Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) is **a web security vulnerability that allows an attacker to manipulate a server to send unauthorized requests to other internal or external resources**. The attacker can exploit this to access internal systems, sensitive data, or services not directly exposed to the internet.

1. How SSRF Works

In an SSRF attack, the attacker **tricks the server into making HTTP or other protocol-based requests to a target resource by supplying a malicious URL or payload**. Since the server initiates the request, it bypasses network restrictions and appears to originate from a trusted source.

Vulnerable Code Example

```
<?php
    $url = $_GET['url'];
    $response = file_get_contents($url);
    echo $response;
?>
```

Legitimate Input:

```
http://example.com/proxy?url=http://trusted-site.com
```

Malicious Input:

```
http://example.com/proxy?url=http://internal-service.local/admin
```

- Exploits the server's ability to reach http://internal-service.local.

2. Common Targets and Impacts of SSRF

a. Common Targets

1. **Internal Services**

- Access internal APIs, databases, or administrative interfaces (http://127.0.0.1, http://169.254.169.254).

2. **Cloud Metadata Services**

- Extract sensitive information like credentials or tokens from cloud services.
- Example
    - AWS Metadata URL: http://169.254.169.254/latest/meta-data/iam/security-credentials/

3. **File Systems**

- Exploit protocols like file:// to read local files.

4. **Other Protocols**

- Exploit protocols like gopher://, ftp://, or smtp:// to perform advanced attacks.

b. Impact

1. **Information Disclosure**

- Access sensitive internal resources or services.

2. **Port Scanning**

- Scan the internal network to identify open ports or services.

3. **Bypass Firewall Rules**

- Exploit the server's ability to access restricted resources.

4. **Remote Code Execution (RCE)**

- Chain SSRF with other vulnerabilities to achieve RCE.

3. Exploitation Techniques

a. Basic SSRF Exploitation

1. **Access internal resources**

```
http://example.com/proxy?url=http://127.0.0.1:8080/admin
```

2. **Extract cloud metadata**

```
http://example.com/proxy?url=http://169.254.169.254/latest/meta-data/
```

b. Advanced SSRF Techniques

1. **Protocol Abuse**

- Use unconventional protocols like gopher:// for advanced attacks:

```
gopher://127.0.0.1:3306/_SHOW%20DATABASES
```

2. **Chaining Attacks**

- Combine SSRF with Remote File Inclusion (RFI) or deserialization vulnerabilities.

3. **DNS Rebinding**

- Trick the server into resolving an external domain to an internal IP.

4. Example Scenarios

a. Cloud Metadata Access

Target

- AWS metadata service at http://169.254.169.254.

Malicious Input

```
http://example.com/proxy?url=http://169.254.169.254/latest/meta-data/
```

Impact

- Extract AWS credentials, tokens, or configuration.

5. Mitigation Techniques

a. Validate and Sanitize User Input

- **Only allow specific, trusted URLs using a whitelist**.
- **Reject suspicious patterns** like 127.0.0.1, 169.254.169.254, or internal hostnames.

b. Restrict Network Access

- **Isolate server roles** to limit their ability to make external or internal requests.
- **Use firewalls** to block access to private IP ranges.

c. Use Allowlists

- Allow requests only to explicitly approved domains or IP addresses.

d. Avoid Direct User Input in Requests

- Use indirect identifiers (e.g., a resource ID instead of a full URL).

e. Disable Unnecessary Protocols

- Restrict the server from accessing non-HTTP protocols like file://, gopher://, or ftp://.

f. Monitor and Log Outgoing Requests

- Track and analyze unusual patterns in outgoing requests.

6. Tools for Detection

1. **Burp Suite**

- Proxy malicious requests to test for SSRF vulnerabilities.

2. **OWASP ZAP**

- Automated scanning for SSRF issues in web applications.

3. **SSRFmap**

- A specialized tool for automating SSRF payload testing.

4. Manual Testing

- Craft custom payloads for vulnerable endpoints.

7. Common SSRF Payloads

Basic Payloads

1. Internal resources

```
http://127.0.0.1/admin
```

2. Cloud metadata

```
http://169.254.169.254/latest/meta-data/
```

Advanced Payloads:

1. File access

```
file:///etc/passwd
```

2. Protocol abuse

```
gopher://127.0.0.1:6379/_PING
```

8. Summary

Aspect	Details
What is SSRF?	A vulnerability allowing attackers to manipulate server-side requests.

Aspect	Details
Common Targets	Internal APIs, cloud metadata, local files, other protocols.
Impacts	Information disclosure, port scanning, RCE, firewall bypass.
Mitigation Techniques	Input validation, network restrictions, allowlists, monitoring.
Tools	Burp Suite, OWASP ZAP, SSRFmap.

SSRF is a critical vulnerability with the potential for severe consequences, particularly in environments like cloud infrastructure. By **validating user input**, **restricting network access**, and **monitoring server behavior**, organizations can mitigate the risks associated with SSRF. Security teams must remain vigilant, as attackers continuously develop new techniques to exploit this vulnerability.