

Server Side Request Forgery (SSRF)

Information Security Inc.

Contents

- About SSRF
- Topology
- Demo
- Mitigations
- References

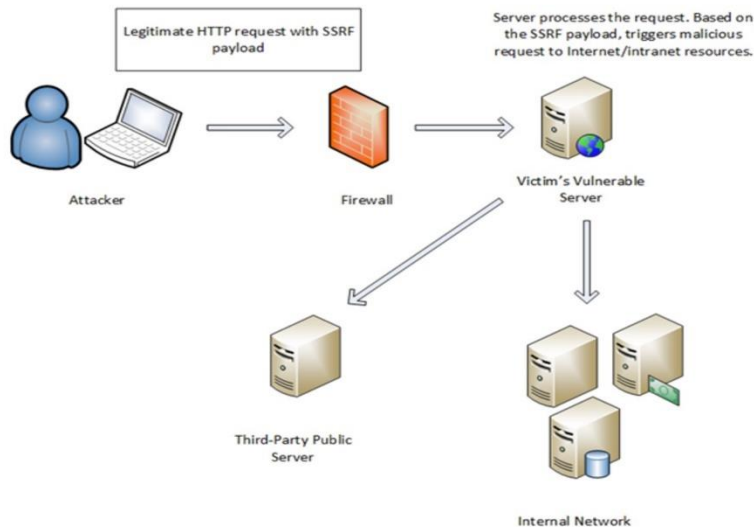
About SSRF

© SSRF - Server Side Request Forgery.

The ability to create requests from the vulnerable server to intra/internet. Using a protocol supported by available URI schemas, you can communicate with services running on other protocols

© application creates internal requests, HTTP requests using `HttpClient / URL.openConnection()`

© attacker can change the internal request to other location or use different parameters



Demo


- ◎ Vulnerable server runs a web server on 4.6.0-kali1-amd64 (Debian based)
- ◎ A vulnerable PHP script `/* Vulnerable.php */` is uploaded
- ◎ Script in browser



← → ⓘ 192.168.10.12/SSRF/Vulnerable.html

よく見るページ Firefox を使いこなそう How to inte

SSRF Demo



Enter the URL to wish to view!



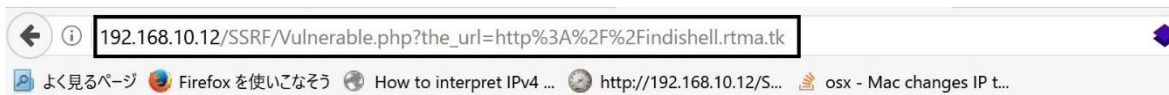
Vulnerable.html



Vulnerable.php

Demo

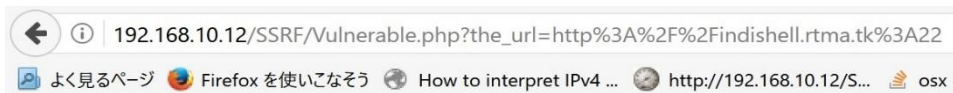
© Loading <http://indishell.rtma.tk> and can see



Welcome to nginx!

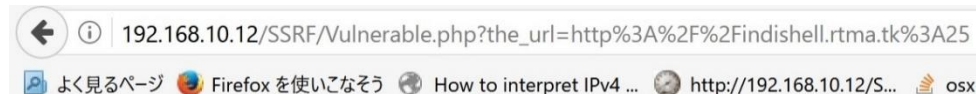
© Checking if the server validates user input or not

▲ URL: `http://indishell.rtma.tk:22`



SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.10

▲ URL: `http://indishell.rtma.tk:25`



220 indishell ESMTP Postfix (Ubuntu) 221 2.7.0 Error: I can break rules, too. Goodbye.

Demo

© Get /etc/passwd from the vulnerable server



Mitigations

- **Error handling and messages** – Display generic error messages to the client in case something goes wrong. If content type validation fails, display generic errors to the client like “Invalid Data retrieved”. Also ensure that the message is the same when the request fails on the backend and if invalid data is received. This will prevent the application from being abused as distinct error messages will be absent for closed and open ports. Under no circumstance should the raw response received from the remote server be displayed to the client.
- **Response Handling** – Validating responses received from remote resources on the server side is the most basic mitigation that can be readily implemented. If a web application expects specific content type on the server, programmatically ensure that the data received satisfies checks imposed on the server before displaying or processing the data for the client.
- **Disable unwanted protocols** – Allow only http and https to make requests to remote servers. Whitelisting these protocols will prevent the web application from making requests over other protocols like file://, gopher://, ftp:// and other URI schemes.
- **Blacklist IP addresses** – Internal IP addresses, localhost specifications and internal hostnames can all be blacklisted to prevent the web application from being abused to fetch data/attack these devices. Implementing this will protect servers from one time attack vectors. For example, even if the first fix (above) is implemented, the data is still being sent to the remote service. If an attack that does not need to see responses is executed (like a buffer overflow exploit) then this fix can actually prevent data from ever reaching the vulnerable device. Response handling is then not required at all as a request was never made.

References

- Wikipedia

http://en.wikipedia.org/wiki/URI_scheme

- CWE

<http://cwe.mitre.org/data/definitions/918.html>

- OWASP

https://www.owasp.org/index.php/Server_Side_Request_Forgery