Detecting TLS Interception in the Wild

Okan Saracbasi

Supervisor: Prof. Dr. Kevin Borgolte

Advisor: Dr. Talha Paracha

Agenda

- Motivation
- Transport Layer Security
- Methodology
- Server Architecture
- Detection Method
- Design Choices
- Results
- Reproducibility
- Future Work

Motivation

- End-to-end security of data sent between applications over the Internet
- Ensuring data confidentiality, integrity, and authentication
- Past showed many vulnerabilities:
  - Heartbleed
  - FREAK
  - Bleichenbacher
  - Logjam
  - ....

Motivation

- Legitimate interceptions: government, enterprise
- Estimated 5–10% of connections are intercepted (in general)*

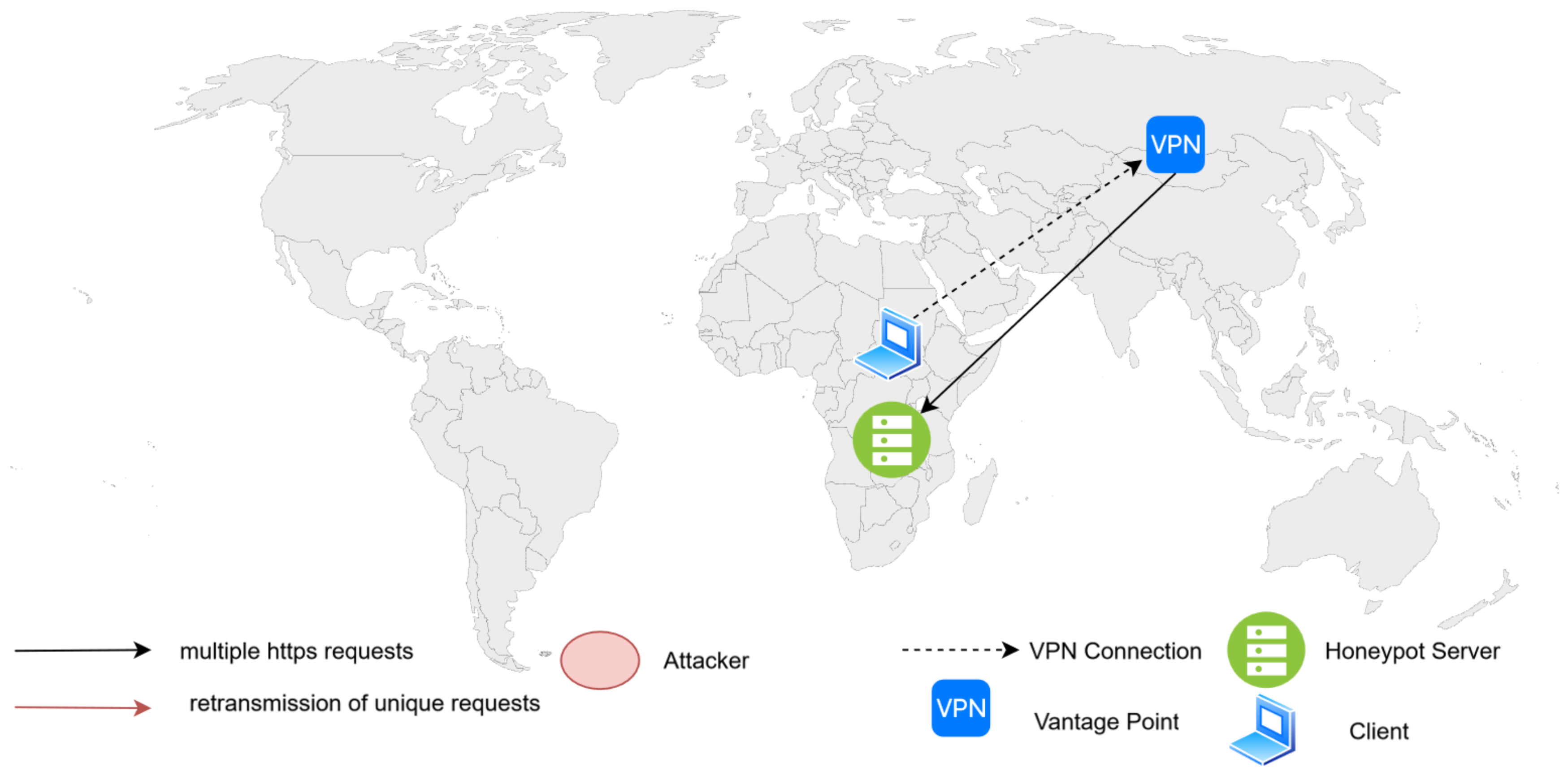
- Goal: detect (harmful) TLS interceptions in the wild

*Durumeric et al., The Security Impact of HTTPS Interception (2017)

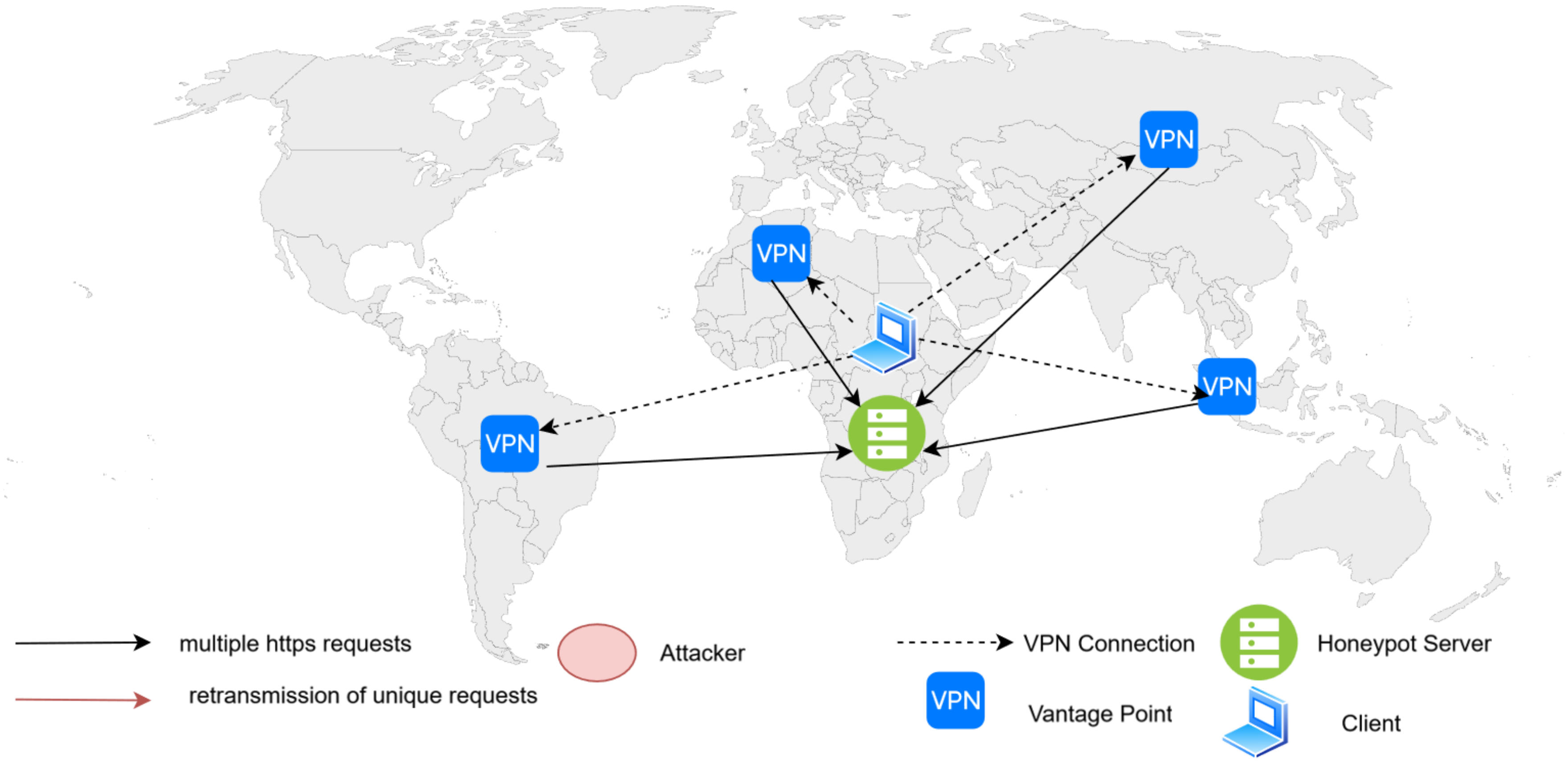
Transport Layer Security

- Handshake protocol
  - Authenticate
  - Encryption techniques
  - Key exchange
- Record protocol
  - confidentiality, integrity, and authentication of application data
  - Fragmentation
  - Encryption
  - Message Authentication

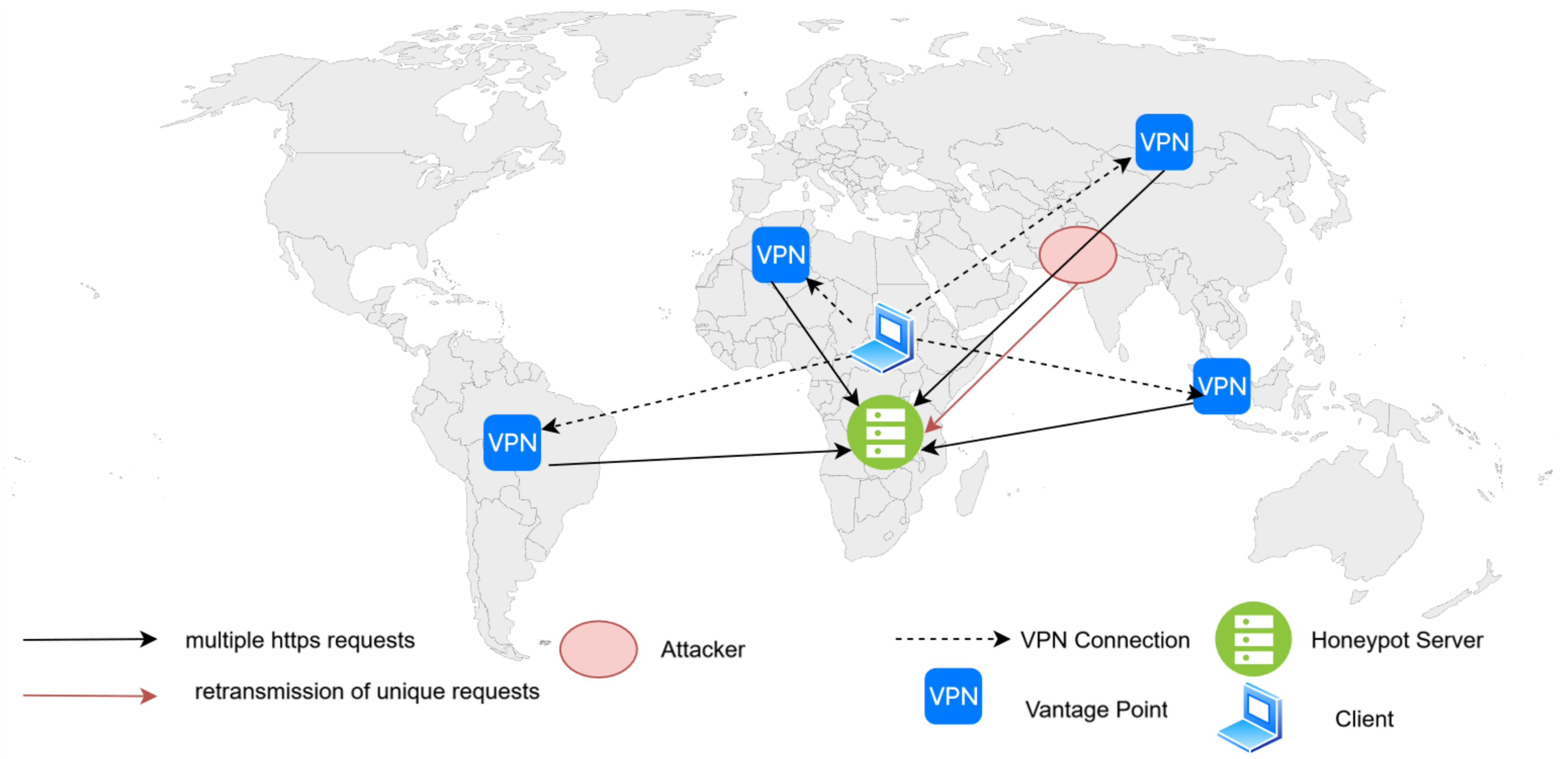
Methodology

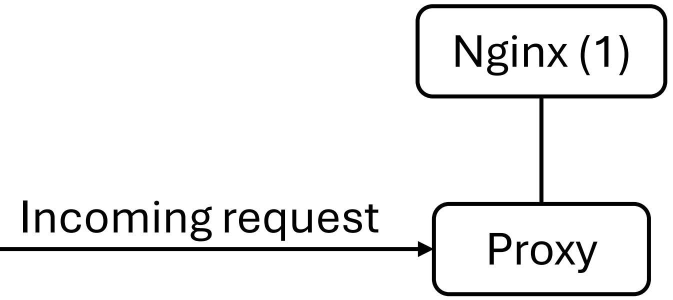


Methodology

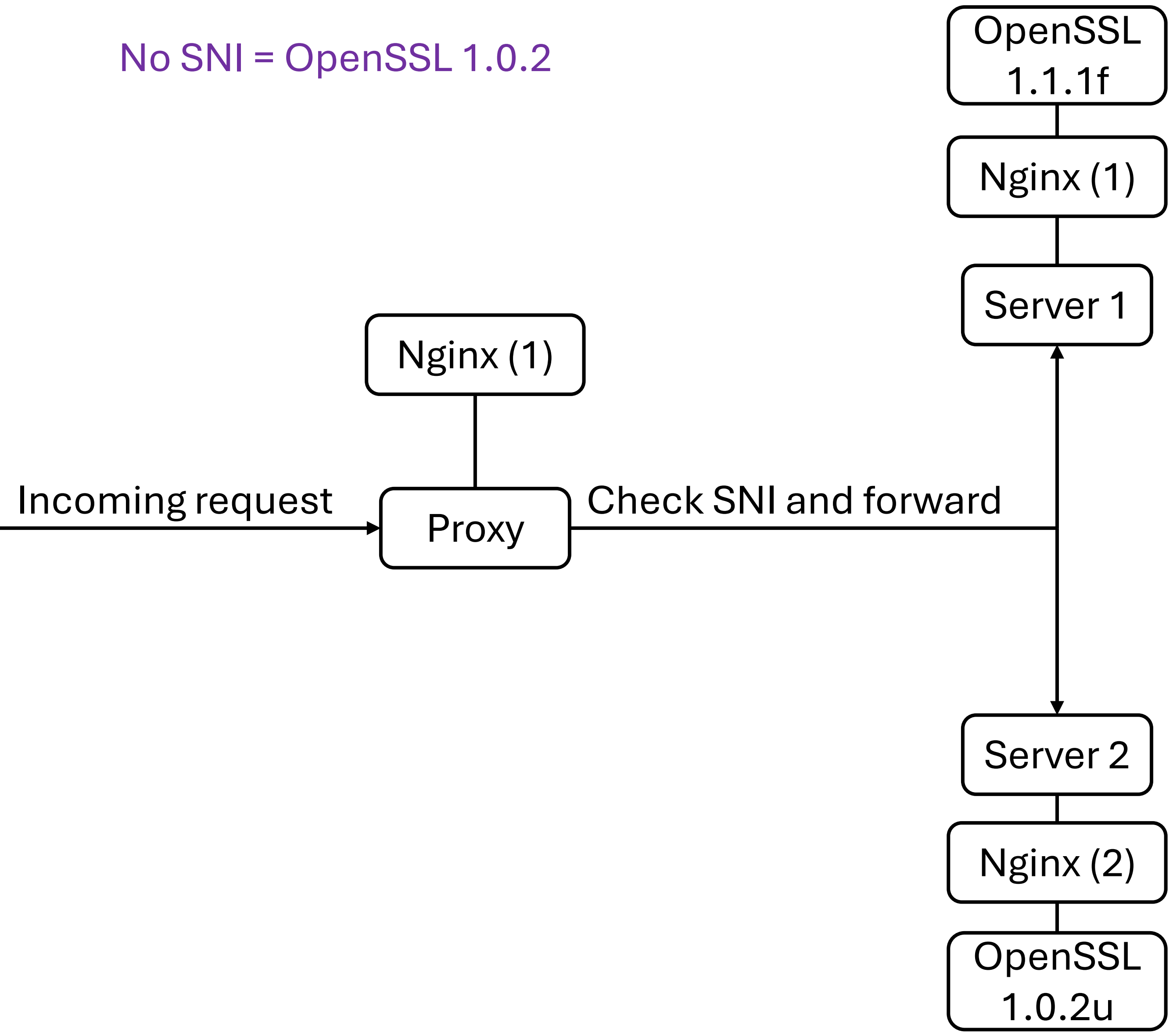


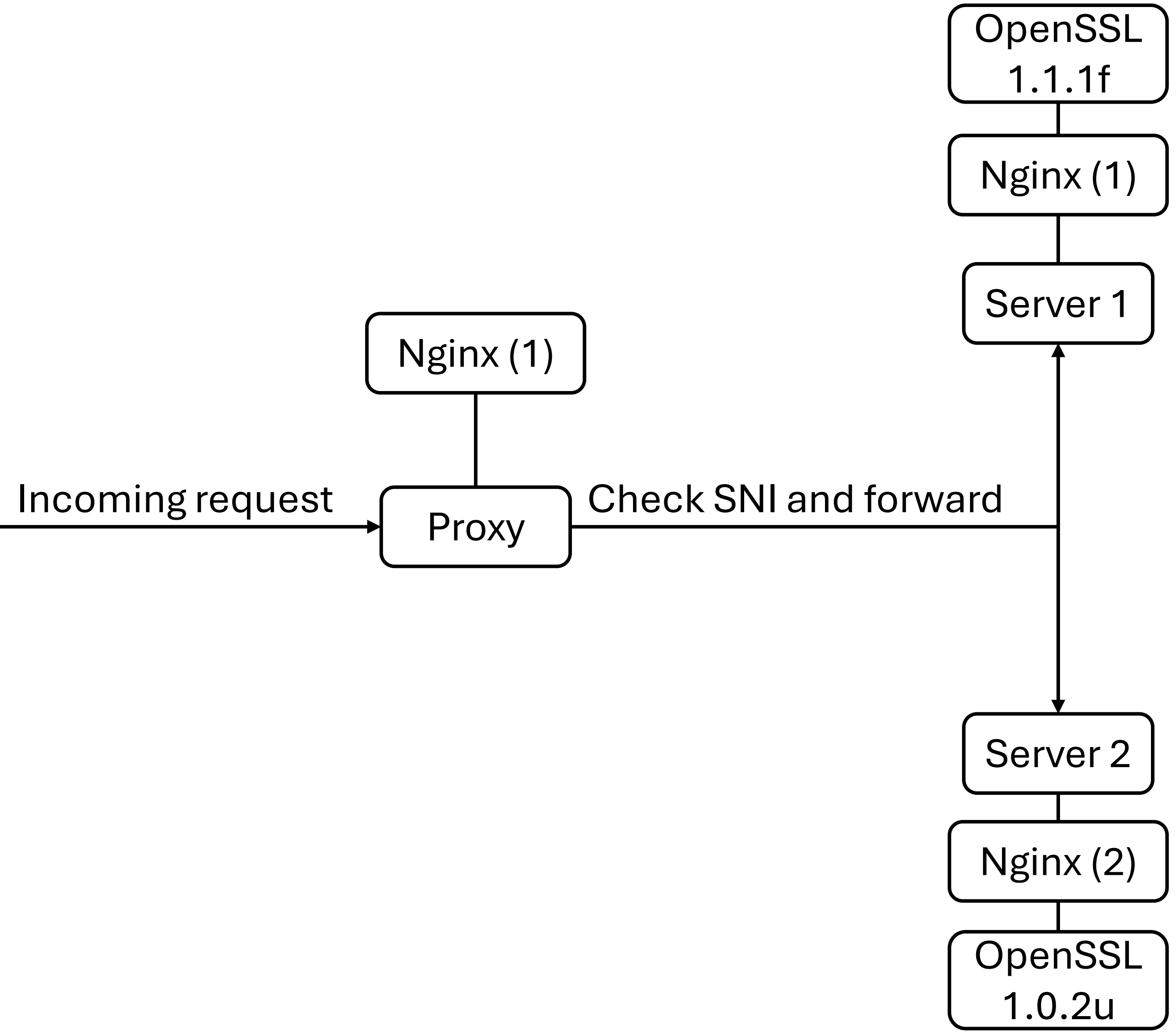
Methodology



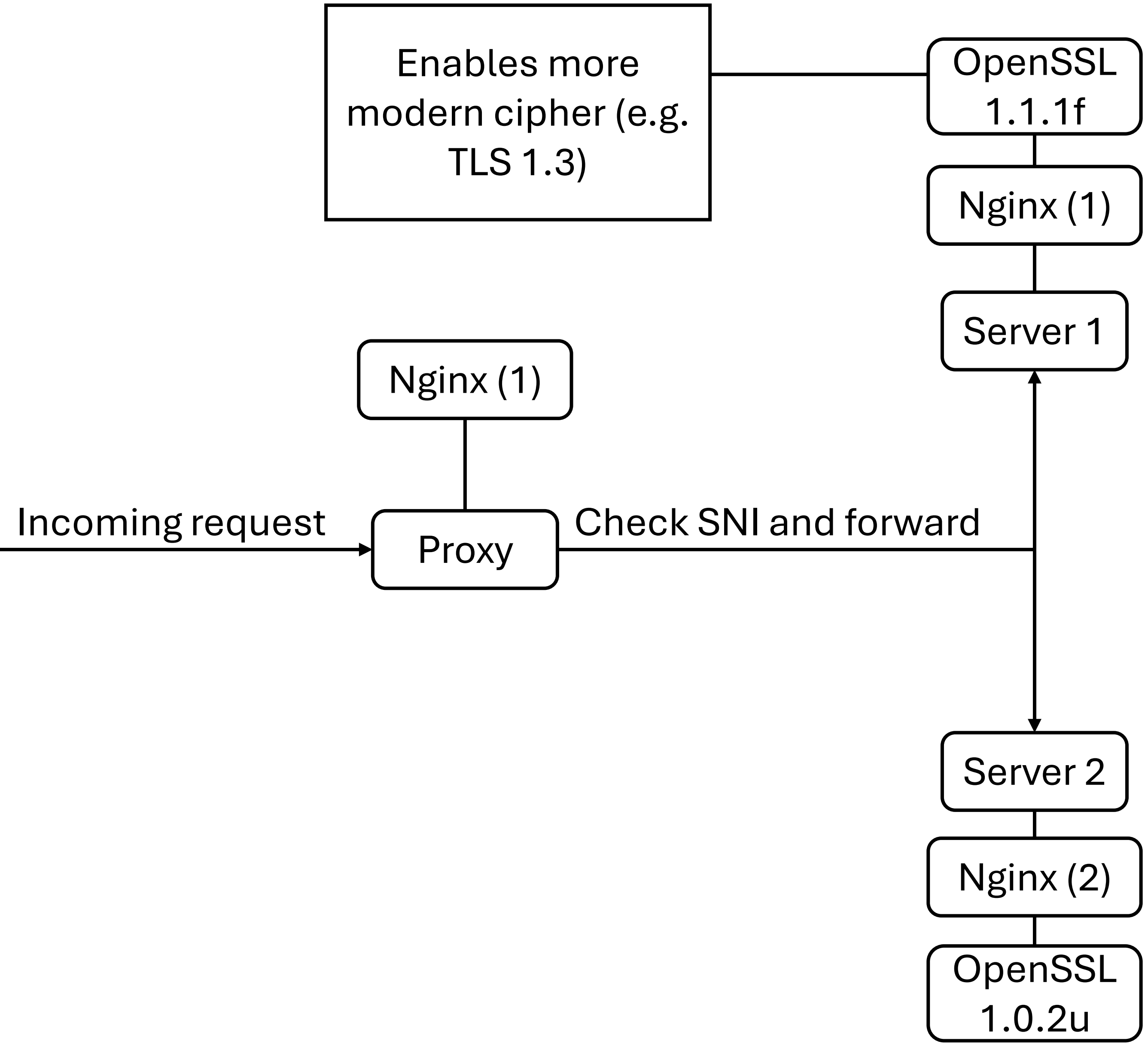

Nginx (1)
Incoming request
Proxy

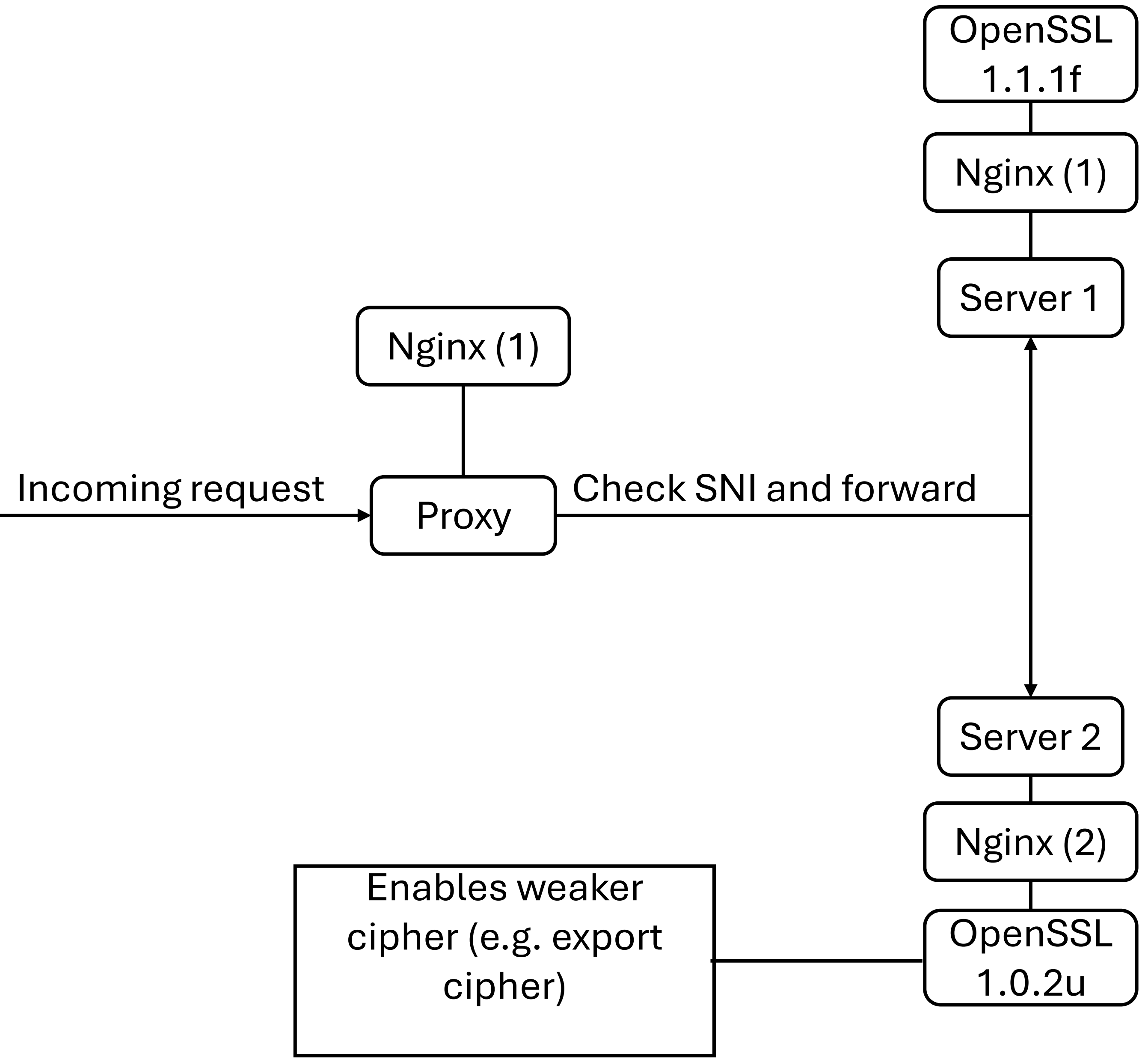
No SNI = OpenSSL 1.0.2

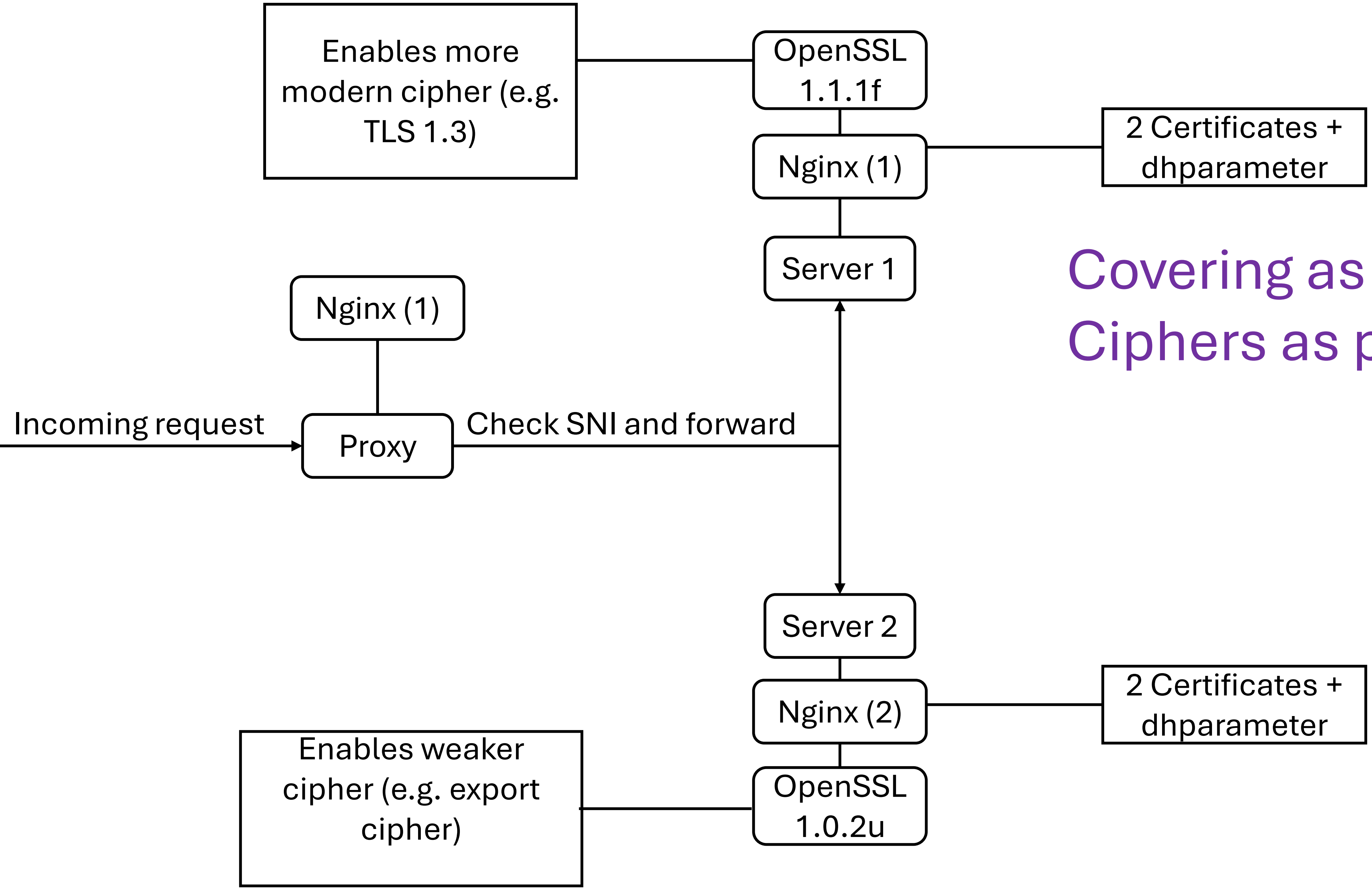




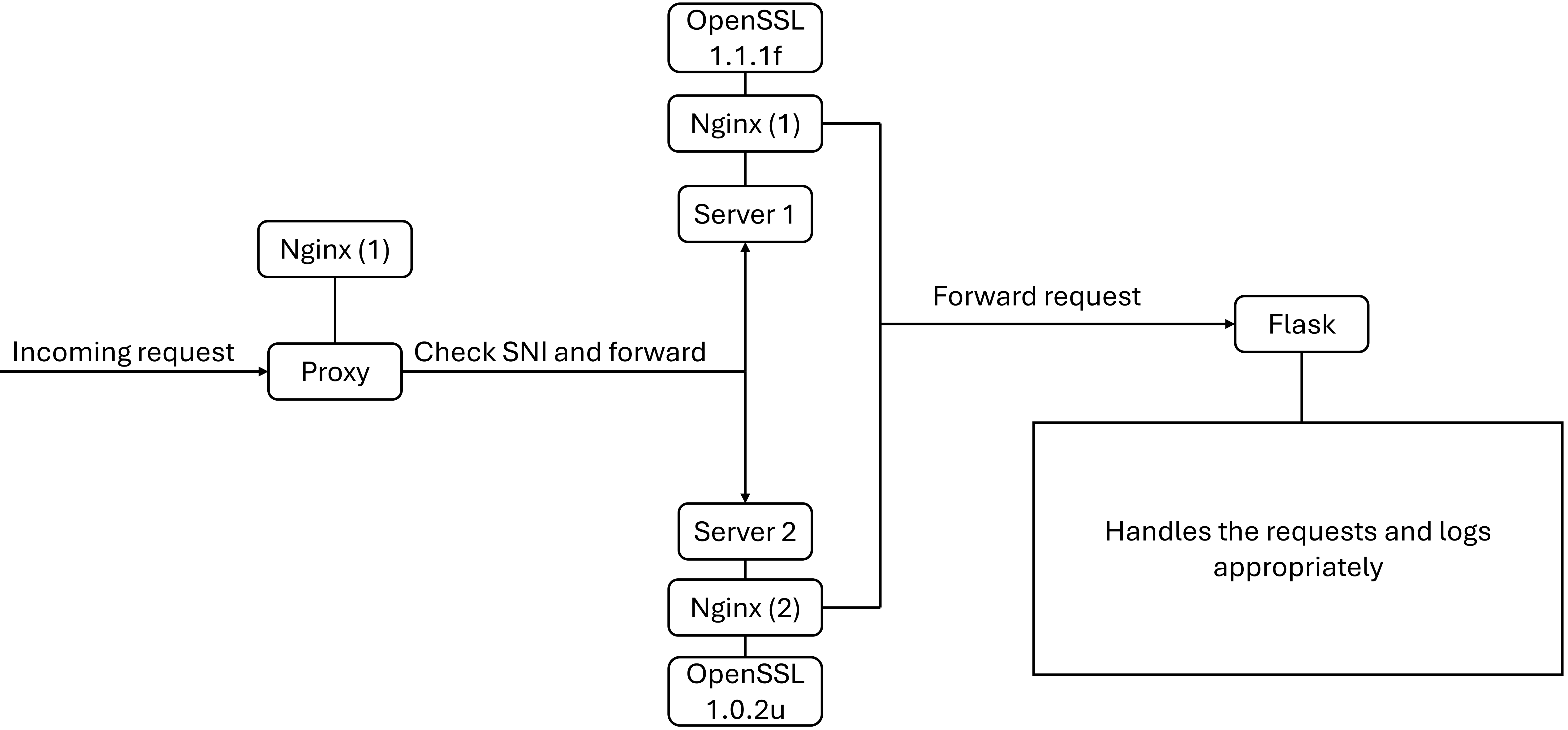
Using two Nginx instances / 2 vHost
=> to utilize two OpenSSL versions


Enables more modern cipher (e.g. TLS 1.3)
OpenSSL 1.1.1f
Nginx (1)
Server 1
Nginx (1)
Incoming request
Proxy
Check SNI and forward
Server 2
Nginx (2)
OpenSSL 1.0.2u

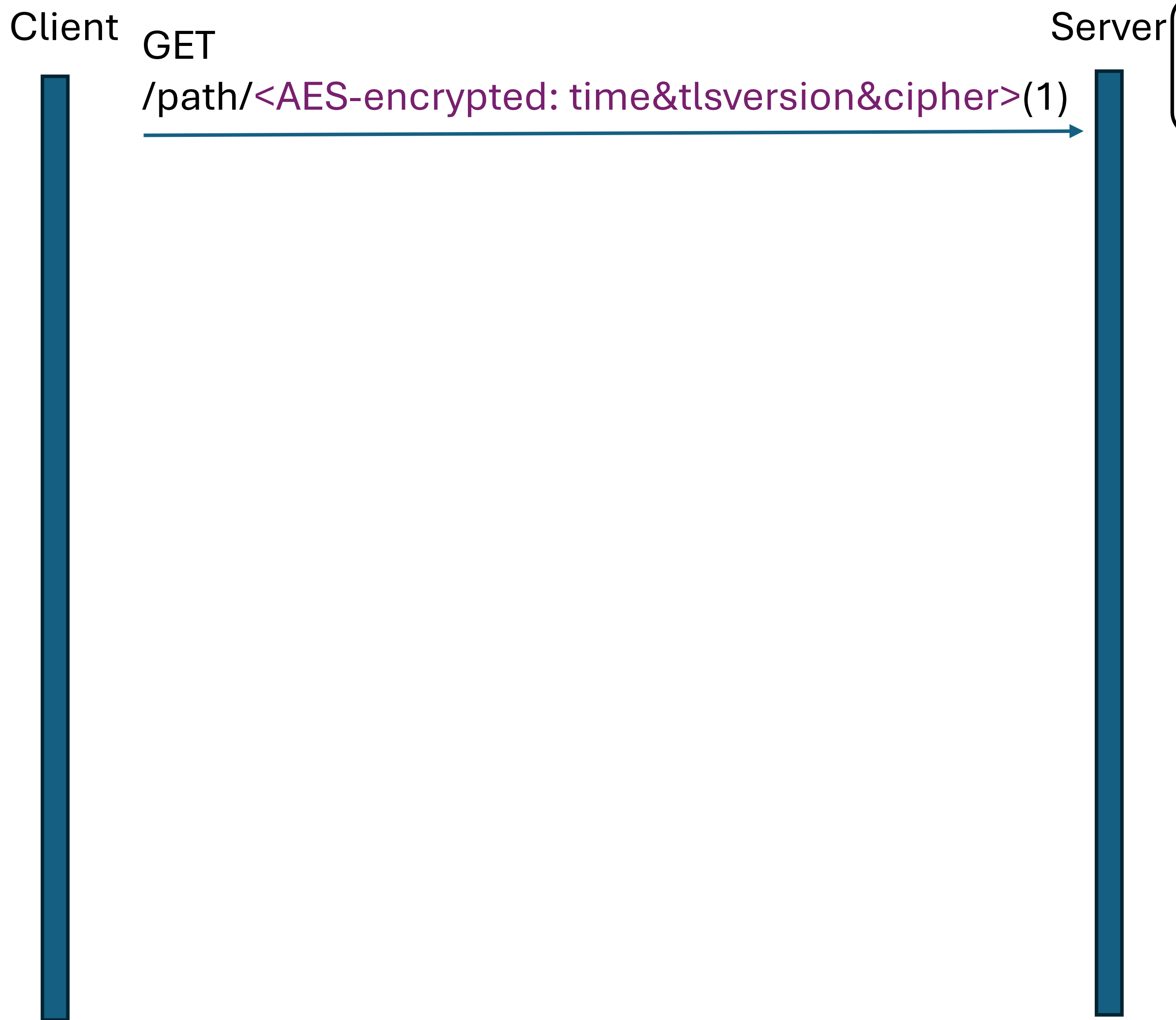

OpenSSL 1.1.1f
Nginx (1)
Server 1
Nginx (1)
Incoming request
Proxy
Check SNI and forward
Server 2
Nginx (2)
Enables weaker cipher (e.g. export cipher)
OpenSSL 1.0.2u


Enables more modern cipher (e.g. TLS 1.3)
OpenSSL 1.1.1f
2 Certificates + dhparameter
Nginx (1)
Server 1
Nginx (1)
Incoming request
Proxy
Check SNI and forward
Server 2
Nginx (2)
2 Certificates + dhparameter
Enables weaker cipher (e.g. export cipher)
OpenSSL 1.0.2u

Covering as many Ciphers as possible


OpenSSL 1.1.1f
Nginx (1)
Server 1
Nginx (1)
Incoming request
Proxy
Check SNI and forward
Forward request
Flask
Handles the requests and logs appropriately
Server 2
Nginx (2)
OpenSSL 1.0.2u

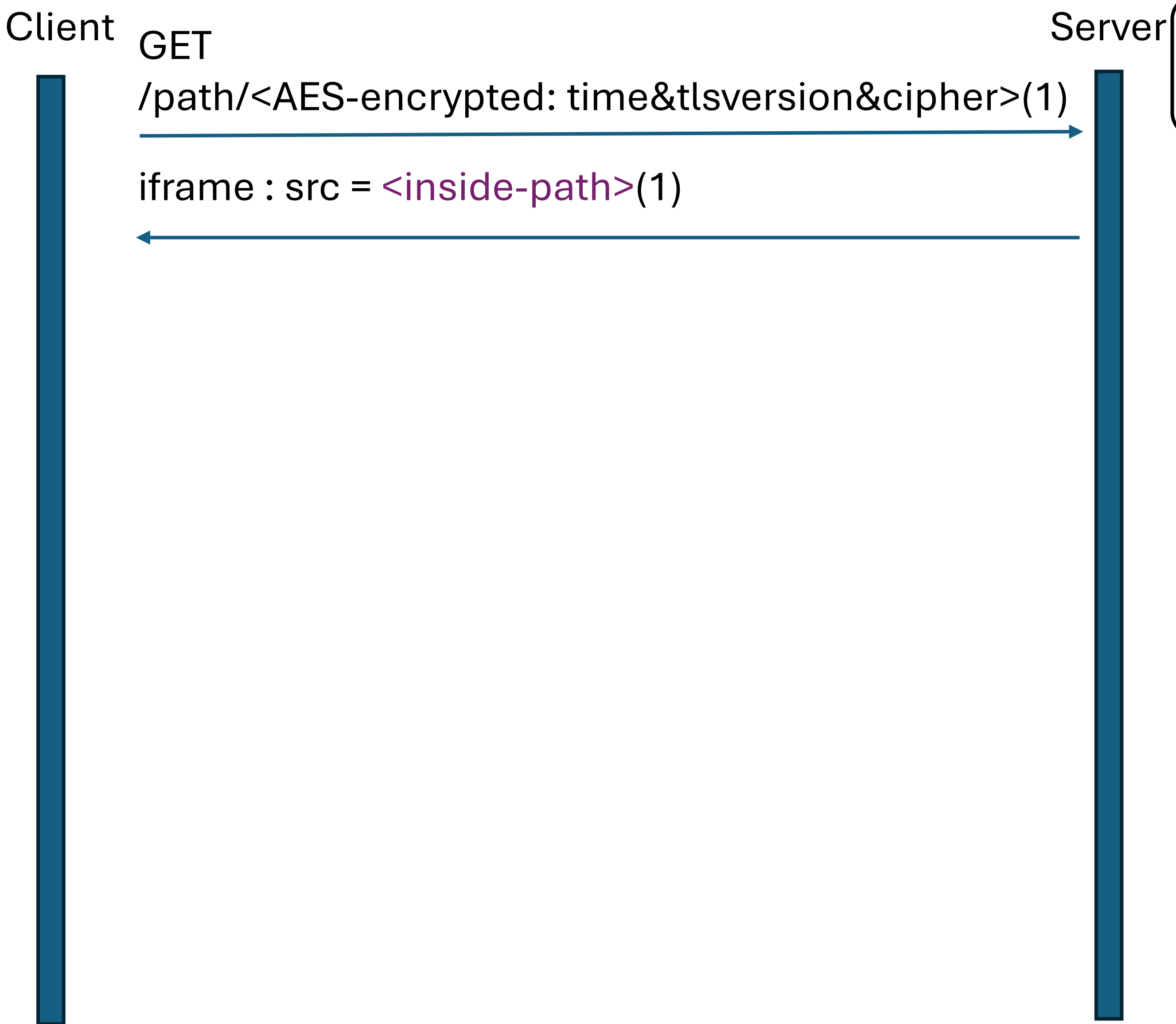
Methodology



GET
/path/<AES-encrypted: time&tlsversion&cipher>(1)

/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>

Methodology

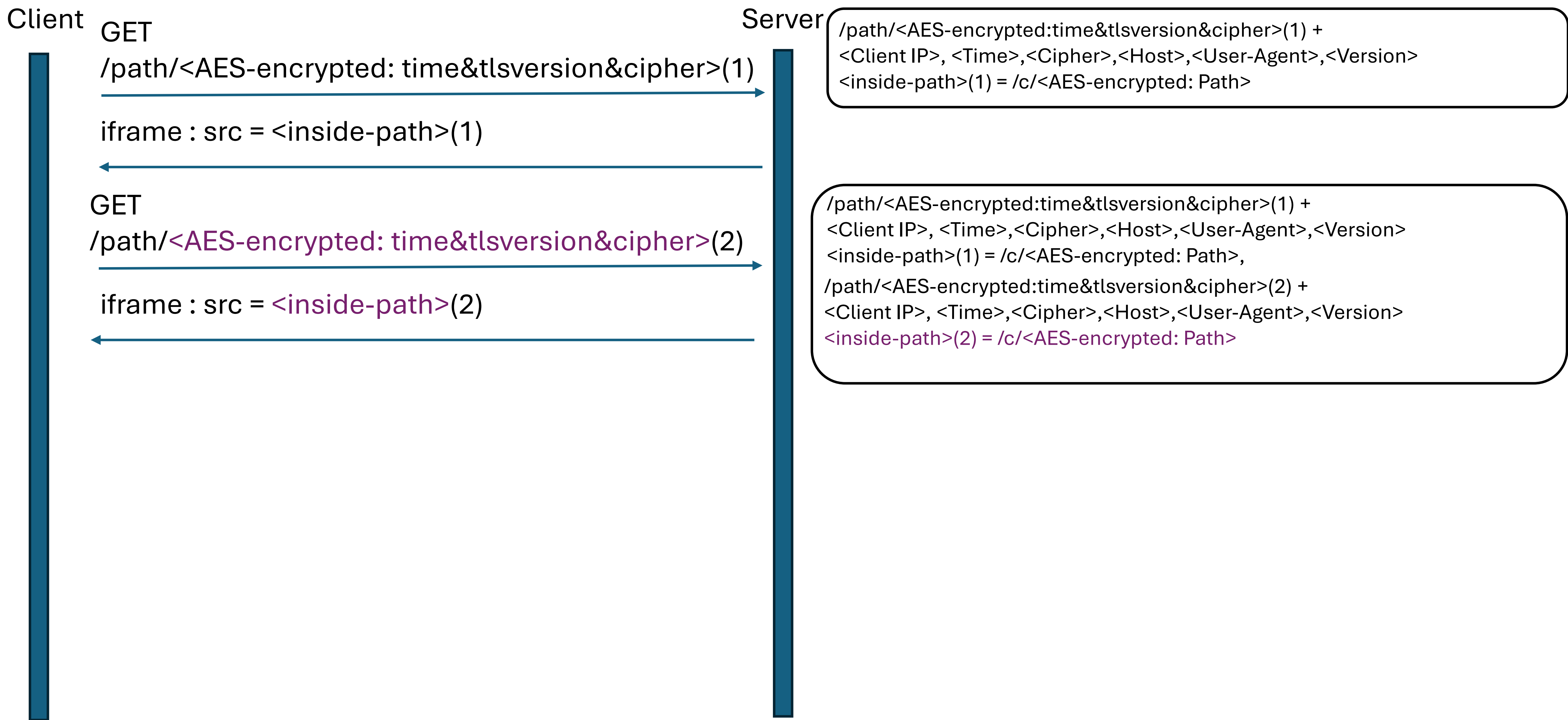


GET
/path/<AES-encrypted: time&tlsversion&cipher>(1)

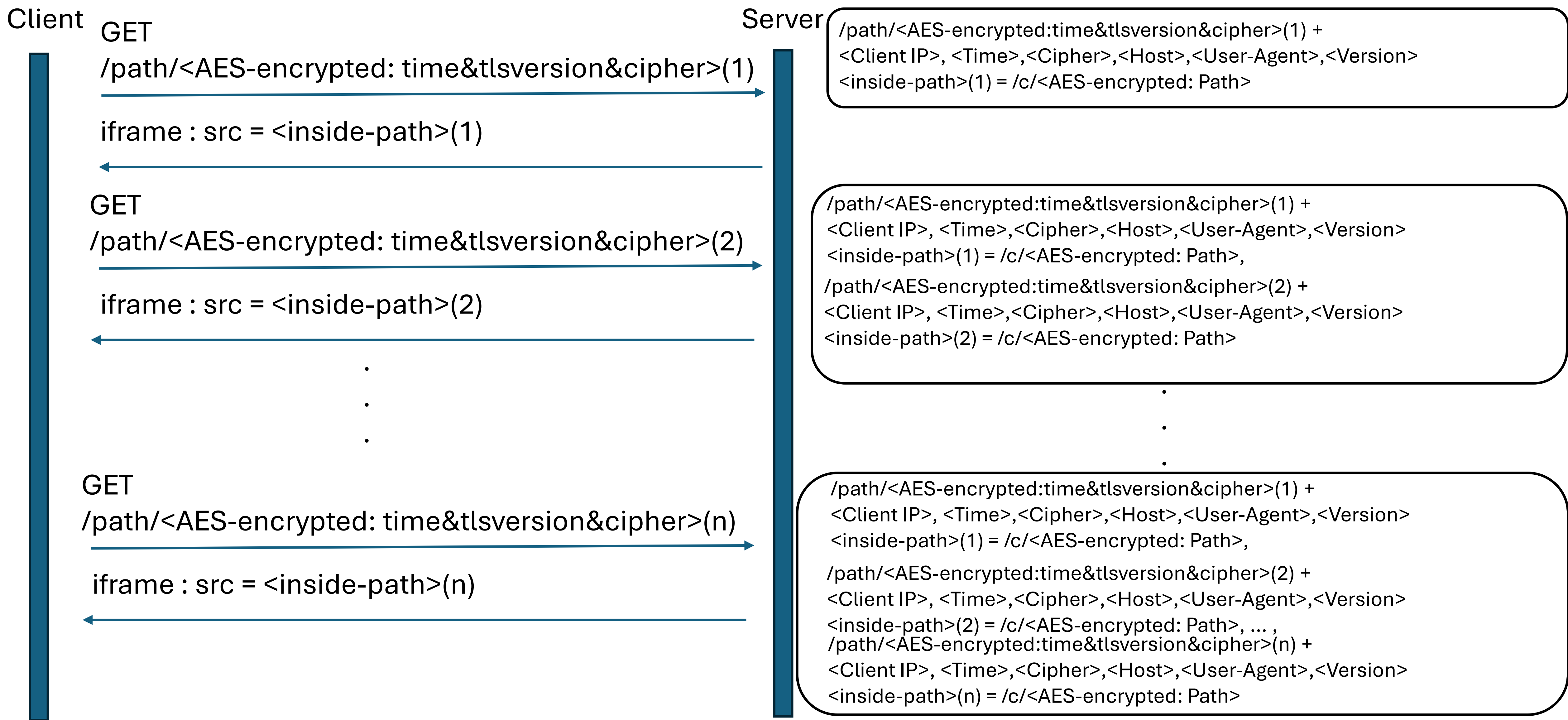
iframe : src = <inside-path>(1)

/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>

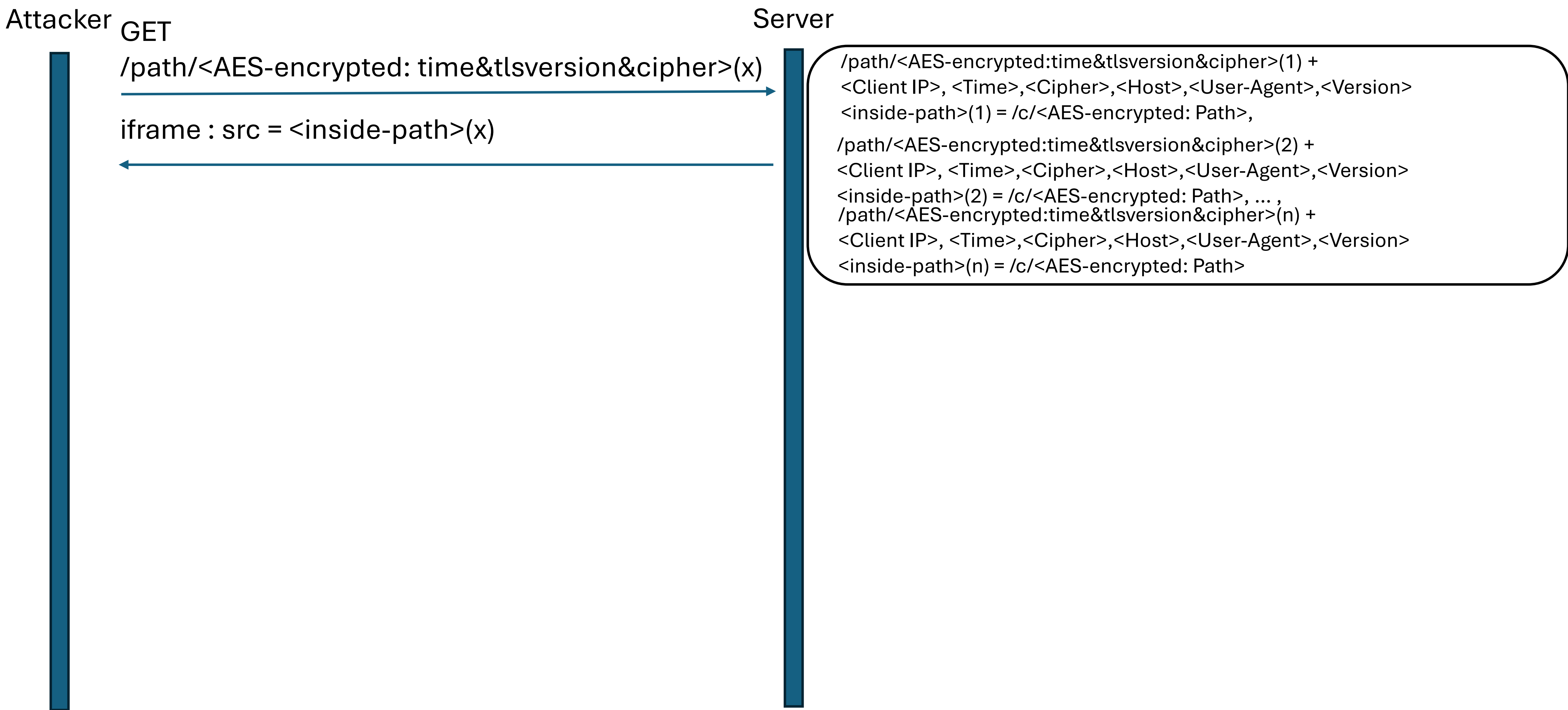
Methodology


Client
Server
GET
/path/<AES-encrypted: time&tlsversion&cipher>(1)
iframe : src = <inside-path>(1)
GET
/path/<AES-encrypted: time&tlsversion&cipher>(2)
iframe : src = <inside-path>(2)
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>

Methodology

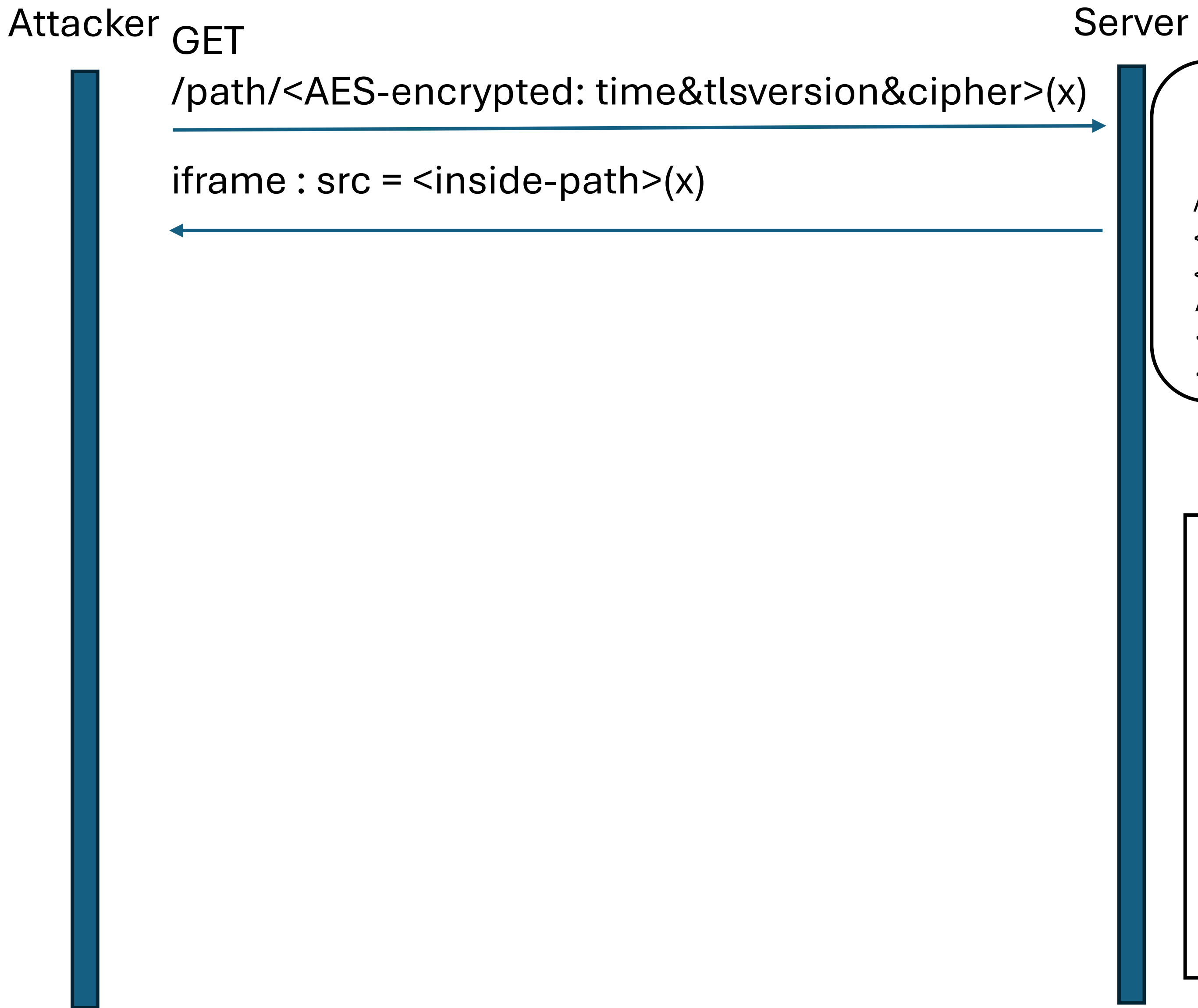

Client
Server
GET /path/<AES-encrypted: time&tlsversion&cipher>(1)
iframe : src = <inside-path>(1)
GET /path/<AES-encrypted: time&tlsversion&cipher>(2)
iframe : src = <inside-path>(2)
.
.
.
GET /path/<AES-encrypted: time&tlsversion&cipher>(n)
iframe : src = <inside-path>(n)
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>
.
.
.
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, ... ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>

Detection


Attacker
Server
GET
/path/<AES-encrypted: time&tlsversion&cipher>(x)
iframe : src = <inside-path>(x)

/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, ... ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>

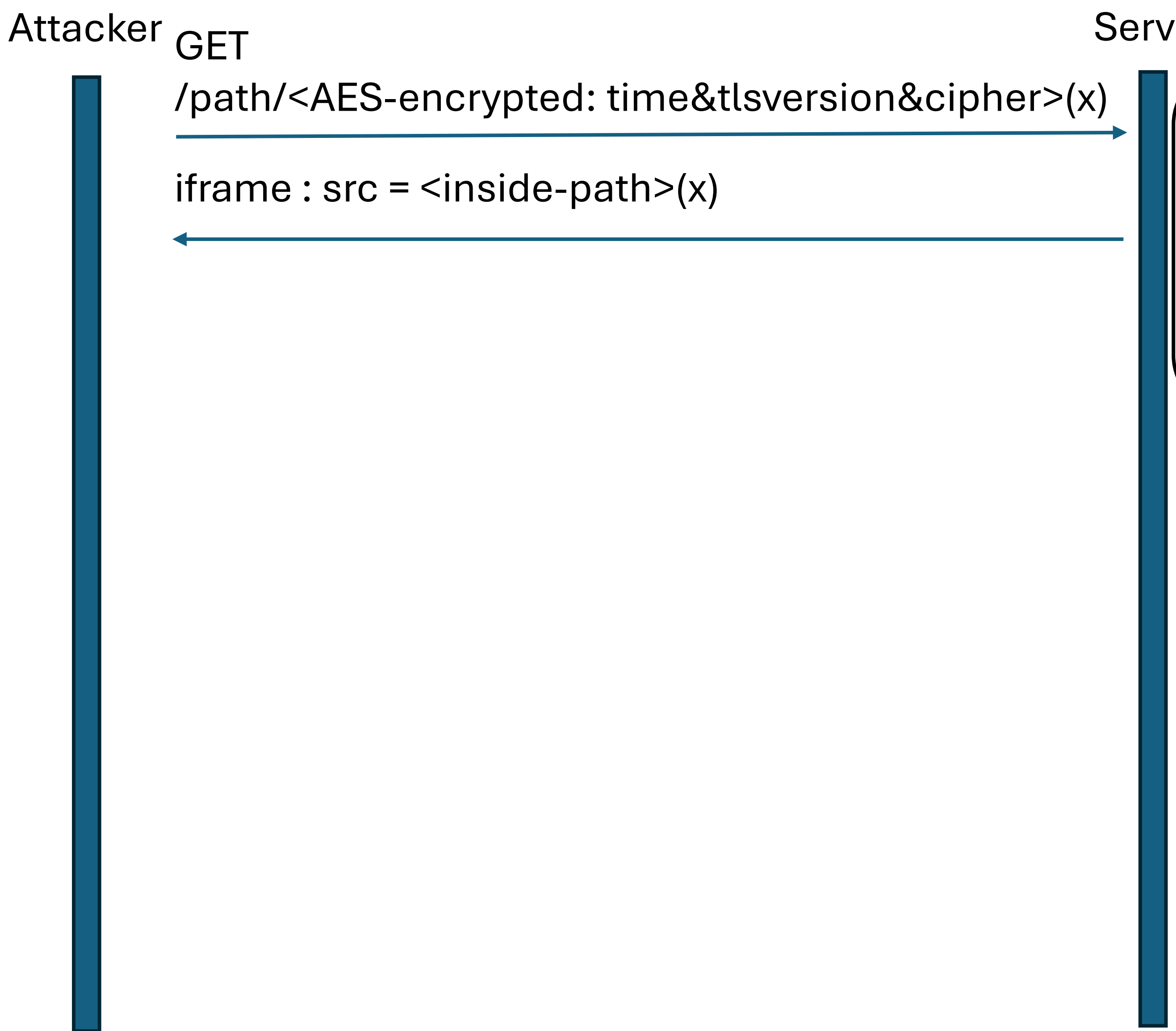
Detection


Attacker
GET
/path/<AES-encrypted: time&tlsversion&cipher>(x)
iframe : src = <inside-path>(x)
Server

/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, … ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>

Check Logs:

Detection


Attacker
Server
GET
/path/<AES-encrypted: time&tlsversion&cipher>(x)
iframe : src = <inside-path>(x)

```
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, ... ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>
```

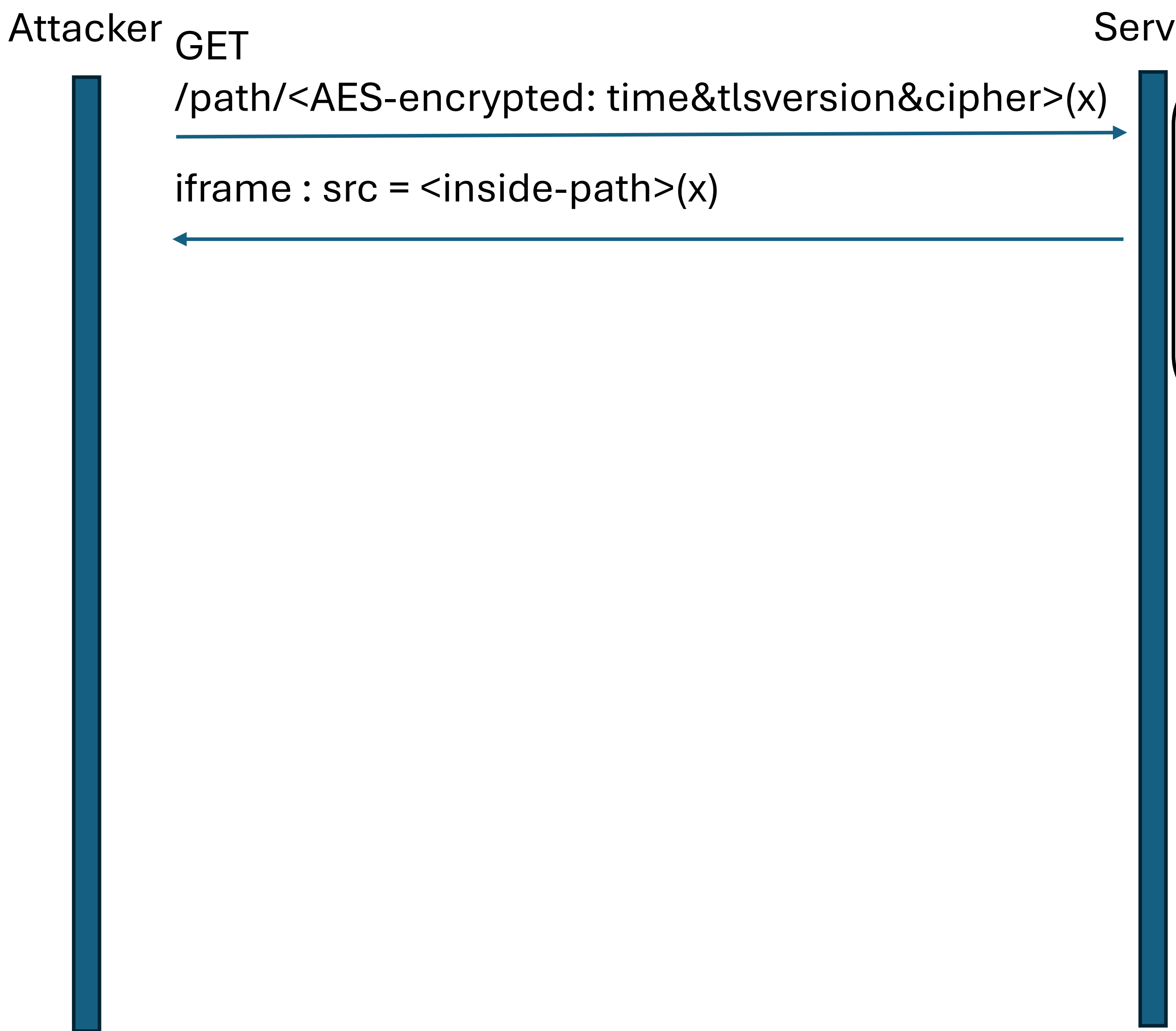
Check Logs:

(Cipher(Path), Version(Path))

==

(Cipher(Log), Version(Log))

Detection


Attacker
Server
GET
/path/<AES-encrypted: time&tlsversion&cipher>(x)
iframe : src = <inside-path>(x)

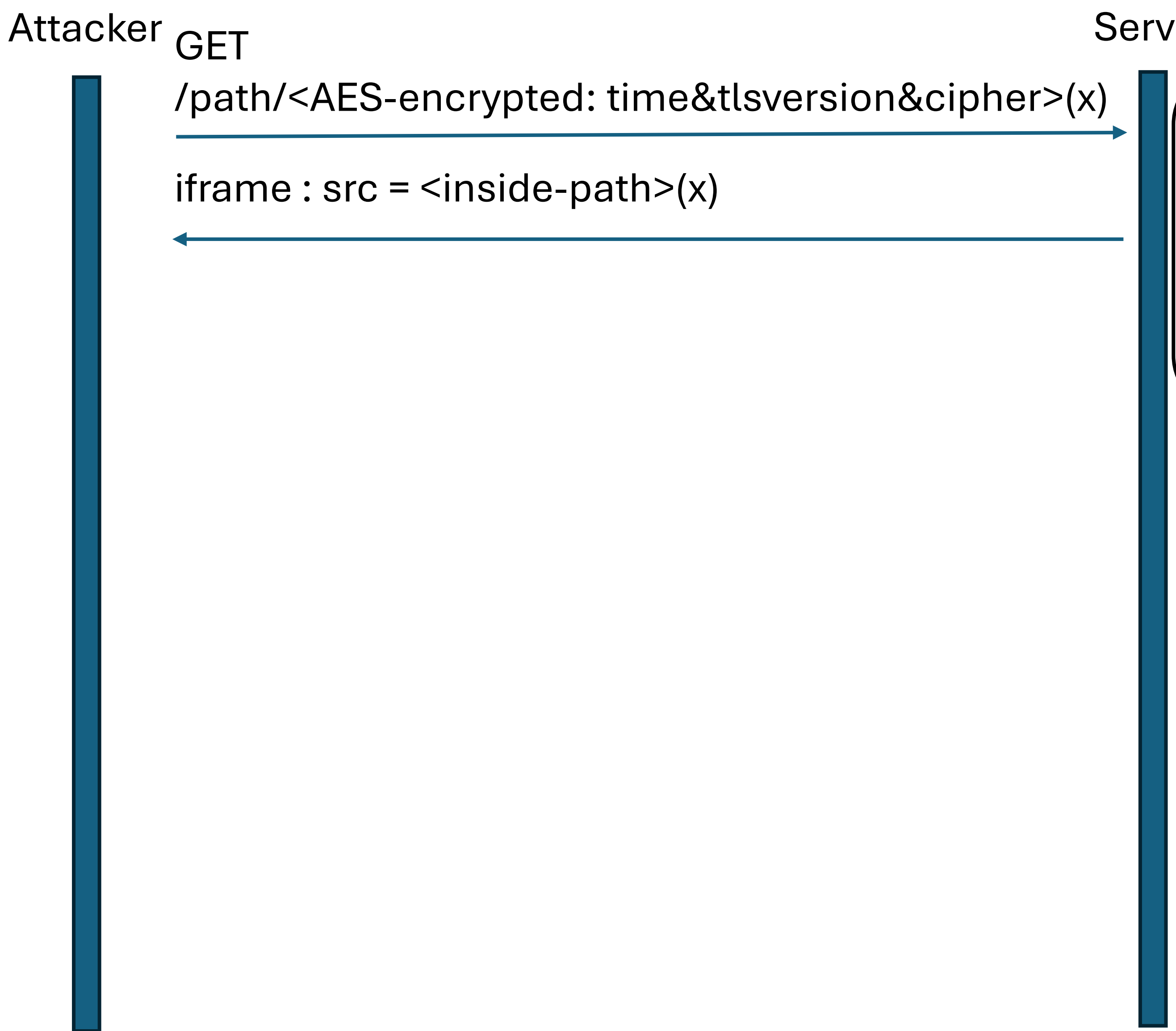
/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, ... ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>

Check Logs:

(Cipher(Path), Version(Path))
==
(Cipher(Log), Version(Log))

If not equal: indicative of Interception

Detection


Attacker
Server
GET
/path/<AES-encrypted: time&tlsversion&cipher>(x)
iframe : src = <inside-path>(x)

/path/<AES-encrypted:time&tlsversion&cipher>(1) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(1) = /c/<AES-encrypted: Path>,

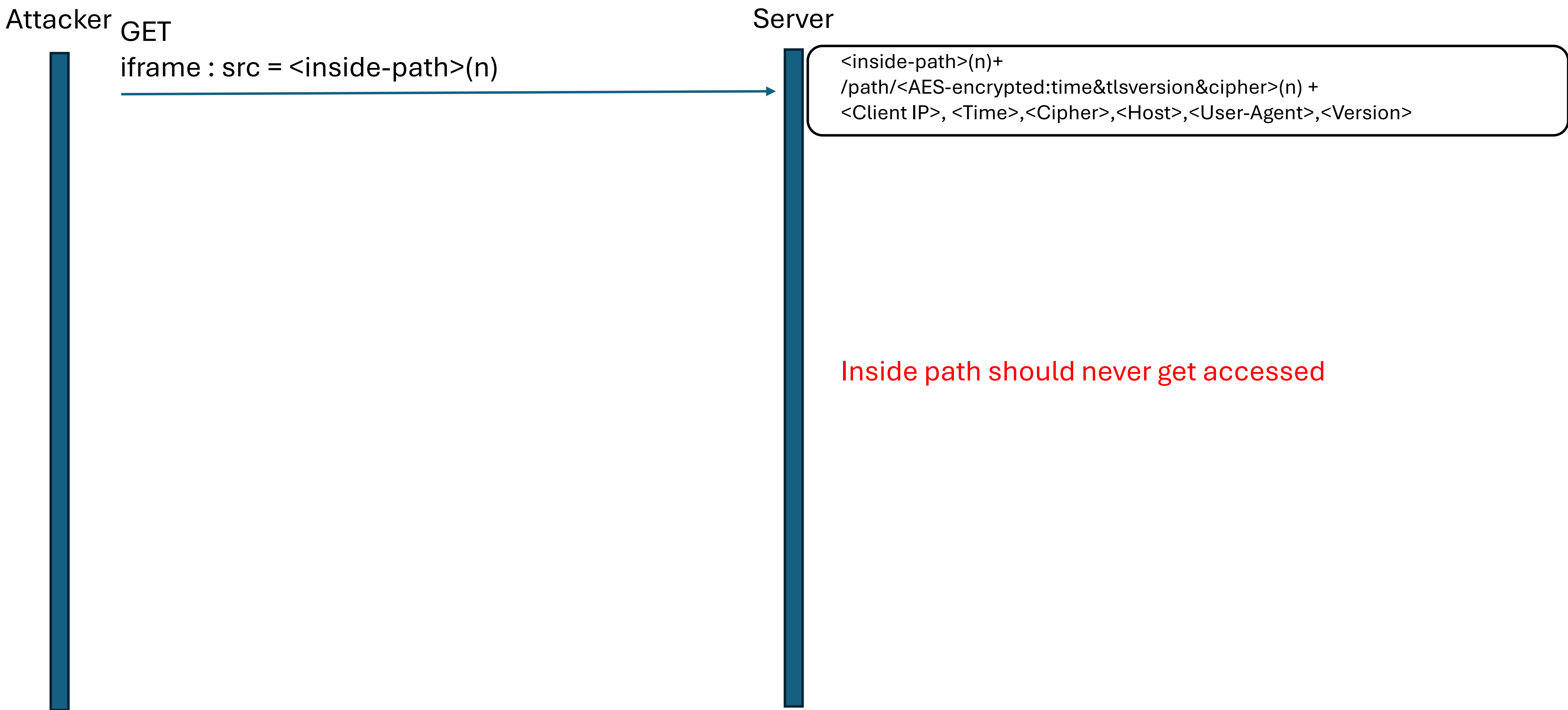
/path/<AES-encrypted:time&tlsversion&cipher>(2) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(2) = /c/<AES-encrypted: Path>, … ,
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>
<inside-path>(n) = /c/<AES-encrypted: Path>

Additional:

Check if diff(Time(Path),Time(Log)) „too great“

Check if Path already in log

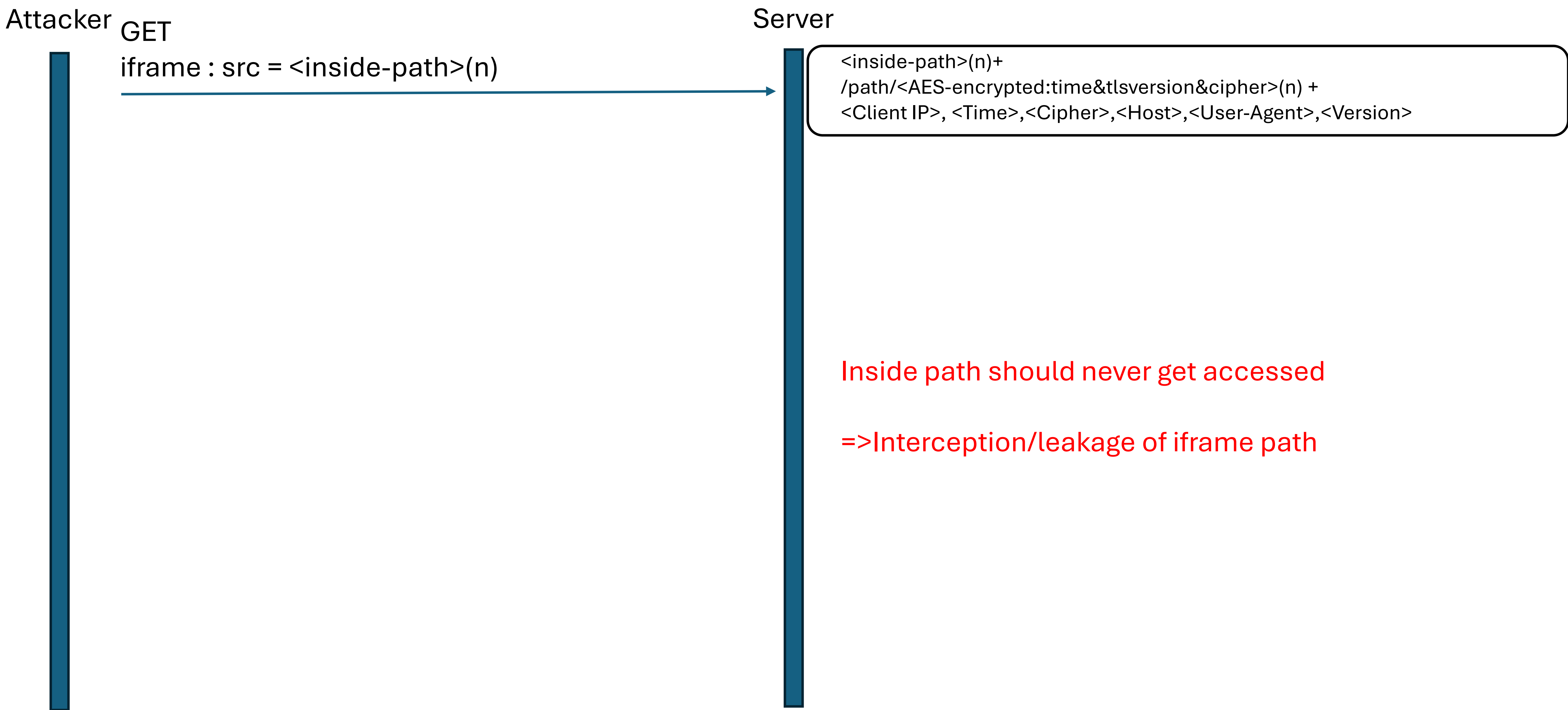
Detection



GET
iframe : src = <inside-path>(n)

<inside-path>(n)+
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>

Detection


Attacker
GET
iframe : src = <inside-path>(n)
Server
<inside-path>(n)+
/path/<AES-encrypted:time&tlsversion&cipher>(n) +
<Client IP>, <Time>,<Cipher>,<Host>,<User-Agent>,<Version>

Inside path should never get accessed

=>Interception/leakage of iframe path

Design Choices

- Using Nginx stream module
  - => to use it as proxy without terminating TLS
- Using Nginx stream preread module
  - => To read SNI without needing TLS
- Using Nginx proxy_protocol
  - =>To ensure the proxy can forward the real IP

Design Choices

- Using AES-GCM
  - =>Make sure that path content stays hidden
  - =>Random nonce => unique paths

Client Script

- Uses OpenSSL 1.1.1f and OpenSSL 1.0.2u
- Tries out every (cipher,version) combination
- Sends path request

VPN

- Diverse network paths
- Different Country = more interception?

- In this experiment :

  25 different VPN configurations using NordVPN

  24 different Countries

Experiment

- Running client script with each VPN
- On 14 different days between January 22 and February 15
- Regulary checking logs for anomalies

Results

- Total of 22,758 connections
- 144 distinct IP addresses
- No retransmission
- No access to iframe
- Embedded information in path match with logs
- No unexpected dropped connections

Results

- Total of 22,758 connections
- 144 distinct IP addresses
- No retransmission
- No access to iframe
- Embedded information in path match with logs
- No unexpected dropped connections

$\longrightarrow$ No interception?

Results

- Total of 22,758 connections
- 144 distinct IP addresses
- No retransmission
- No access to iframe
- Embedded information in path match with logs
- No unexpected dropped connections

$\longrightarrow$ No interception? maybe

Interpretation

- No manipulation on Ciphersuite or TLS version
  - More potential ways to intercept (e.g. decrpyting weak cipher encryption)
- No retransmission
  - No reason to: response just consists of iframe
- No access to iframe
  - does an attacker even care?

Limitations

- Heavily relies on attacker behavior
- No valuable content
- Short duration
- Only a few VPN servers

Reproducibility Client

- Requirements client:
  - (install OpenSSL 1.1.1f or higher)
  - Build OpenSSL 1.0.2u configured to support weak cipher
- Script uses subprocesses to get ciphers

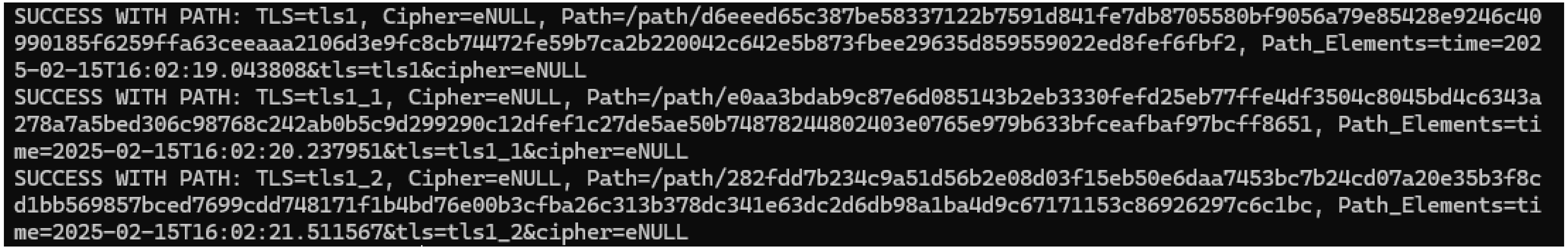
```
result = subprocess.run([openssl_path, "ciphers", "-v", "ALL"], capture_output=True, text=True)
```

- Script uses every (cipher,version) combination to send encrypted path

```
for cipher in legacy_ciphers:
    for tls_version in ["tls1", "tls1_1", "tls1_2"]:
        if (tls_version, cipher) not in tested_combinations:
        result = test_connection(args.legacy_openssl, cipher, tls_version, args.host, args.port,args.ignore_cert, "proxy1.com")
        results.append(result)
        tested_combinations.add((tls_version, cipher)
```

Reproducibility Client

- Logging depending of the http response



```
SUCCESS WITH PATH: TLS=tls1, Cipher=eNULL, Path=/path/d6eeed65c387be58337122b7591d841fe7db8705580bf9056a79e85428e9246c40
990185f6259ffa63ceeaaa2106d3e9fc8cb74472fe59b7ca2b220042c642e5b873fbee29635d859559022ed8fef6fbf2, Path_Elements=time=202
5-02-15T16:02:19.043808&tls=tls1&cipher=eNULL
SUCCESS WITH PATH: TLS=tls1_1, Cipher=eNULL, Path=/path/e0aa3bdab9c87e6d085143b2eb3330fefd25eb77ffe4df3504c8045bd4c6343a
278a7a5bed306c98768c242ab0b5c9d299290c12dfef1c27de5ae50b748782448024 03e0765e979b633bfceafbaf97bcff8651, Path_Elements=ti
me=2025-02-15T16:02:20.237951&tls=tls1_1&cipher=eNULL
SUCCESS WITH PATH: TLS=tls1_2, Cipher=eNULL, Path=/path/282fdd7b234c9a51d56b2e08d03f15eb50e6daa7453bc7b24cd07a20e35b3f8c
d1bb569857bced7699cdd748171f1b4bd76e00b3cfba26c313b378dc341e63dc2d6db98a1ba4d9c67171153c86926297c6c1bc, Path_Elements=ti
me=2025-02-15T16:02:21.511567&tls=tls1_2&cipher=eNULL
```

- Using OpenVPN

- Bash script used to connect to VPNs and run client script

Reproducibility Server

- Requirements:
  - 2 Nginx instances built with OpenSSL1.1.1f and OpenSSL 1.0.2u (with weak cipher)

```
/home/okan/openssl-1.0.2u/.openssl/include/openssl/ssl.h: objs/Makefile
 cd /home/okan/openssl-1.0.2u \
 && if [ -f Makefile ]; then $(MAKE) clean; fi \
 && ./config --prefix=/home/okan/openssl-1.0.2u/.openssl \
 no-shared no-threads zlib \
 enable-weak-ssl-ciphers enable-ssl2 enable-rc5 enable-rc2 \
 enable-cms enable-md2 enable-mdc2 enable-ec enable-ec2m \
 enable-ecdh enable-ecdsa enable-seed enable-camellia enable-idea \
 enable-rfc3779 \
 && $(MAKE) \
 && $(MAKE) install_sw LIBDIR=lib
```

  - Configure Nginx with stream,ssl,preread,realip module
  - Flask

Reproducibility Server

- One Nginx instance proxy and webserver

```
server {
      listen 443;
      proxy_pass $upstream_server;
      ssl_preread on;
      proxy_protocol on;
}
```

```
listen 8002 ssl proxy_protocol;
```

- Other Nginx instance webserver
- Certificates OpenSSL generated

Reproducibility Server

- Flask handles requests

```
location / {
	proxy_pass http://127.0.0.1:5000;
	proxy_set_header SSL-PROTOCOL $ssl_protocol;
	proxy_set_header SSL-CIPHER $ssl_cipher;
	proxy_set_header Host $host;
	proxy_set_header X-Real-IP $remote_addr;
	proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
	proxy_set_header X-Forwarded-Proto $scheme;
}
```

```
@app.route('/path/<encrypted_path_hex>', methods=['GET'])

@app.route('/c/<encrypted_data>', methods=['GET'])
```

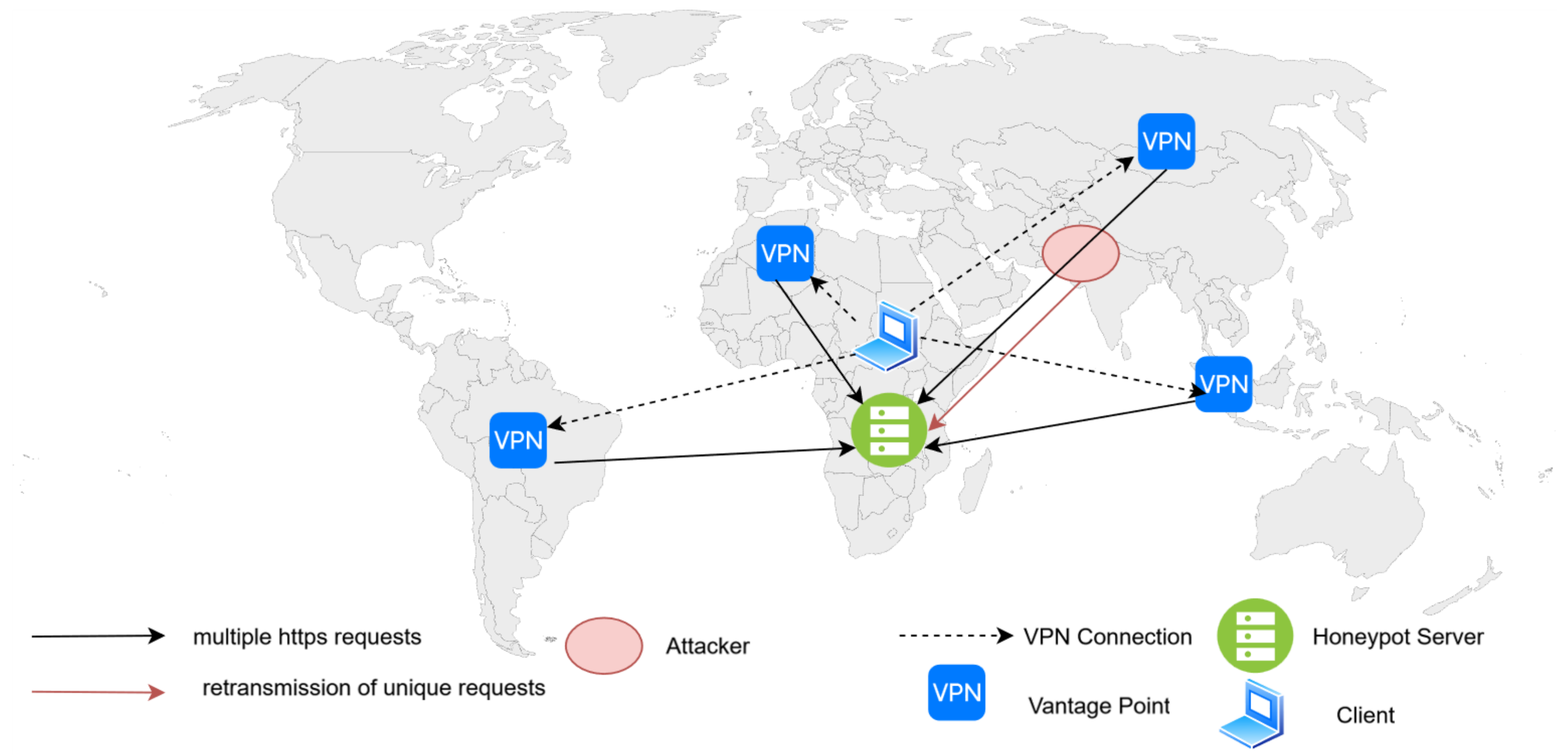
Reproducibility Server

- Flask logs accordingly

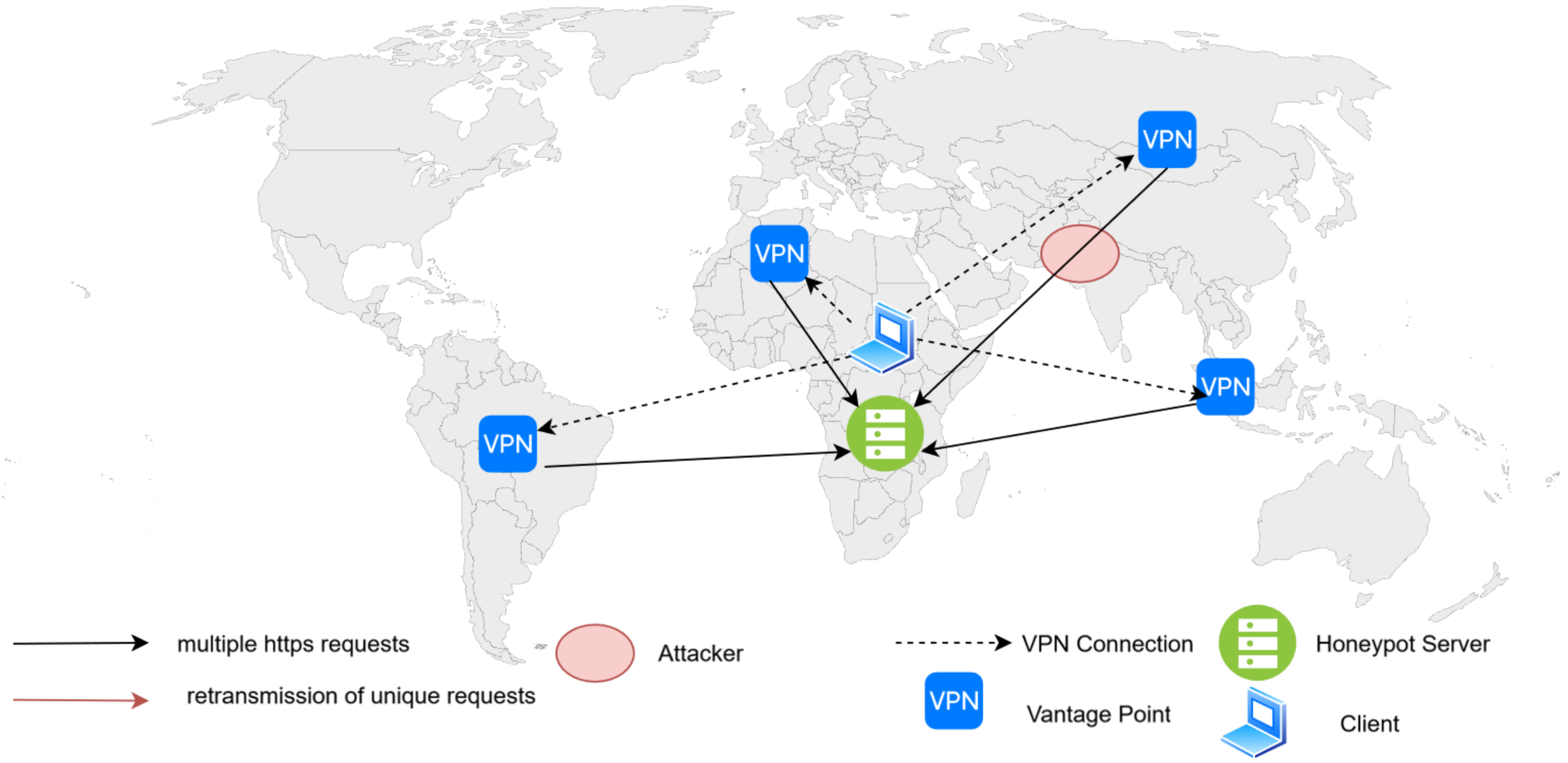
```
log.write(
f"Accessed path: {decrypted_path_elements}\n"
f"Client IP: {client_ip}\n"
f"x_real_ip: {x_real_ip}\n"
f"x_forwarded_for: {x_forwarded_for}\n"
f"Host: {vHost}\n"
f"User-Agent: {user_agent}\n"
f"TLS Version: {tls_version}\n"
f"Cipher Suite: {cipher_suite}\n"
f"Timestamp: {datetime.utcnow().isoformat()}\n"
f"Encrpted Path: {encrypted_path_hex}\n"
f"Inside Path: {inside_path}\n"
f"{'-'*60}\n"
)
```

```
log.write(
f"Accessed Main path: {decrypt(encrypted_data,key)}\n"
f"Main path Elements: {decrypt(decrypt(encrypted_data,key),key)}\n"
f"Client IP: {client_ip}\n"
f"x_real_ip: {x_real_ip}\n"
f"x_forwarded_for: {x_forwarded_for}\n"
f"Host: {vHost}\n"
f"User-Agent: {user_agent}\n"
f"TLS Version: {tls_version}\n"
f"Cipher Suite: {cipher_suite}\n"
f"Timestamp: {datetime.utcnow().isoformat()}\n"
f"Inside Path Value: {encrypted_data}\n"
f"{'-'*60}\n"
)
```

Methodology

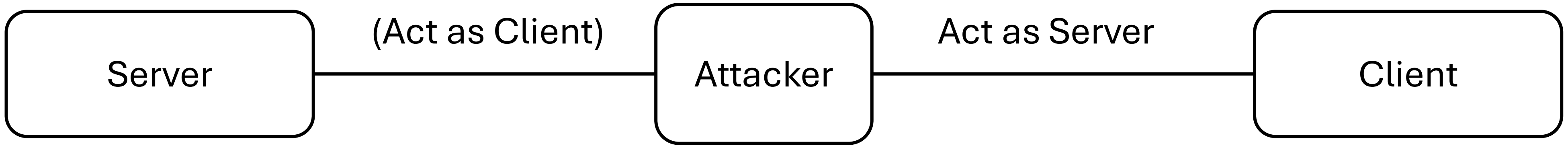


Methodology



ServerHello Fingerprinting

- Contains : TLS Version, Ciphersuite, Extensions
- Integratable to Honeypot
- Needs „unique“ ServerHello
- Assumes attacker doesnt replicate original ServerHello



„Unique“ Server Hello

- Extension order
- Custom extensions

```
int SSL_CTX_add_server_custom_ext(SSL_CTX *ctx, unsigned int ext_type,
                           custom_ext_add_cb add_cb,
                           custom_ext_free_cb free_cb, void *add_arg,
                           custom_ext_parse_cb parse_cb,
                           void *parse_arg);
```

Questions?