Network Fingerprinting for Securing User Accounts

Opportunities and Challenges

~ whoami

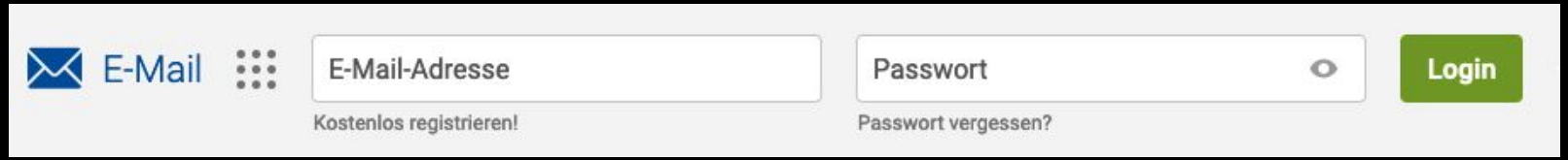
- Stephan Pinto Spindler
- Leading the Security Engineering team at 1&1 Mail & Media (web.de, GMX, mail.com)

https://github.com/s-spindler/

https://www.linkedin.com/in/stephan-pinto-spindler-61a511b2/

Protecting Users

How can you protect your users?



- Credential stuffing
- Password spraying
- Brute forcing

How can you protect your users?

Rate limiting and (temporary) locks/bans!

… but on what?

On accounts

> Only works for targeted attacks

On source IPs

> VPNs, NAT, distributed attacks

Do you even notice the attack?

Metadata

Looking at metadata

method

```
GET / HTTP/1.1
Host: web.de
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)...
Accept: text/html,application/xhtml+xml,...
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,de-DE;q=0.8
Referer: https://www.google.com/
Connection: keep-alive
Cookie: sessionid=123456789; userid=987654321...
```

version

headers

cookies

```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 649
    Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 645
      Version: TLS 1.2 (0x0303)
      Random: 56c5e19d41daf2c2b5f3a066fe5465580d2cbf8013f22abc49aa94a831c86722
      Session ID Length: 32
      Session ID: b07556f3340e00c2776a0a706ec2277f8cb877f70afc26e22bd56911420a7193
      Cipher Suites Length: 34
      Cipher Suites (17 suites)
      Compression Methods Length: 1
      Compression Methods (1 method)
      Extensions Length: 538
      Extension: server_name (len=11) name=web.de
      Extension: extended_master_secret (len=0)
      Extension: renegotiation_info (len=1)
      Extension: supported_groups (len=14)
      Extension: ec_point_formats (len=2)
      Extension: session_ticket (len=0)
      Extension: application_layer_protocol_negotiation (len=14)
      Extension: status_request (len=5)
      Extension: delegated_credentials (len=10)
      Extension: key_share (len=107) x25519, secp256r1
      Extension: supported_versions (len=5) TLS 1.3, TLS 1.2
      Extension: signature_algorithms (len=24)
      Extension: psk_key_exchange_modes (len=2)
      Extension: record_size_limit (len=2)
      Extension: encrypted_client_hello (len=281)
```

There’s more…

- TCP/QUIC metadata: duration between steps in the handshake
- HTTP2: settings for, e.g., frame sizes

Focus today: HTTP and TLS

Fingerprints

Common fingerprint "standards" - JA3

- **JA3:** invented at Salesforce but no longer maintained
- Still supported in various tools
- Uses these fields from Client Hello:

SSLVersion, Cipher, SSLExtension, EllipticCurve, EllipticCurvePointFormat

```
[JA3 Fullstring: 771,4865-4867-4866-49195-49199-52393-52392-49196-49200-491
[JA3: b5001237acdf006056b409cc433726b0]
```

screenshot straight out of Wireshark

Common fingerprint "standards" - JA4

JA4+ suite: by FoxIO, builds on JA3, adds some more

- Methods for TLS, HTTP, TCP, SSH
- Both client and server

Licensing note:

- Only TLS client fingerprinting (JA4) is BSD licensed
- The rest: FoxIO license - still good to go for internal business use

FOXIO

BSD 3-Clause License

JA4: TLS Client Fingerprint

- Protocol, TCP = “t” QUIC = “q”
- TLS version, 1.2 = “12”, 1.3 = “13”
- SNI, SNI = “d” (to domain), no SNI = “i” (to IP)
- Number of Cipher Suites
- Number of Extensions
- First ALPN value (00 if no ALPN)

JA4=t13d1516h2_acb858a92679_e5627efa2ab1

JA4_a JA4_b JA4_c

- Truncated SHA256 hash of the Cipher Suites, sorted
- Truncated SHA256 hash of the Extensions, sorted
  + Signature Algorithms, in the order they appear

Source: https://raw.githubusercontent.com/FoxIO-LLC/ja4/refs/heads/main/technical_details/JA4.png

FoxIO

Patent Pending
FoxIO License 1.1

JA4H: HTTP Client Fingerprint

- HTTP Method, GET = “ge”, PUT = “pu”, POST = “po”, etc
- HTTP Version, 2.0 = “20”, 1.1 = “11”
- Cookie, if there’s a Cookie “c”, if no Cookie “n”
- Referer, if there’s a Referer “r” if no Referer “n”
- Number of HTTP Headers (ignoring Cookie and Referer)
- First 4 characters of primary Accept-Language (0000 if no Accept-Language)

JA4H=ge20cr13enus_974ebe531c03_b66fa821d02c_e97928733c74

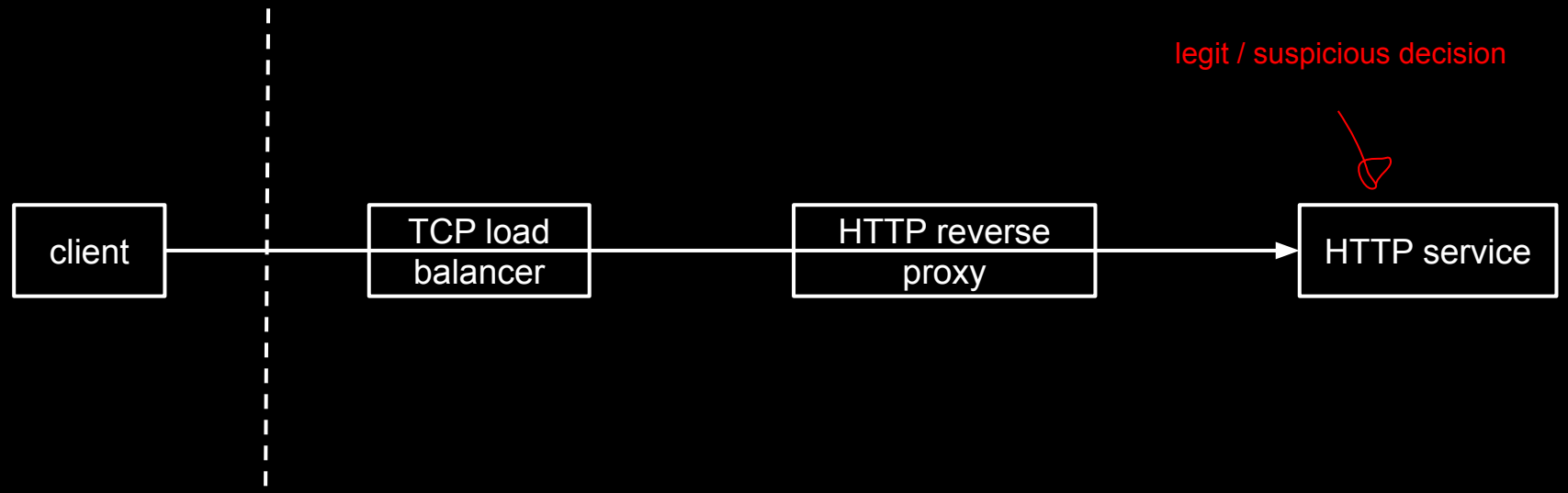
JA4H_a JA4H_b JA4H_c JA4H_d

- Truncated SHA256 hash of Headers, in the order they appear
- Truncated SHA256 hash of Cookie Fields, sorted
- Truncated SHA256 hash of Cookie Fields + Values, sorted

Source: https://raw.githubusercontent.com/FoxIO-LLC/ja4/refs/heads/main/technical_details/JA4H.png

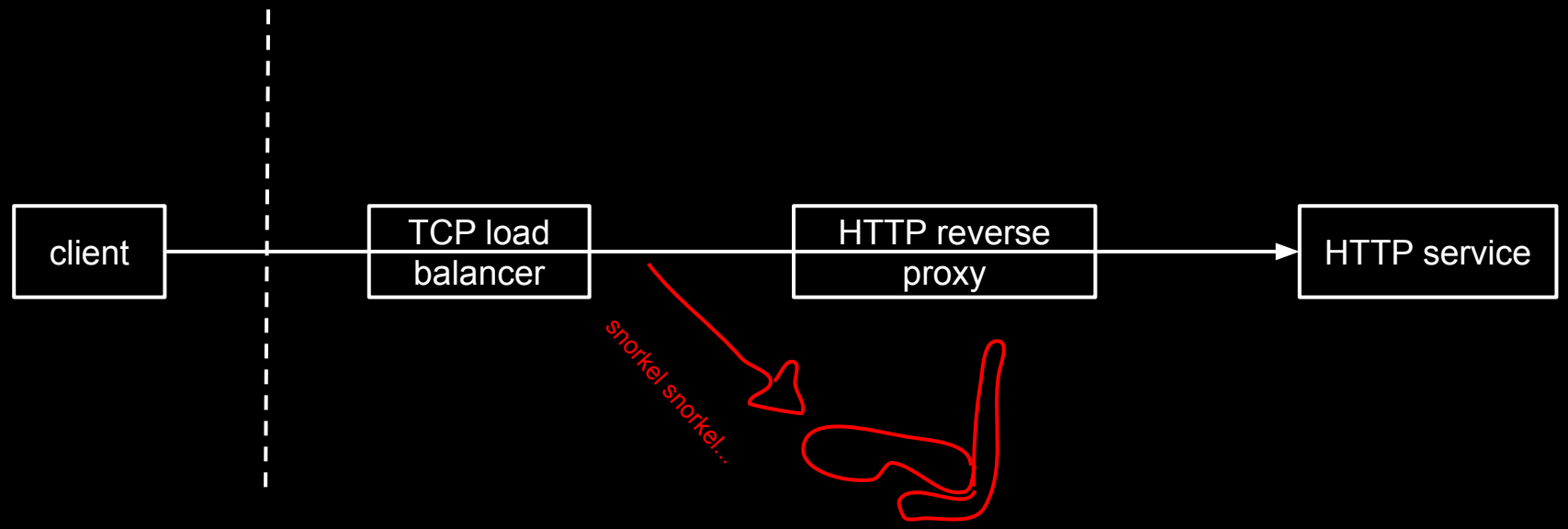
Technical Challenges

Sample setup

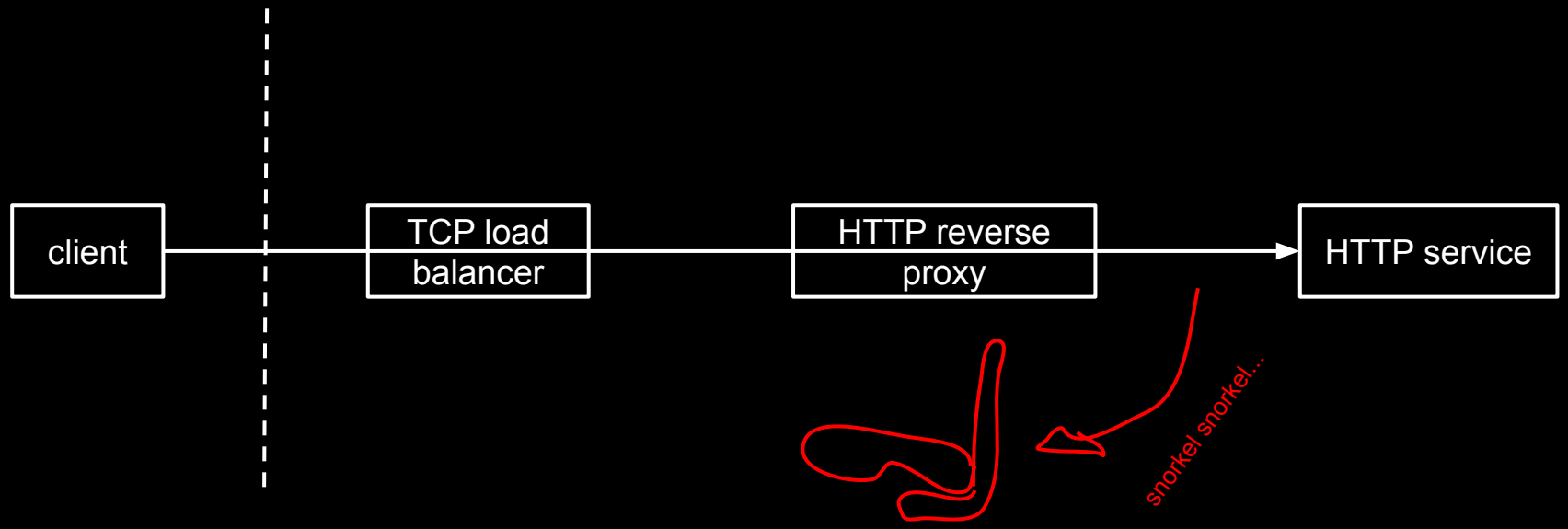


Out-of-Band Fingerprint Creation

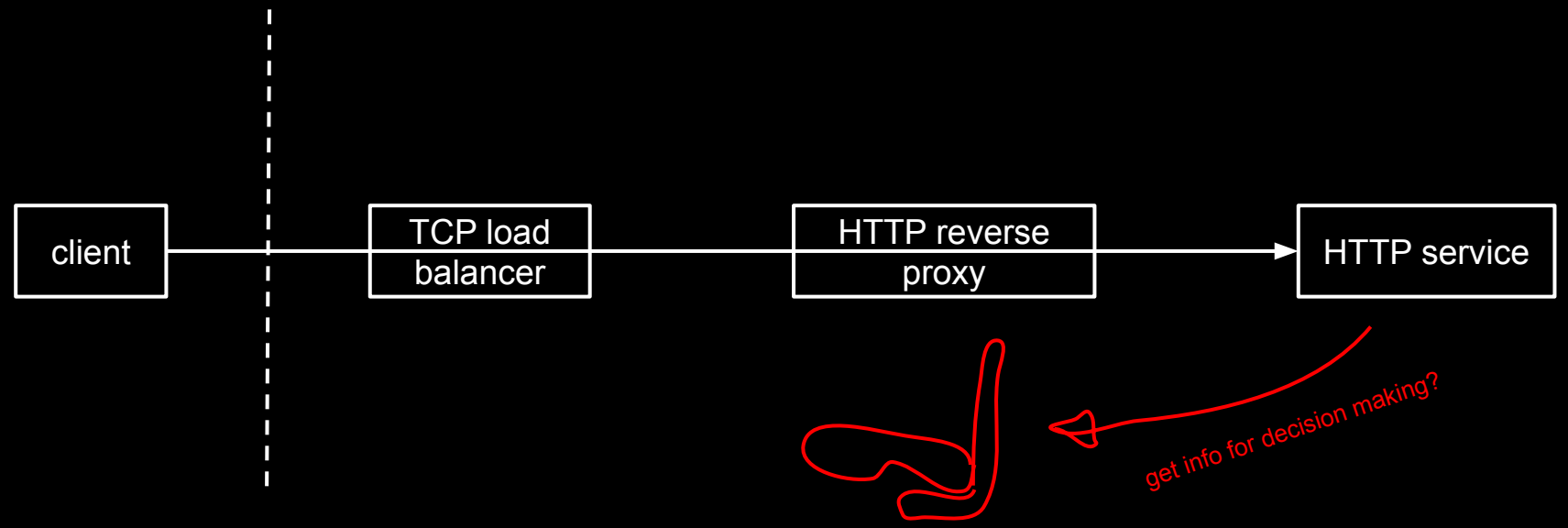
Challenges for out-of-band fingerprint creation



Challenges for out-of-band fingerprint creation

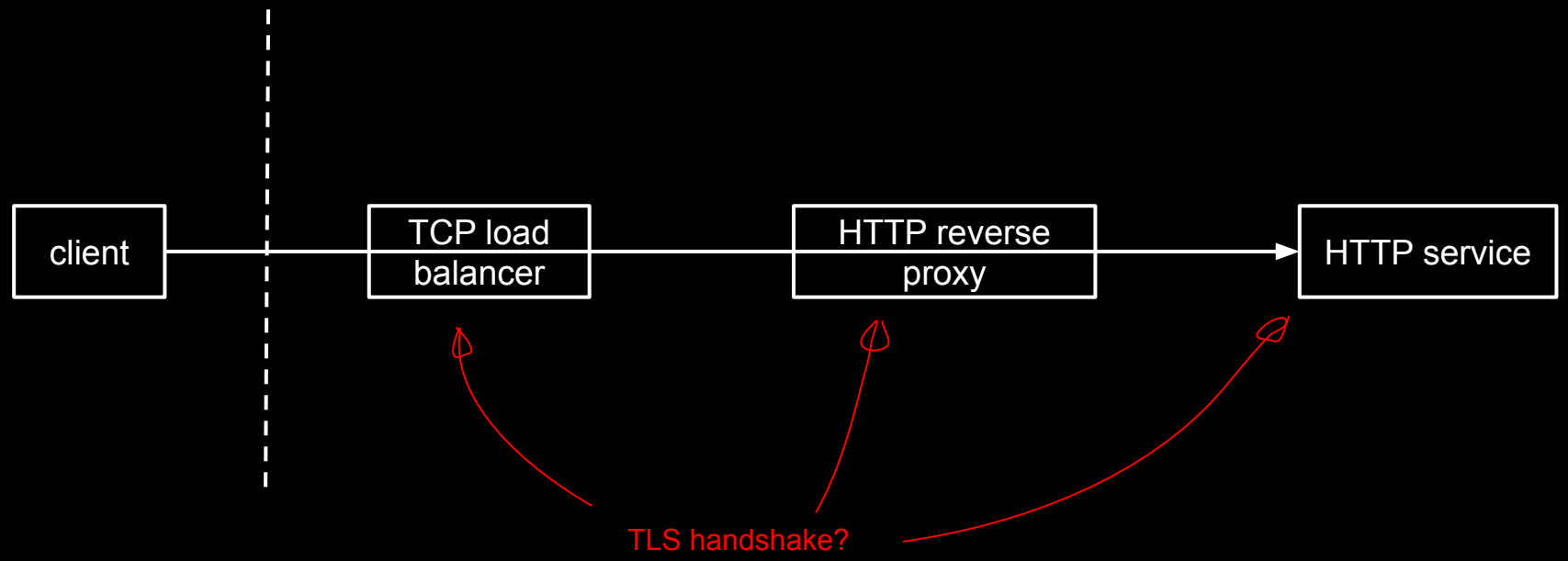


Challenges for out-of-band fingerprint creation

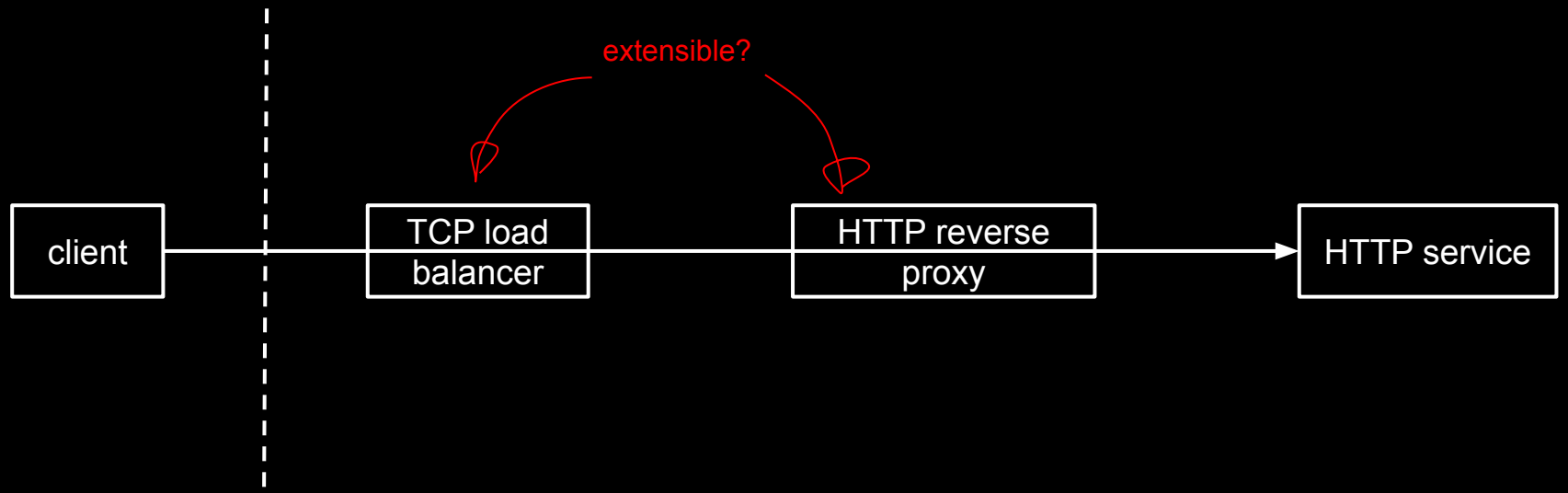


In-Stream Fingerprint Creation

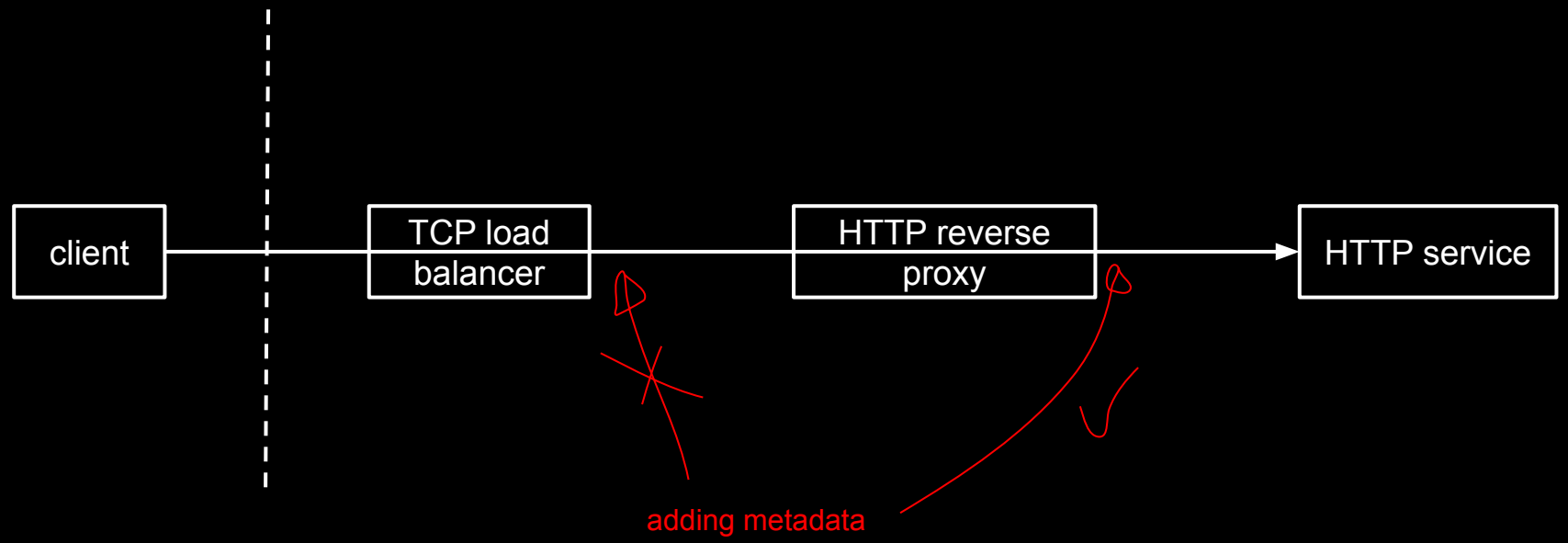
Challenges for in-stream fingerprint creation



Challenges for in-stream fingerprint creation

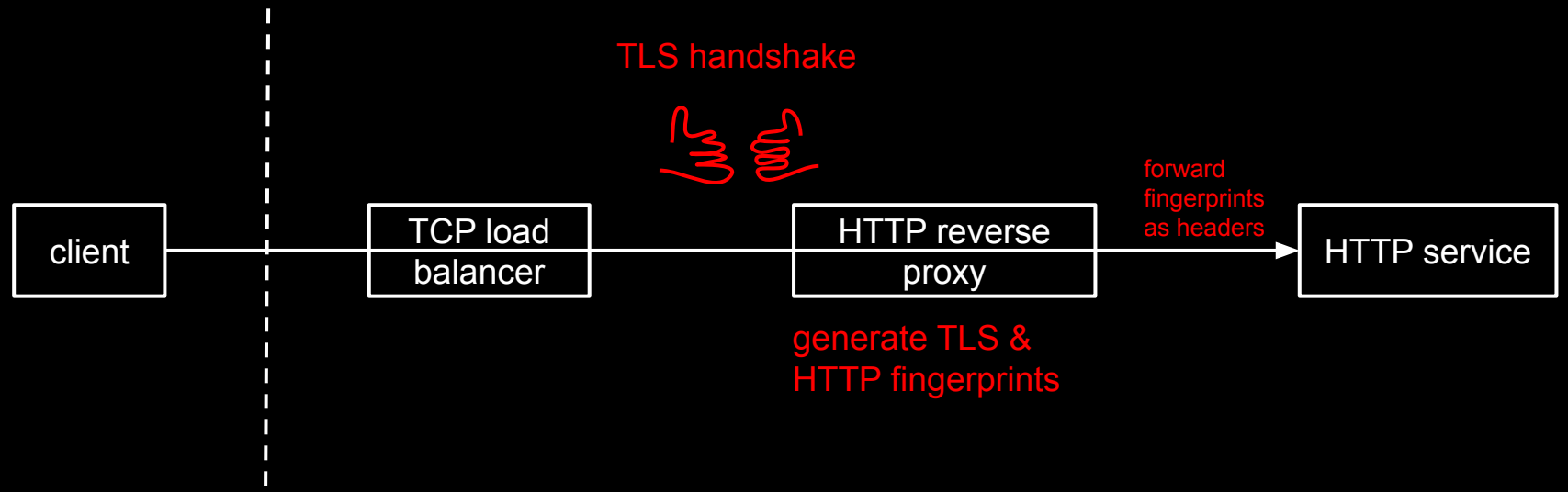


Challenges for in-stream fingerprint creation



Possible setup for in-stream creation

TLS offloading in reverse proxy (can add TLS on upstream connection again)



Implementing HTTP fingerprinting - traefik plugin example

```go
func (p *Plugin) ServeHTTP(rw http.ResponseWriter, req *http.Request) {

    method := req.Method
    version := fmt.Sprintf("%d%d", req.ProtoMajor, req.ProtoMinor)
    cookies := req.Cookies()
    referer := req.Referer()
    headers := req.Header
    lang := req.Header.Get("Accept-Language")

    req.Header.Add("X-FP-HTTP", createFingerprint(method, version, cookies, referer, headers, lang))

    p.next.ServeHTTP(rw, req)
}
```

Manipulating HTTP metadata - Go example

```go
client := &http.Client{
    Transport: &http.Transport{
        TLSNextProto: map[string]func(string, *tls.Conn) http.RoundTripper{},
    },
}

req, err := http.NewRequest("GET", "https://web.de", nil)
if err != nil {
    panic(err)
}
req.AddCookie(&http.Cookie{Name: "mycookie", Value: "blabla"})
req.Header.Set("Custom-Header", "some value")
req.Header.Set("User-Agent", "Chrome/123.4.5.67")

resp, err := client.Do(req)
```

forces HTTP 1.1

manipulate headers and cookies

Manipulating HTTP metadata - curl example

```
curl https://web.de \
   --http1.1 \
   --user-agent "Chrome/123.4.5.67" \
   --cookie "mycookie=blabla" \
   --header "Custom-Header: some value"
```

Implementing TLS fingerprinting - traefik plugin example

```
func (p *Plugin) ServeHTTP(rw http.ResponseWriter, req *http.Request) {

   negotiated_version := req.TLS.Version
   sni := req.TLS.ServerName
   alpn := req.TLS.NegotiatedProtocol
   negotiated_cipher := req.TLS.CipherSuite

   req.Header.Add("X-FP-TLS", createFingerprint(negotiated_version, sni, alpn, negotiated_cipher))

   p.next.ServeHTTP(rw, req)
}
```

> Missing TLS handshake information (supported versions, cipher suites, extensions)

Manipulating TLS metadata - curl example

```
curl https://httpbin.ams.fe-intg-iz2-bap.poinfra.server.lan/anything \
   --alpn \
   --tlsv1.2 \
   --tls-max "1.3" \
   --ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384 \
   --tls13-ciphers TLS_AES_128_GCM_SHA256
```

sets minimum TLS version to 1.2

specifies ciphers for TLS versions <= 1.2

> No easy way of manipulating extensions

Technical Challenges - Bonus Round

FoxIO

BSD 3-Clause License

JA4: TLS Client Fingerprint

- Protocol, TCP = “t” QUIC = “q”
- TLS version, 1.2 = “12”, 1.3 = “13”
- SNI, SNI = “d” (to domain), no SNI = “i” (to IP)
- Number of Cipher Suites
- Number of Extensions
- First ALPN value (00 if no ALPN)

always 2 digits
max. 99

JA4=t13d1516h2_acb858a92679_e5627efa2ab1

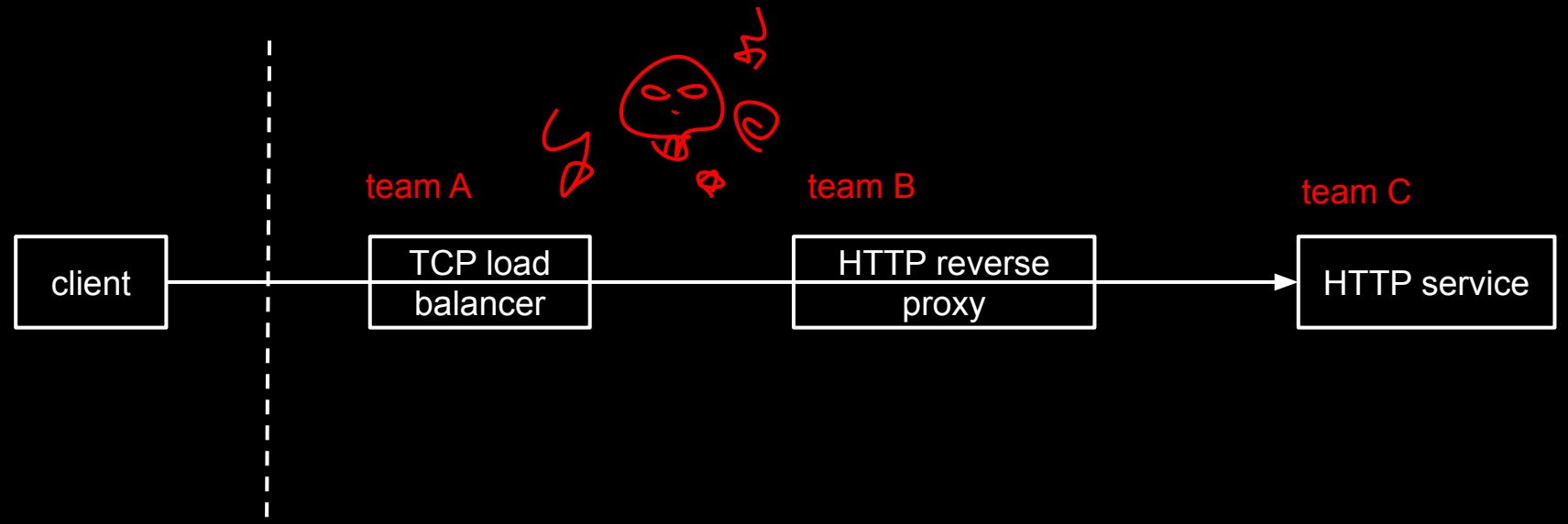
JA4_a JA4_b JA4_c

- Truncated SHA256 hash of the Cipher Suites, sorted
- Truncated SHA256 hash of the Extensions, sorted
  + Signature Algorithms, in the order they appear

Source: https://raw.githubusercontent.com/FoxIO-LLC/ja4/refs/heads/main/technical_details/JA4.png

Organizational Challenges

Organizational challenges



Regulatory Challenges

Privacy

- Fingerprints can count as personally identifiable information
- Storage might be off-limits
- Talk to your data privacy officer

Conclusion

Still worth the effort?

- Probably yes
- Avoid being an easy target
- Drive up cost for attackers

Thank you!