.



1. **Remote File Inclusion (RFI)**

Description: RFI vulnerabilities allow attackers to include remote files, potentially leading to remote code execution. This often happens when an application includes files based on user input.

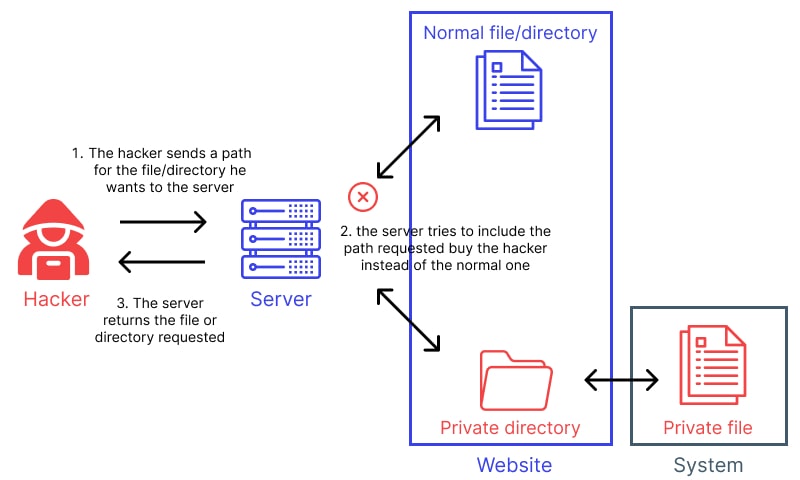
Mitigation: Avoid using user-controlled input for file inclusion, use whitelists, and implement input validation.



1. **Path Traversal**

Description: Path Traversal vulnerabilities occur when an attacker manipulates file paths to access files or directories outside the intended scope. This can lead to unauthorized data disclosure or code execution.

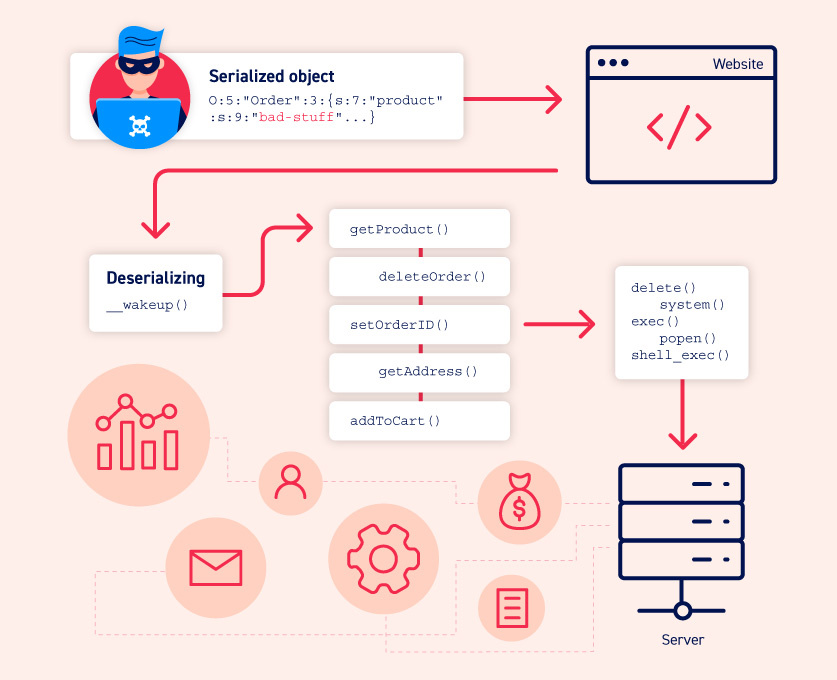
Mitigation: Validate and sanitize file paths, implement proper access controls, and restrict directory access.



1. **Insecure Deserialization**

Description: Insecure Deserialization vulnerabilities can result in remote code execution and other attacks when an application process serialized data from untrusted sources.

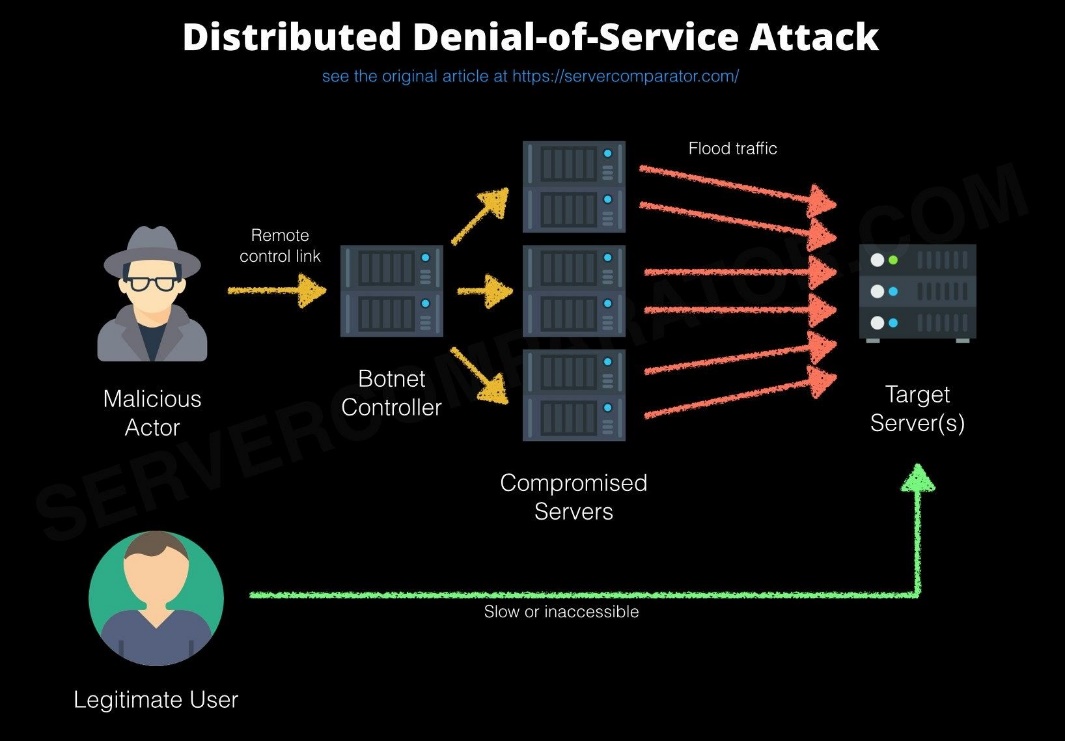
Mitigation: Avoid deserializing untrusted data, validate serialized input, and use safe deserialization libraries.



1. **Denial of Service (DoS)**

Description: Denial of Service attacks disrupt the availability of a system or application, making it inaccessible to users. These attacks overwhelm resources or exploit vulnerabilities to crash a system.

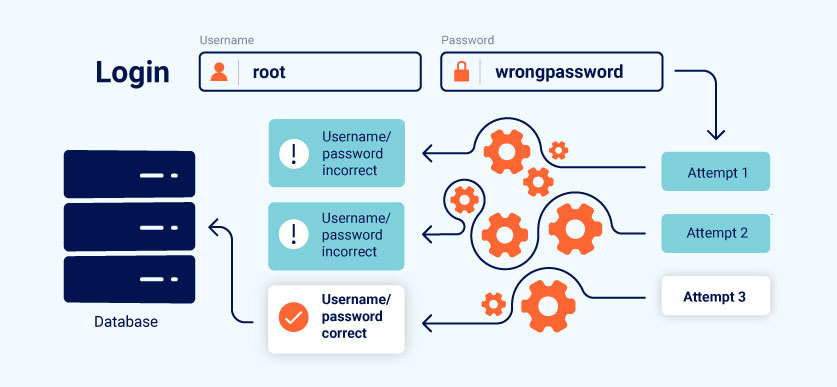
Mitigation: Implement rate limiting, use intrusion detection systems, and keep software and systems patched to mitigate DoS attacks.



1. **Security Bypass via Business Logic Flaws**

Description: These vulnerabilities involve flaws in the logic of an application that allow attackers to bypass security checks. This can lead to unauthorized access or data manipulation.

Mitigation: Carefully design and test business logic to ensure proper security checks and authorization are enforced.



1. **DOM-based Cross-Site Scripting (DOM XSS)**

Description: DOM XSS vulnerabilities occur when an application dynamically generates or manipulates the Document Object Model (DOM) based on untrusted data. Attackers can inject malicious scripts into the DOM, leading to XSS attacks.

Mitigation: Sanitize and validate client-side input, escape output, and implement Content Security Policy (CSP) headers.

