HTTP Request Smuggling

Brodie, Guy, Ryan,
Ben, Tanbir

https://github.com/Rhyz-jpg/CCSEP-Group1-HTTPRequestSmuggling

HTTP

- Hypertext Transfer Protocol
- An application layer protocol in Internet Suite Model for distributed system.
- Initializing a web communication between browsers and servers
- Stateless protocol - does not store data long term and

  keep track of state information

HTTP Requests

- HTTP requests are created when you try to access resources from a server.
- HTTP requests involve a request line, header fields and a message if required.
- The request line consists of a method (GET,POST), the path for the request and the HTTP version number.
- HTTP headers contain information for the server to respond to the request.

```
GET / HTTP/1.1
Host: google.com
Cookie: 1P_JAR=2021-08-26-10; NID=
222=EPwc729K_CTaK6IXqo3b0qJaz9u5Q4a_oMtbbMzenz6SfJABzNvCjj4cDuojXUxGzPrQwN2AxxoPDYVUe1roEZyG233Cf-g9_SPe3XczUb9YVlncu6D5VGEd
Kcdt0XP2s-VG8EgiDKzJEerm0kF9Azdxq8AYKN4m2iIRLiNix-A
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
X-Client-Data: CKmQywE=
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Header parsing

- To parse the data that comes in the header part of the http request.
- Keep parsing the stream as header fields until it finds the blank line
- Content-length, Transfer Encoding

Front end & Back end servers

- Front end servers are what the user will be able to see and interact with and this server forwards the request to backend server
- Back end servers store most of the important data that users don’t have access too
- These 2 servers need to communicate with each other via HTTP requests to get what the user has requested.

What is HTTP Request Smuggling

- HTTP request smuggling is a way of interfering with the way servers process HTTP requests that are received from users.
- Request smuggling are usually critical vulnerabilities as attackers can easily bypass security and have unauthorised access to confidential data.
- The attack mostly occurs when the front and back end server are out of sync.

HTTP Request Smuggling Prerequisite

- This vulnerability only exists when at least 2 servers are chained together
- One server needs to parsing the http request header differently from other
- A keep alive connection between these 2 servers is also required

```
if "chunked" in header.transferEncoding.value()
    return true


if header.transferEncoding.value() == "chunked"
    return true
```

Types HTTP Request Smuggling

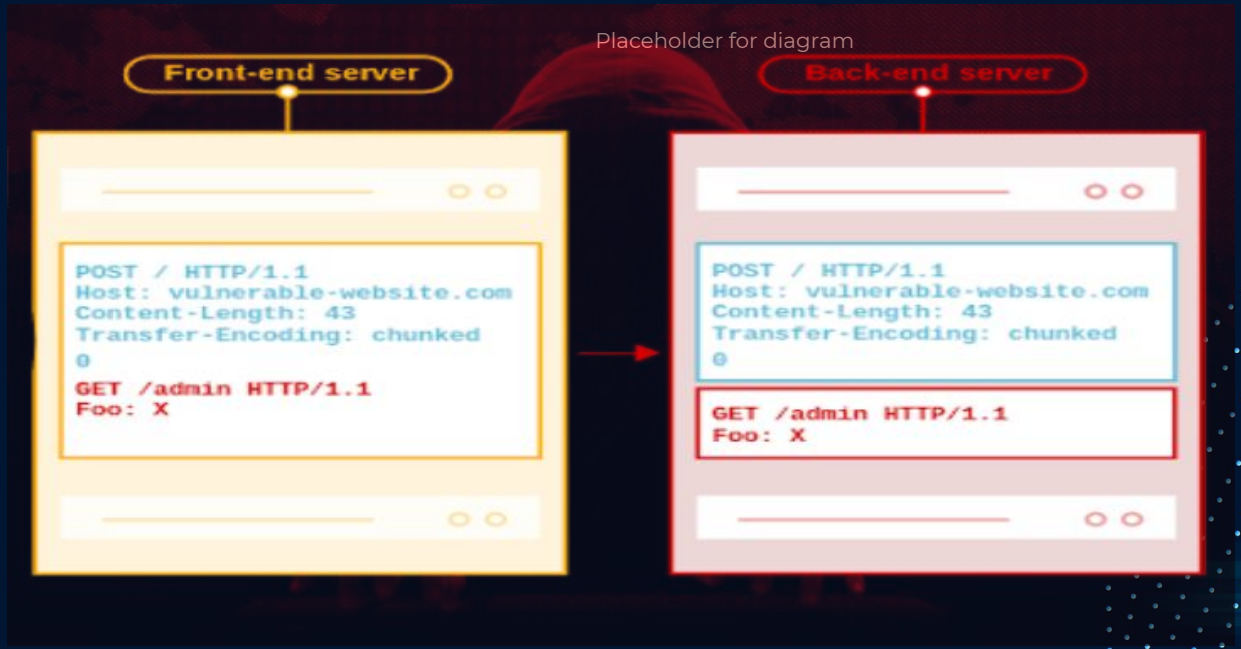
- CL.TE: Content Length .Transfer Encoding
- TE.CL Transfer Encoding. Content Header
- TE.TE

CL.TE Content Length.Transfer Encoding

- CL.Te: Content Length .Transfer Encoding
- Both headers placed in same request
- Front end uses CL and ignores TE
- Back end uses TE and ignores CL

```
POST / HTTP/1.1
Host: 127.0.0.1:5000
Content-Length: 8
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1

```

CL.TE Visual example



TE.CL Transfer Encoding. Content Length

- TE.CL: Transfer Encoding. Content Length
- Both headers placed in same request
- Front end users TE and ignores CL
- Back end uses CL and ignores TE

```
1  POST / HTTP/1.1
2  Host: 127.0.0.1:5000
3  Content-Length: 4
4  Transfter-Encoding: chunked
5
6  12
7  GET /admin HTTP/1.1
8  Content-Length: 10
9
10 xxx
11 0
12
13
```

Simple example of HTTP Request Smuggling

- This example is using the CL.TE vulnerability the front end and back end disagree on where the request ends. The back end believes after the 0 the next segment is a new HTTP request not the same one. Thus it gets executed with the next HTTP request that comes in.

```
POST / HTTP/1.1
Host: 127.0.0.1:5000
Content-Length: 8
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1

```

Impacts of HTTP Request Smuggling

- HTTP request smuggling is a critical vulnerability, attackers can easily have access to an admins cookies or web pages they should not have access too.
- An attacker can cause harm to users of the website as well, although if the website has high traffic the attacker can not attack a specific person.
- Session Hijacking, Privilege Escalation and cache poisoning can be the impact of this vulnerability as well.

Real world Incident

- Paypal bug bounty, using HTTP request smuggling James Kettle was able to put in his javascript with cache poisoning and was able to view people's passwords from paypal in plain text from his site.
- Paypal patched the bug quickly, but a week later he used a line-wrapped header and was again able to de-synchronise the servers and produce the same attack.
- Researcher used CL.TE on slack to take victims cookies and hijack their sessions.

HTTP Request Smuggling Detection

The Timing Technique.

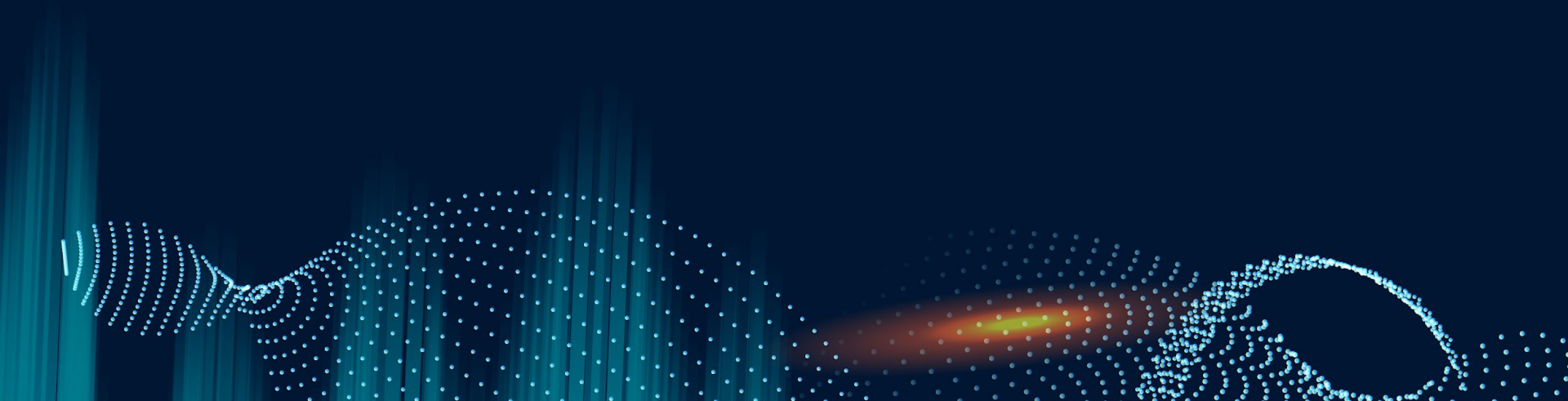
```
POST / HTTP/1.1
Host: 127.0.0.1:8000
Transfer-Encoding: chunked
Content-Length: 6

0

X
```

HTTP Request Smuggling Prevention

- Ensure all servers in the chain parse headers the same way, to ensure they all agree on the length of requests.
- Have the front end server not allow ambiguous requests through.
- Disable keep alive connection.
- Ensure front end servers communicate with backend servers in http 2.0.

Pros	Cons
Efficient.	Complicated.
Robust.	Hard to implement.
Simple.	Reduces performance.
Simple.	Not completely safe.

Demonstration



Conclusion

Crux of HTTP request smuggling vulnerability:

- Header, heasers, headers
- Reuse of TCP connection to backend

Paypal was vulnerable to a HTTP request smuggling attack

HTTP request header parsing is a critical security function

Reference List

https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn

HTTP Request Smuggling (whitehatsec.com)

HTTP Request Smuggling / HTTP Desync Attack - HackTricks

HTTP Request Smuggling: Complete Guide to Attack Types and Prevention (neuralegion.com)

(1) New Messages! (extrahop.com)

https://cobalt.io/blog/a-pentesters-guide-to-http-request-smuggling

https://blog.deteact.com/gunicorn-http-request-smuggling/

Reference List

Detection

- https://portswigger.net/web-security/request-smuggling/finding

Prevention.

https://portswigger.net/web-security/request-smuggling
https://cobalt.io/blog/a-pentesters-guide-to-http-request-smuggling
https://www.neuralegion.com/blog/http-request-smuggling-hrs/#hrs-prevention

Thank you!