The World of Server-Side Request Forgery

In this article, you'll learn the concept of Server-Side Request Forgery.

What We'll Be Learning

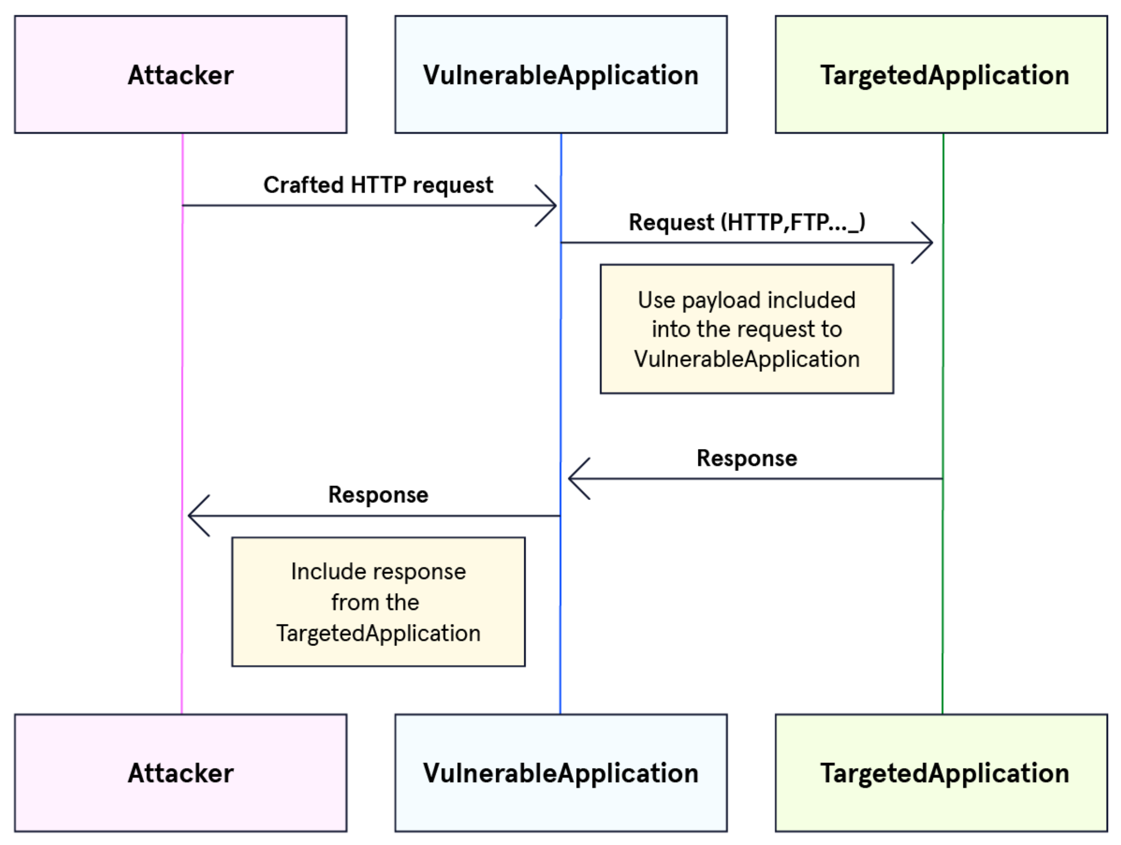
The Open Web Application Security Project (OWASP) identified one specific vulnerability attributed to a few major cybersecurity incidents in recent years and assigned it its own place in the OWASP Top 10. Category A10 of the 2021 OWASP Top 10 covers the Server-Side Request Forgery (SSRF) vulnerability. Due to the effectiveness of SSRF attacks, OWASP emphasizes SSRF to improve awareness and security of the vulnerability. In this article, we will:

- Define and understand SSRF Attacks.
- Understand how SSRF attacks flow.
- Learn why SSRF attacks are successful.
- Understand why SSRF attacks are dangerous.
- Explore how SSRF attacks may be prevented.

What is a Server-Side Request Forgery?

Server-Side Request Forgery

To understand server-side request forgery vulnerabilities, it is best to visualize the common flow of a Server-Side Request Forgery attack [see figure below].



A server-side request forgery attack takes advantage of an application (hosted on an application server or service) to interact maliciously with other servers, networks, or the host server. In the diagram above, the flow of the SSRF attack initiates at the attacker line, then passes a payload through the vulnerable application to the targeted application.

In the SSRF attack, a malicious HTTP request is sent to a vulnerable application. The crafted HTTP request contains a payload that executes on the vulnerable application server which facilitates another request targeting a different application or server. Once the payload is executed, the results of the executed payload is returned to the attacker through the vulnerable application.

How do SSRF Come to Be?

SSRF vulnerabilities are enabled through the lack of strong network access security policy implementation, server hardening, and input validation. By simply preventing user input from manipulating internal applications, an SSRF attack could be avoided. Victim systems of SSRF attacks overlook the trust given to the user's input, allowing the user's input to perform unauthorized actions.

Why are SSRF Attacks Dangerous?

SSRF attacks are dangerous since a single malicious HTTP request can allow an attacker to instruct the back end to perform malicious actions. There are many capabilities that the SSRF attack vector gives the attacker that makes them dangerous, such as:

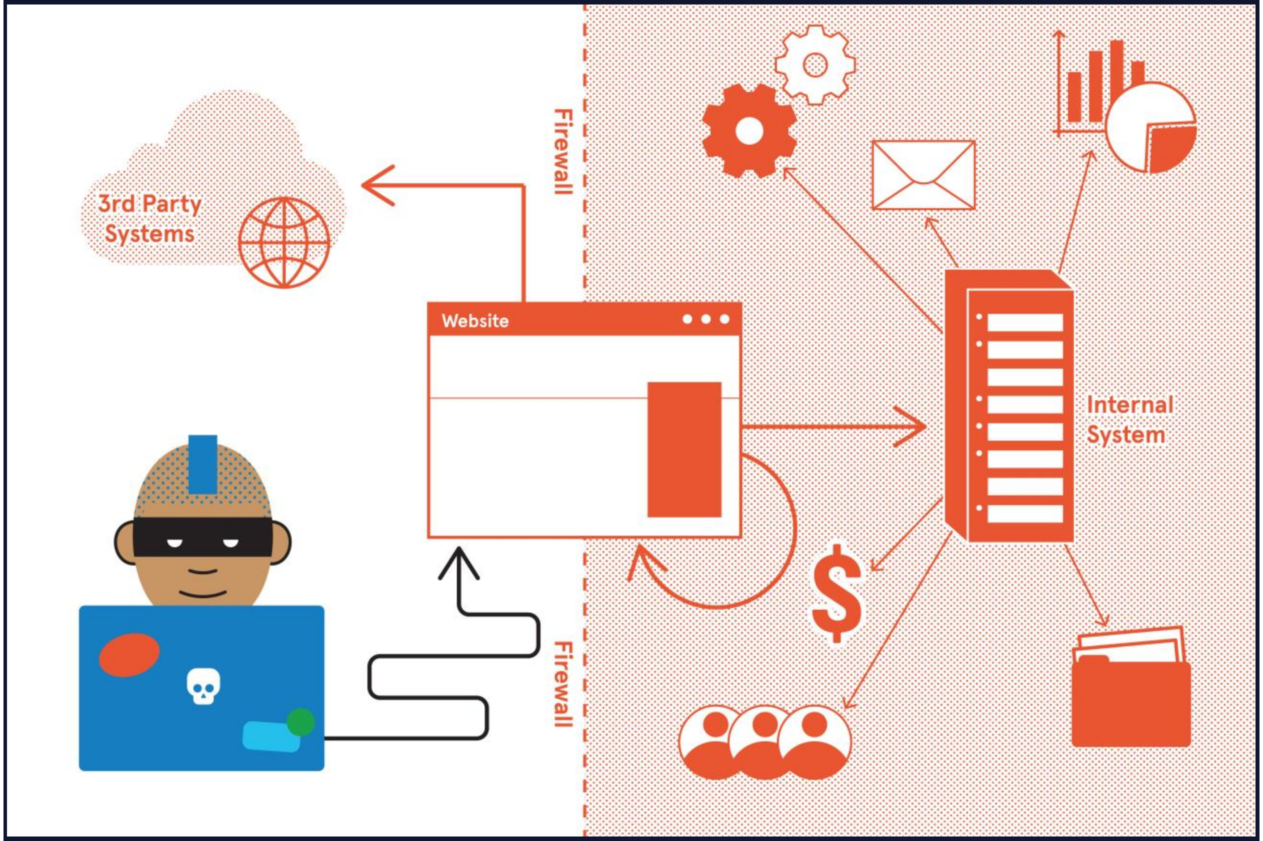
- Port scans of other machines on the target network. Successful SSRF attacks may allow attackers to scan other systems for vulnerabilities through scanning tools like NMAP or PowerShell scripts.
- Retrieve server files.
- Evade firewalls on target systems to perform malicious requests.
- Remote file inclusion attacks.
- Identify running services on target servers.

SSRF Prevention

The best prevention tactic that protects against SSRF attacks is the implementation of network access control policies. Strong network access policies will incorporate whitelisting techniques for internal IP addresses and DNS responses. Well-implemented network access control policies will block all external traffic sent to an application or web server by default and only allow specific whitelisted protocols and ports to communicate internally. Validation of client-supplied input data should also be considered as a prevention technique. Input validation is handled at the application layer to identify inputs and HTTP responses that may contain payloads. An emphasis should be placed on HTTP response handling techniques, given the nature of SSRF techniques. Additionally, servers internal to the public-facing web and application servers must be hardened to eliminate any additional attack vectors. If not needed for production operations, unnecessary URLs may be disabled on any system in the application tier. For example, the URLs `file://` or `ftp://` may not be necessary and should be disabled on back-end servers to avoid remote file retrieval of an SSRF attack.

Lastly, implementing the concept of zero trust is another practice that may significantly improve security against SSRF attacks. Zero trust will enforce the authentication of services from external and internal sources. With zero trust implemented correctly within a network, SSRF actions may be blocked from target back-end systems.

Conclusion



SSRF attacks exploit vulnerable applications and systems through malicious input and HTTP responses. As a result, the exploited system performs malicious actions on the compromised system or back-end servers. SSRF attacks are dangerous for many reasons, including remote file retrieval and command execution. To prevent SSRF attacks, a handful of techniques may be implemented, such as zero trust and HTTP response handling.