HTTP
Encrypted
Information can be
Stolen through
TCP-windows

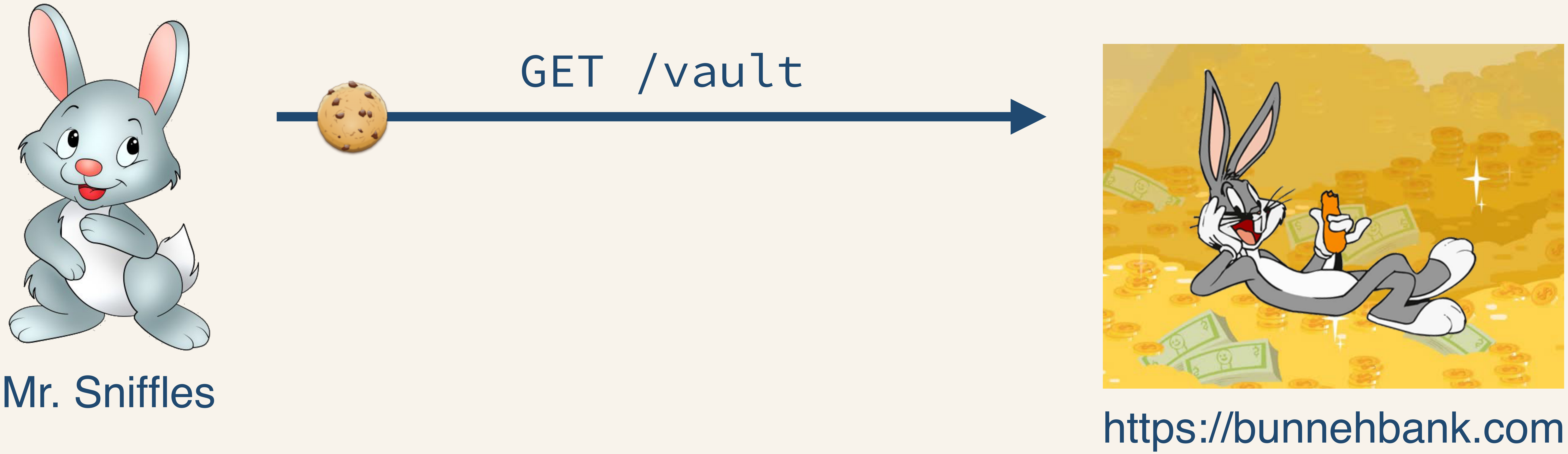
by

Mathy Vanhoef & Tom Van Goethem

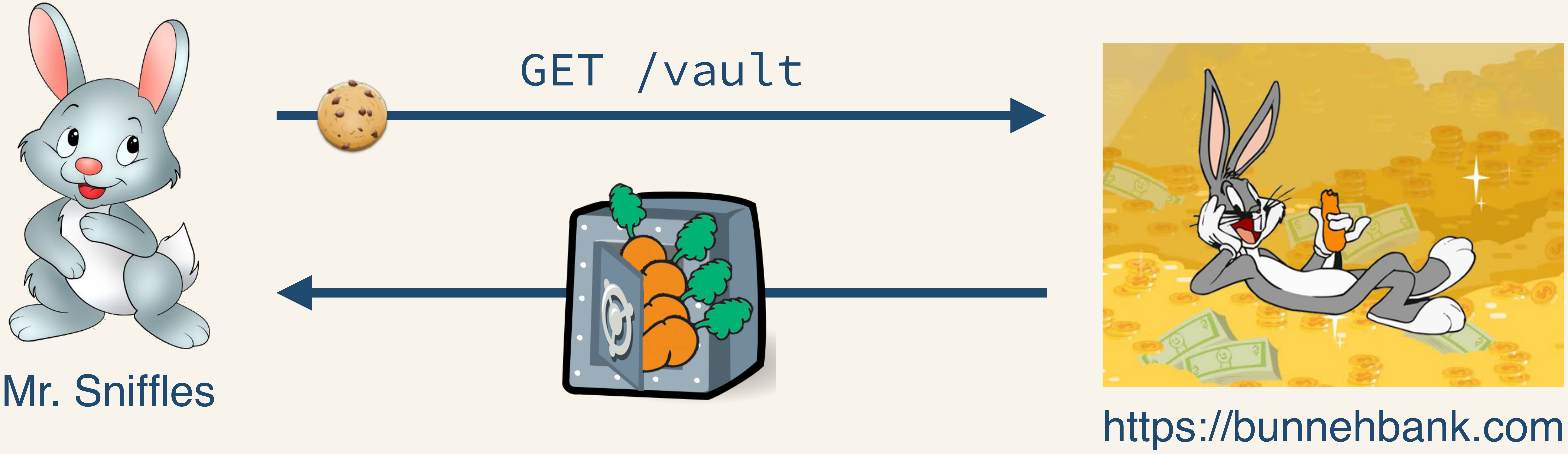
Agenda

- Technical background
  - Same-Origin Policy
  - Compression-based attacks
  - SSL/TLS & TCP
- Nitty gritty HEIST details
- Demo
- Countermeasures

Same-Origin Policy



Same-Origin Policy



the World Wide Web

Mr. Sniffles

https://bunnehbank.com

the World Wide Web

JS

Mr. Sniffles

https://bunnehbank.com

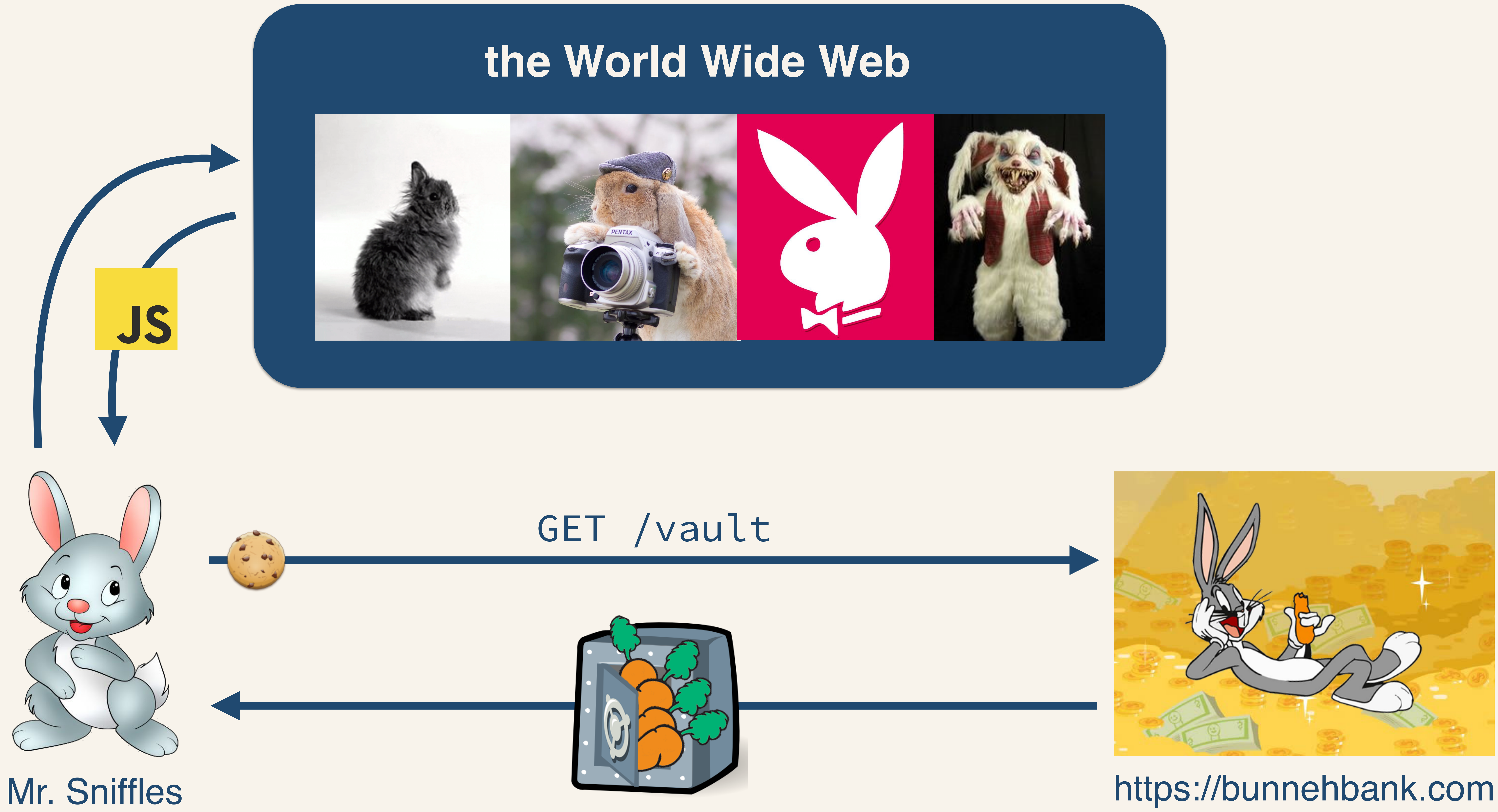
the World Wide Web

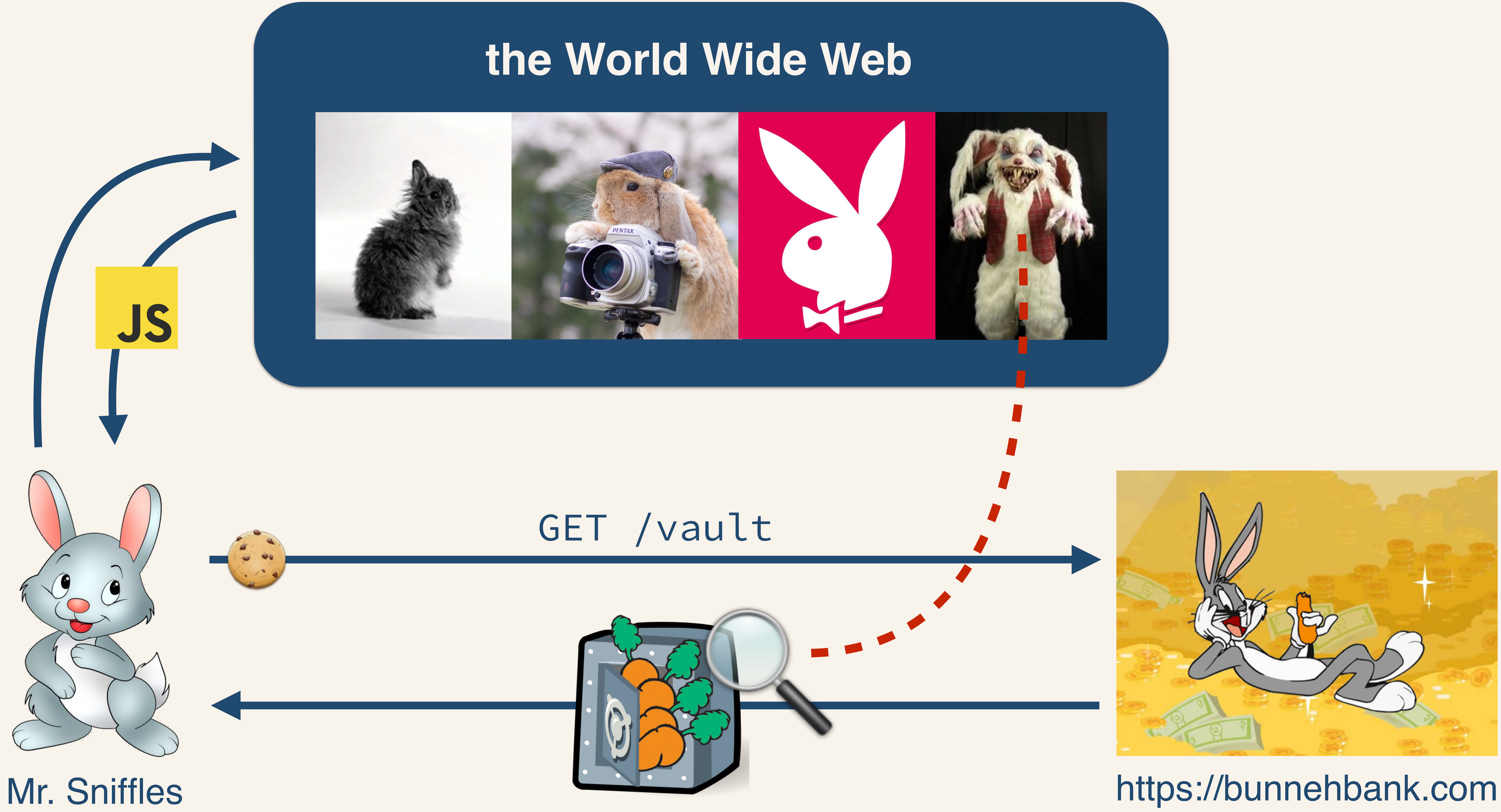
JS

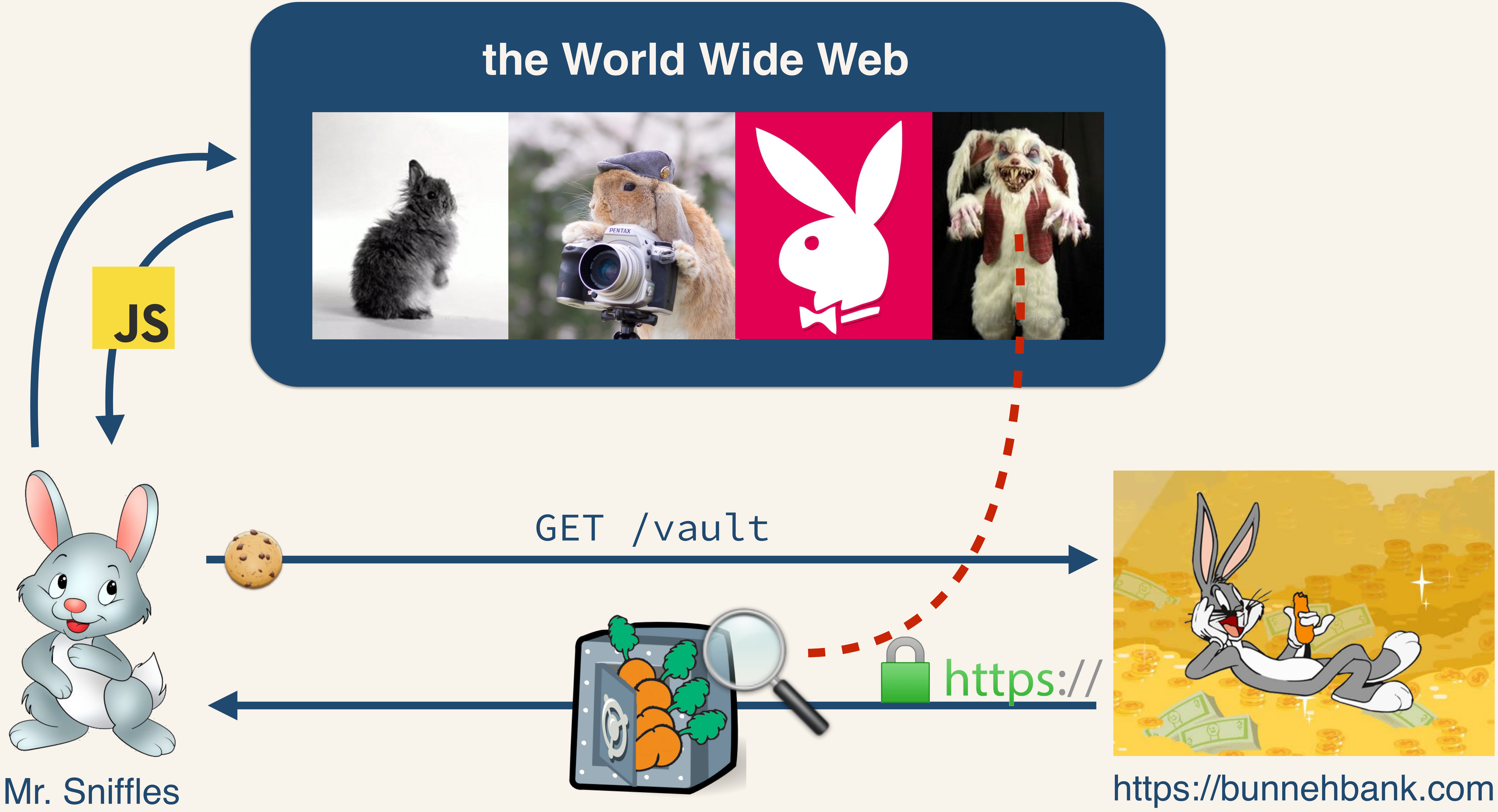
Mr. Sniffles

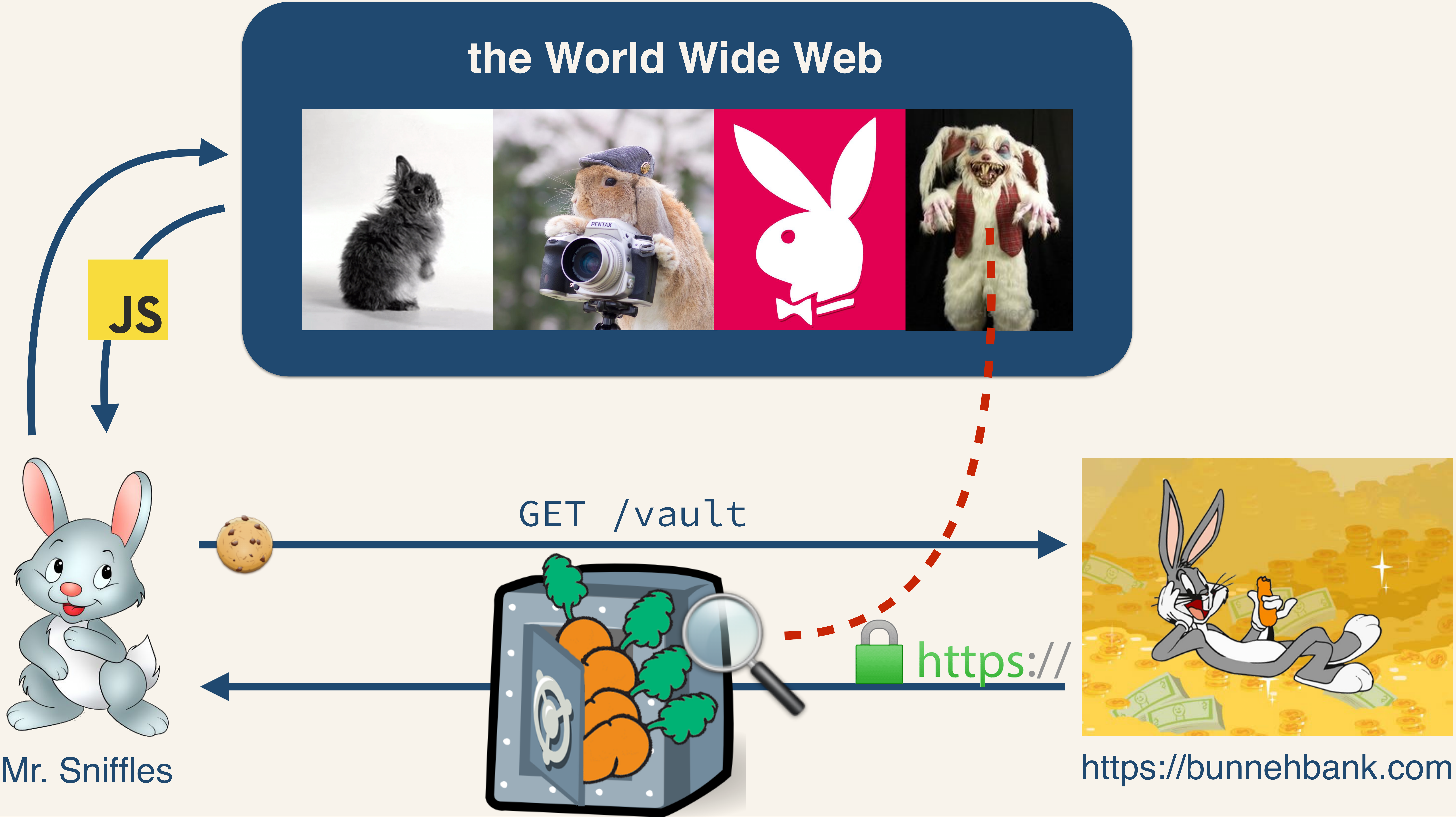
`GET /vault`

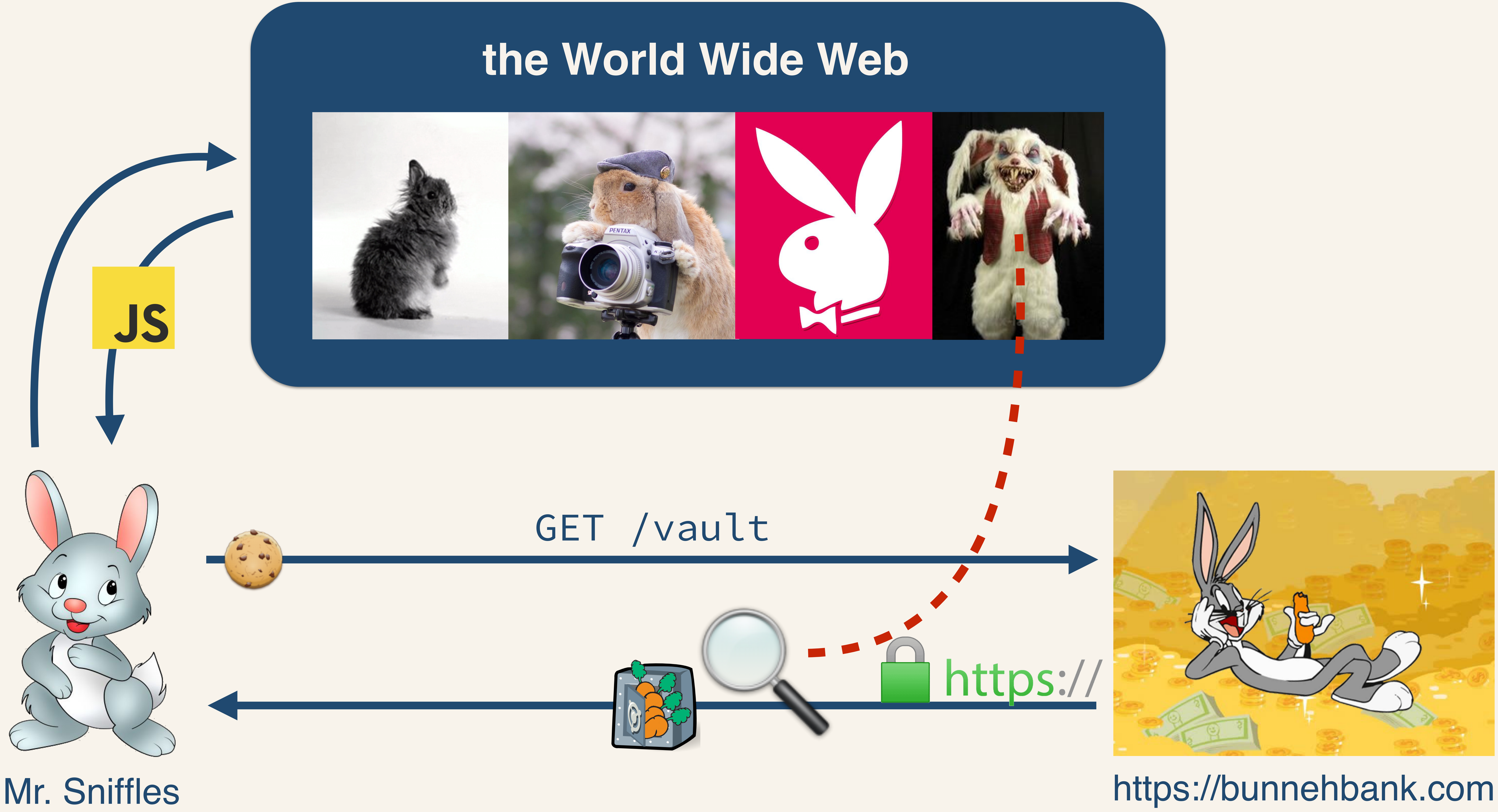
https://bunnehbank.com


the World Wide Web
JS
GET /vault
Mr. Sniffles
https://bunnehbank.com


the World Wide Web
JS
GET /vault
Mr. Sniffles
https://bunnehbank.com


the World Wide Web
JS
GET /vault
https://
Mr. Sniffles
https://bunnehbank.com


the World Wide Web
JS
GET /vault
https://
Mr. Sniffles
https://bunnehbank.com


the World Wide Web
JS
GET /vault
https://
Mr. Sniffles
https://bunnehbank.com

Agenda

- Technical background
  - Same-Origin Policy
  - **Compression-based attacks**
  - SSL/TLS & TCP
- Nitty gritty HEIST details
- Demo
- Countermeasures

/vault

Uncompressed

```
You requested:
  /vault

vault_secret=carrots4life
```

→ 51 bytes

Compressed

```
You requested:
  /vault

     _secret=carrots4life
```

→ 47 bytes

/vault?secret=a

You requested:
  /vault?secret=a

_   carrots4life

→ 50 bytes

/vault?secret=c

You requested:
  /vault?secret=c

_   arrots4life

→ 49 bytes

`/vault?secret=a`

`/vault?secret=c`

49 bytes < 50 bytes → 'c' is a correct guess

→ 50 bytes

→ 49 bytes

/vault?secret=ca

You requested:
/vault?secret=ca

_ rrots4life

→ 49 bytes

/vault?secret=cb

You requested:
/vault?secret=cb

_ arrots4life

→ 50 bytes

/vault?secret=ca

/vault?secret=cb

49 bytes < 50 bytes → 'ca' is a correct guess

→ 49 bytes

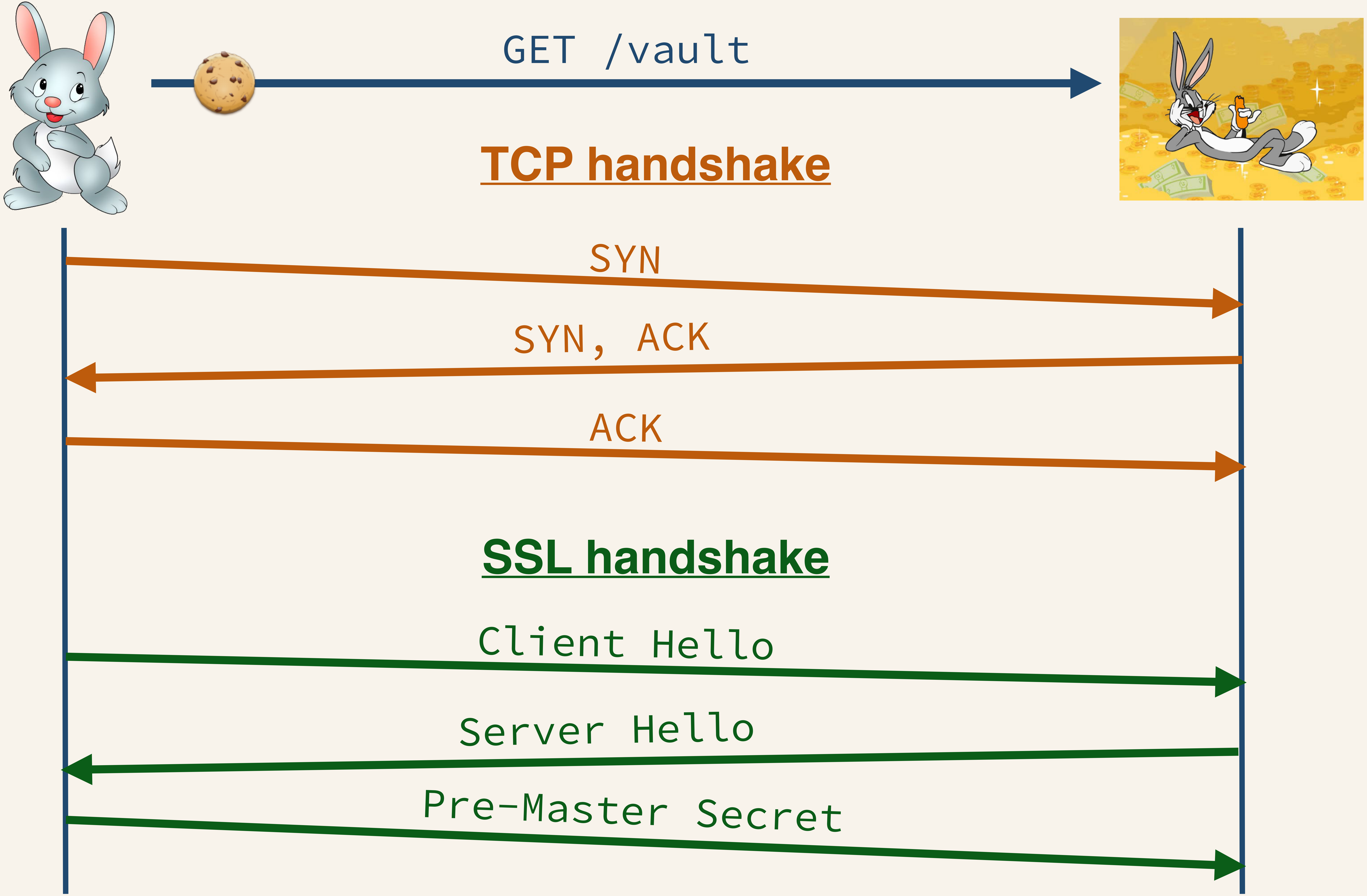
→ 50 bytes

Compression-based Attacks

- Compression and Information Leakage of Plaintext [FSE'02]
  - Chosen plaintext + compression = plaintext leakage
- CRIME [ekoparty'12]
  - Exploits SSL compression
- BREACH [Black Hat USA'13]
  - Exploits HTTP compression

Agenda

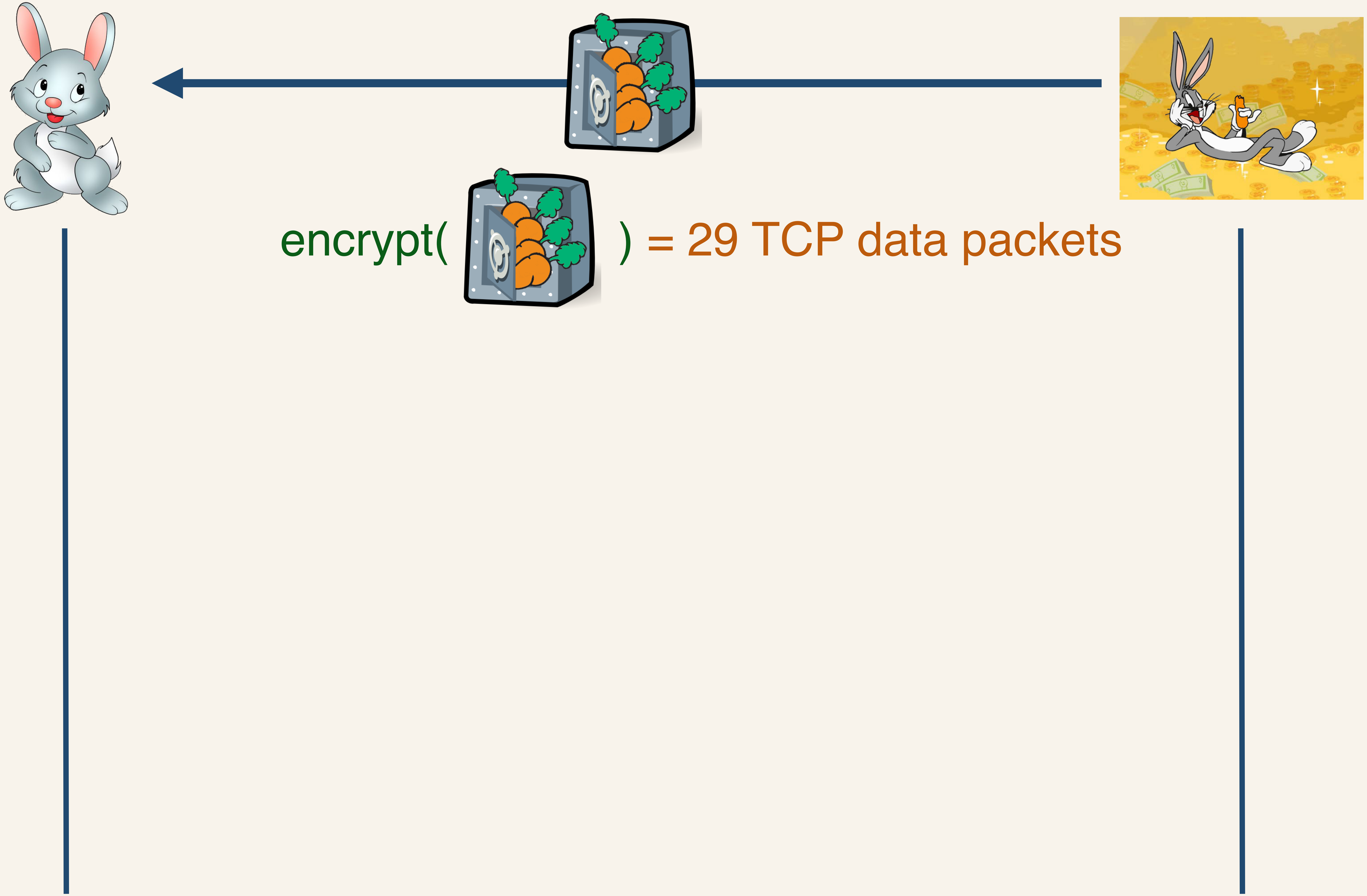
- Technical background
  - Same-Origin Policy
  - Compression-based attacks
  - **SSL/TLS & TCP**
- Nitty gritty HEIST details
- Demo
- Countermeasures

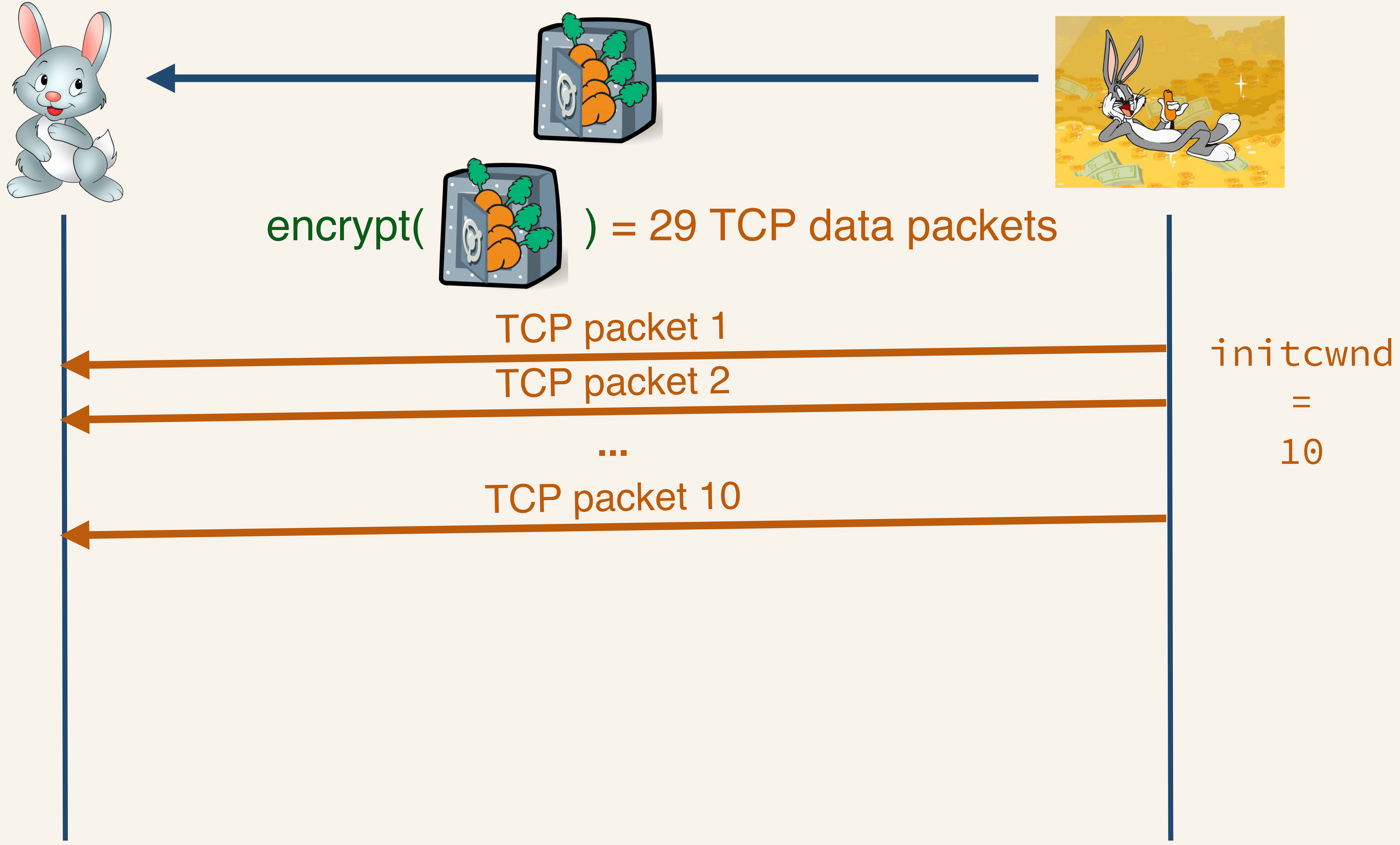

GET /vault
TCP handshake
SYN
SYN, ACK
ACK
SSL handshake
Client Hello
Server Hello
Pre-Master Secret

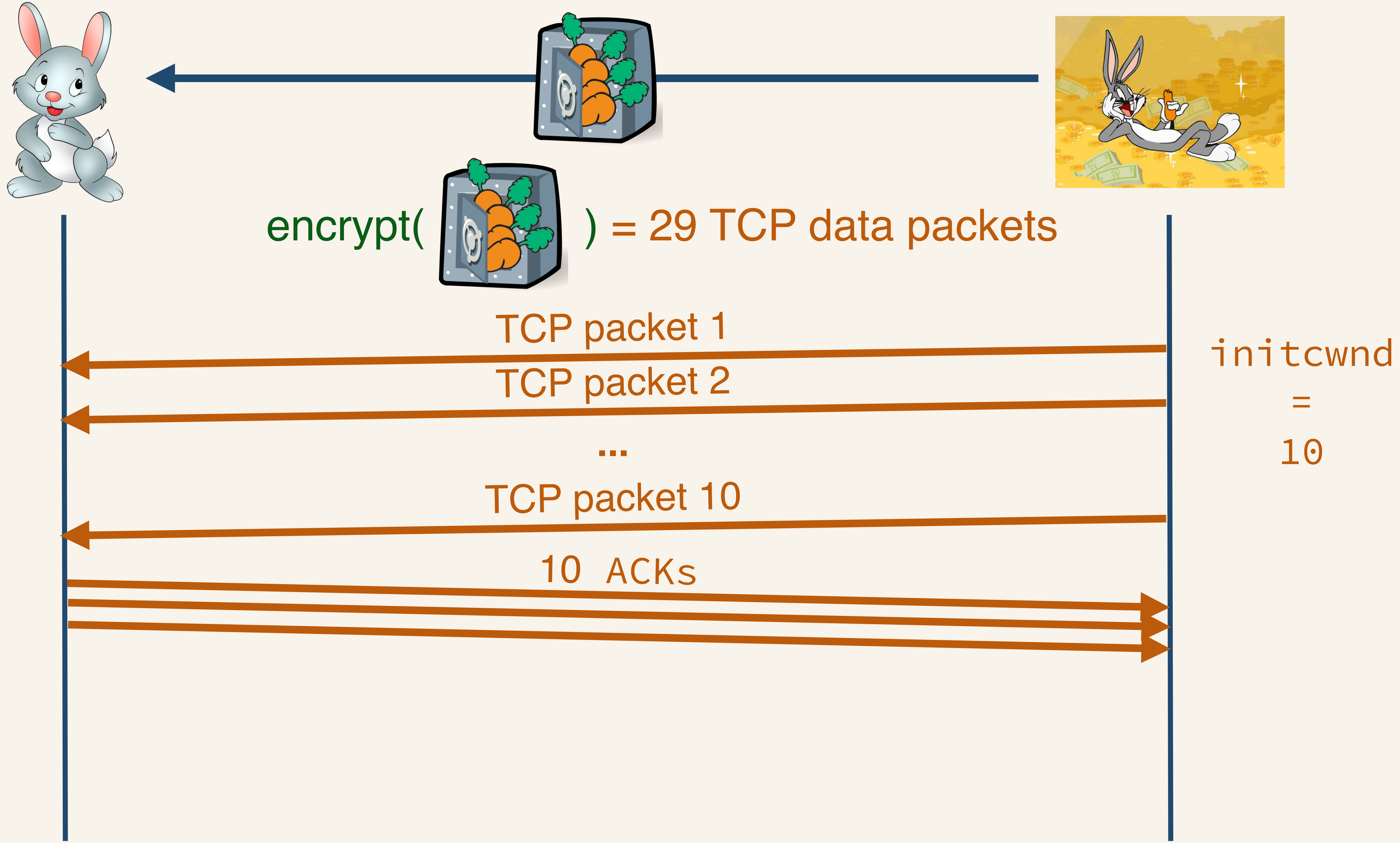
GET /vault

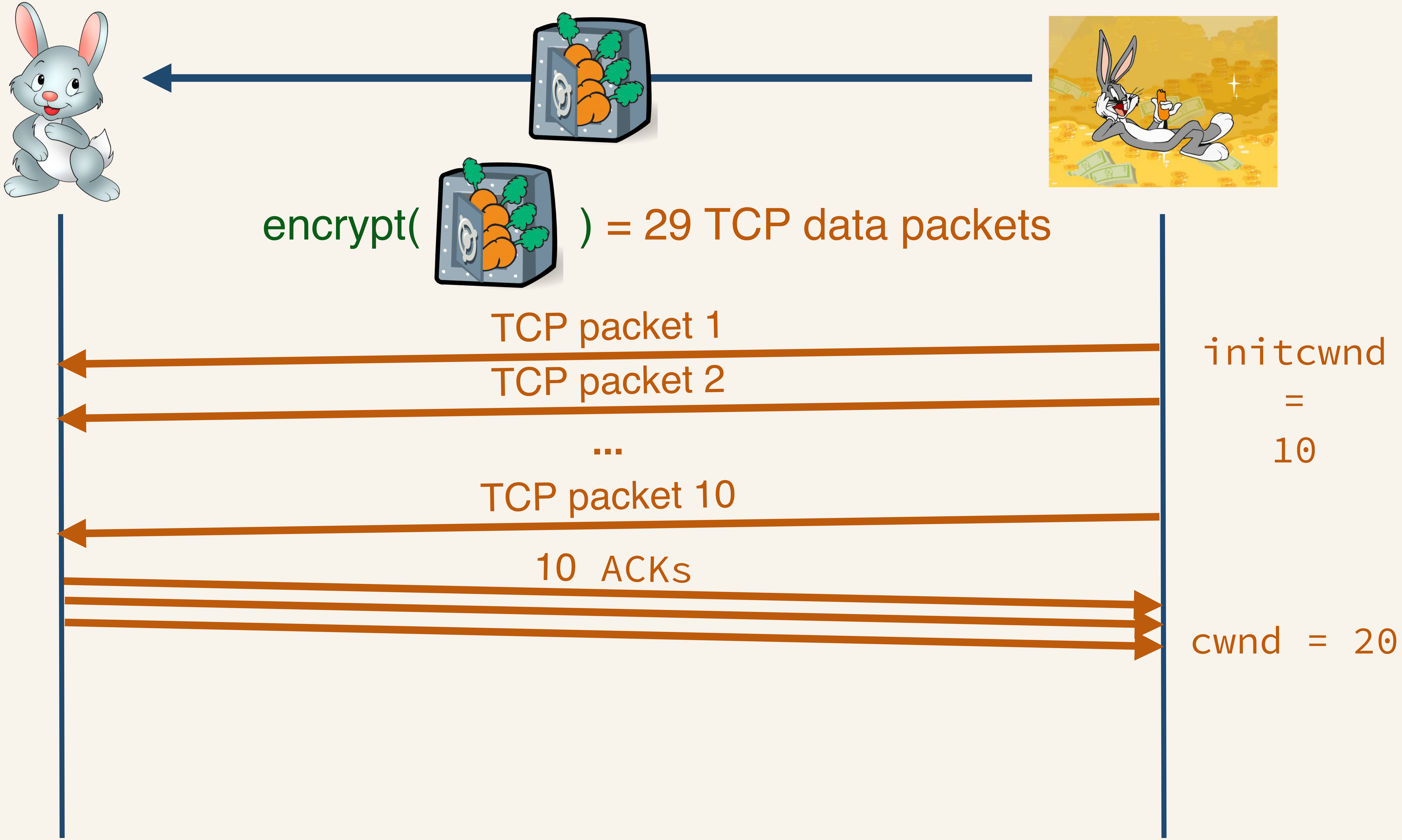
encrypt(
GET /vault HTTP/1.1
Cookie: user=mr.sniffles
Host: bunnehbank.com
....
)

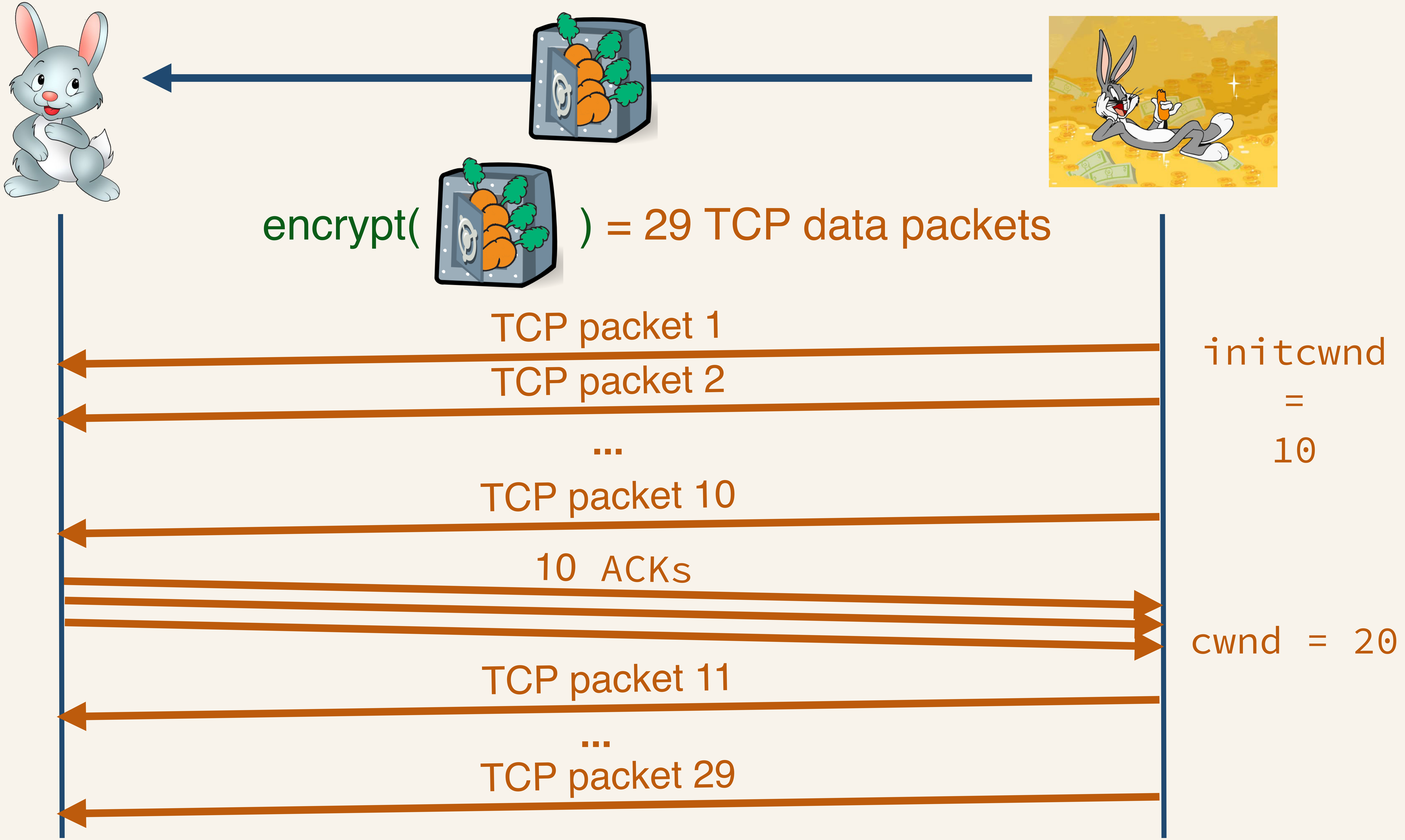
1 TCP data packet


encrypt( ) = 29 TCP data packets


encrypt( ) = 29 TCP data packets
TCP packet 1
TCP packet 2
...
TCP packet 10
initcwnd
=
10


encrypt( ) = 29 TCP data packets
TCP packet 1
TCP packet 2
...
TCP packet 10
10 ACKs
initcwnd
=
10


encrypt( ) = 29 TCP data packets
TCP packet 1
TCP packet 2
...
TCP packet 10
10 ACKs
initcwnd
=
10
cwnd = 20


encrypt( ) = 29 TCP data packets
TCP packet 1
TCP packet 2
...
TCP packet 10
initcwnd
=
10
10 ACKs
cwnd = 20
TCP packet 11
...
TCP packet 29

HEIST

- A set of techniques that allow attacker to determine the exact size of a network response
- ... **purely in the browser**
- Can be used to perform compression-based attacks, such as CRIME and BREACH, in the browser

Browser Side-channels

- Send authenticated request to `/vault` resource

```
fetch('https://bunnehbank.com/vault',
        {mode: "no-cors", credentials:"include"})
```

- Returns a `Promise`, which resolves as soon as browser receives the first byte of the response

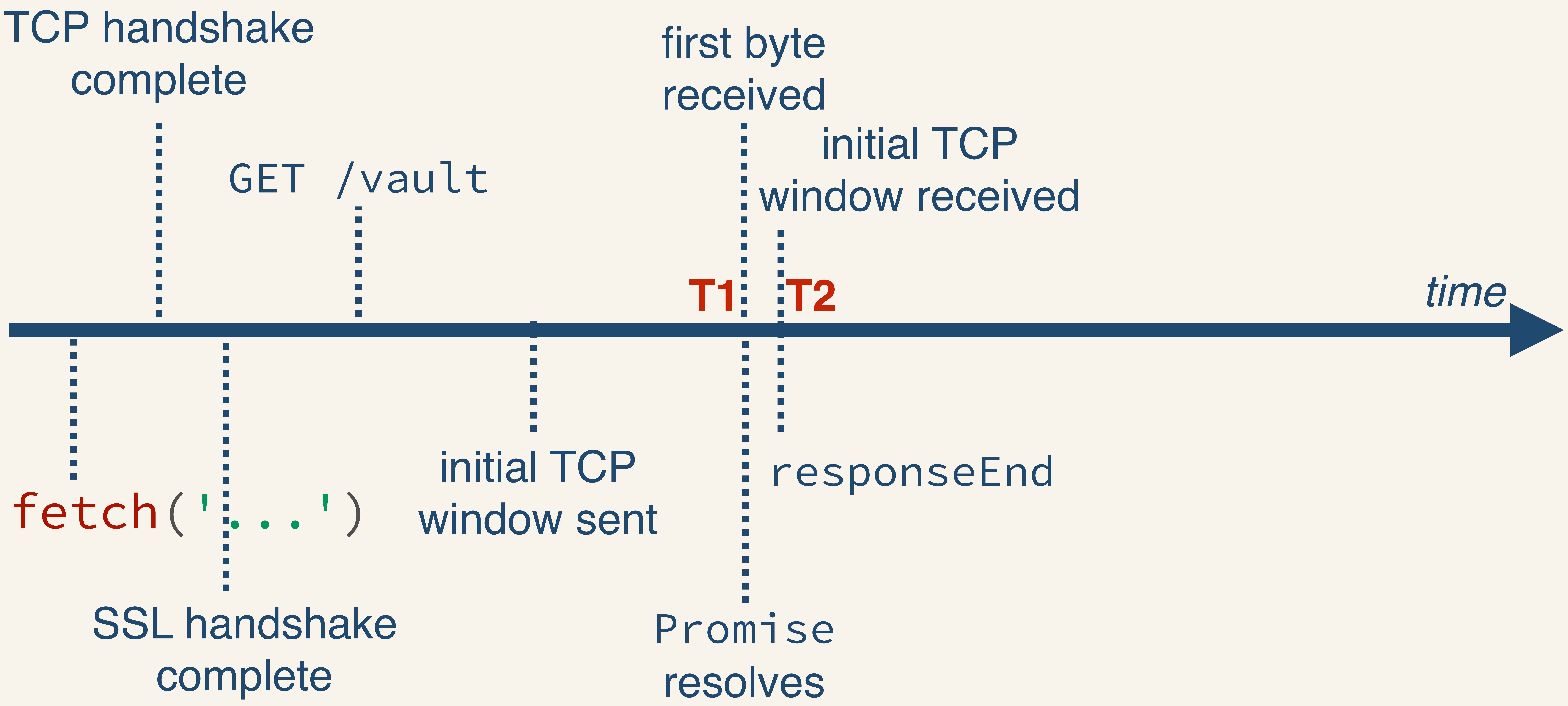
```
performance.getEntries()[-1].responseEnd
```

- Returns time when response was completely downloaded

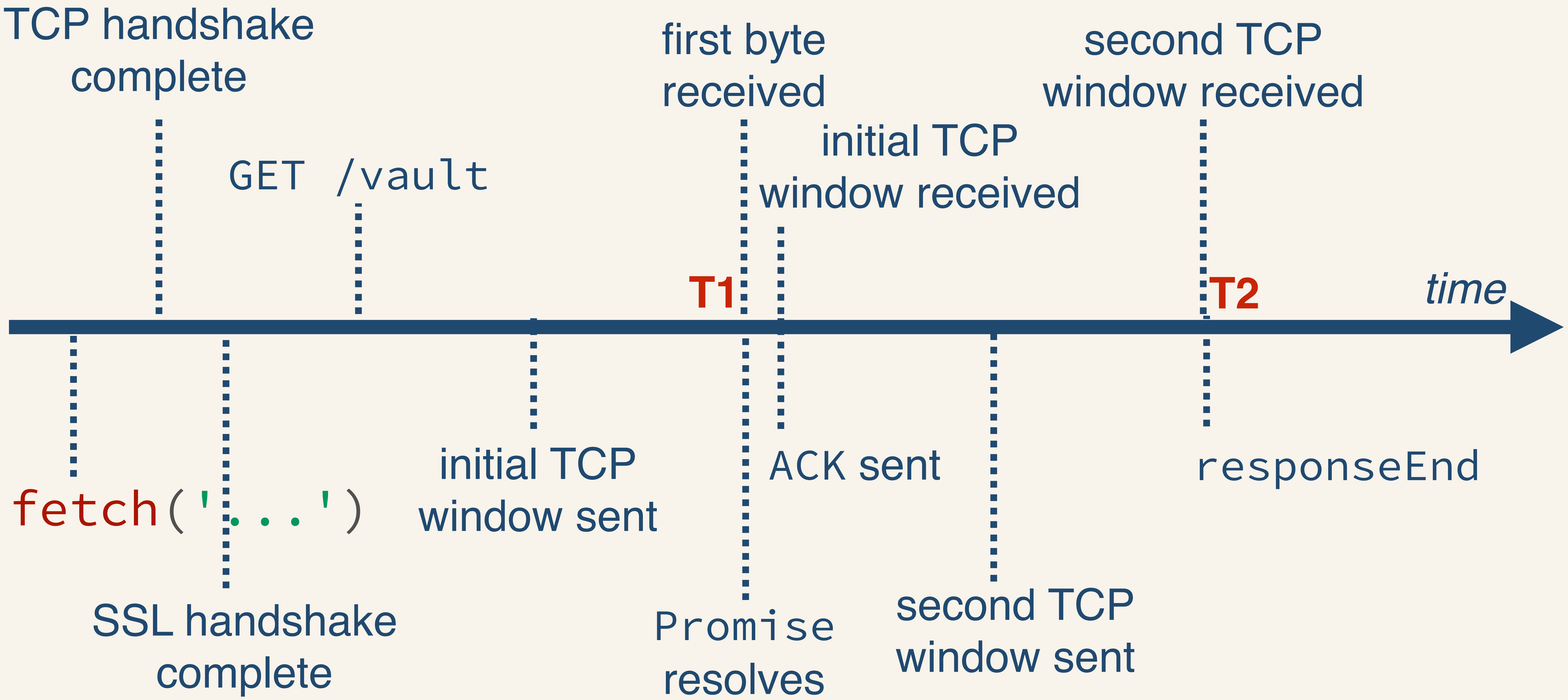
HEIST

- Step 1: find out if response fits in a single TCP window

Fetching small resource: T2 - T1 is very small



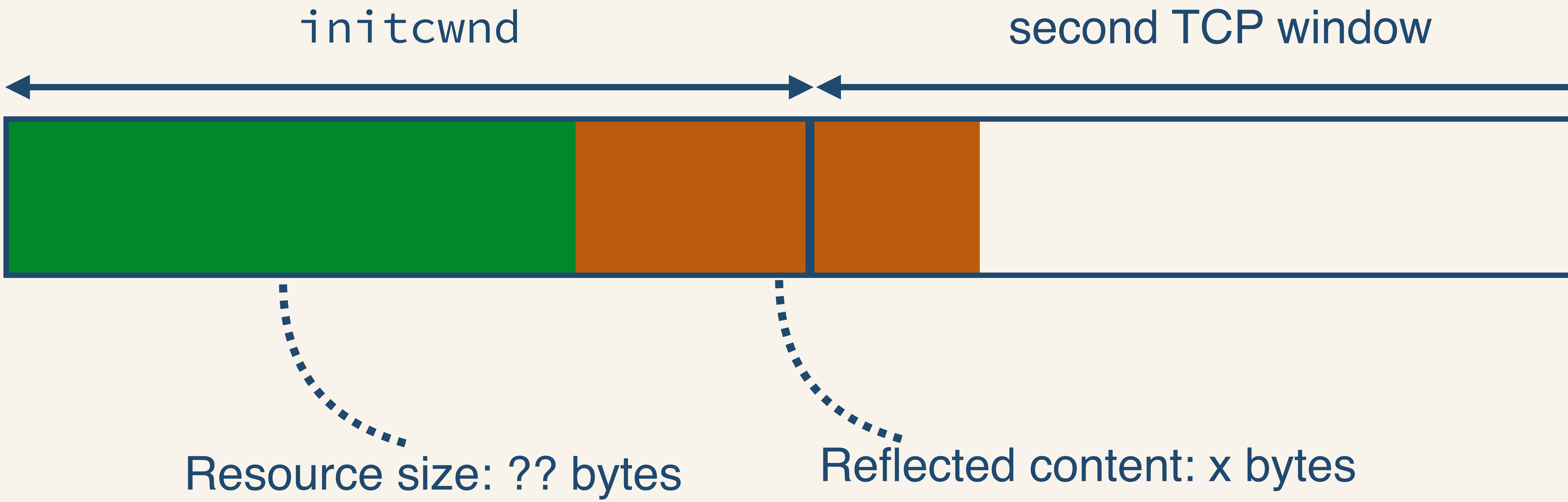
Fetching large resource: T2 - T1 is round-trip time



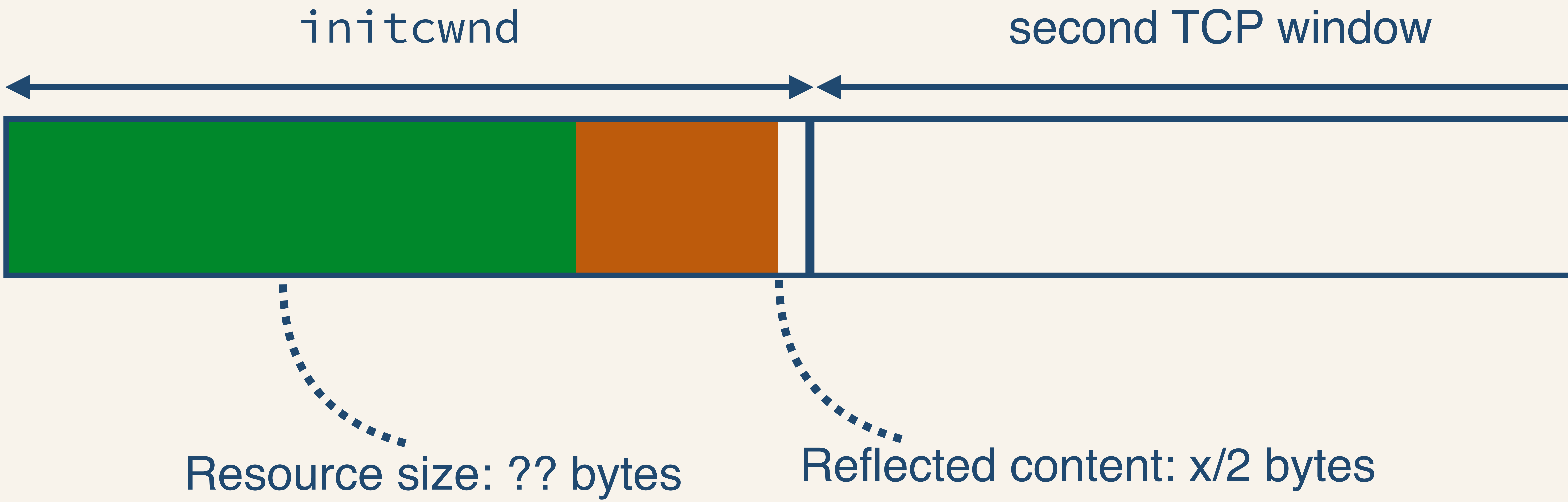
HEIST

- Step 1: find out if response fits in a single TCP window
- Step 2: discover exact response size

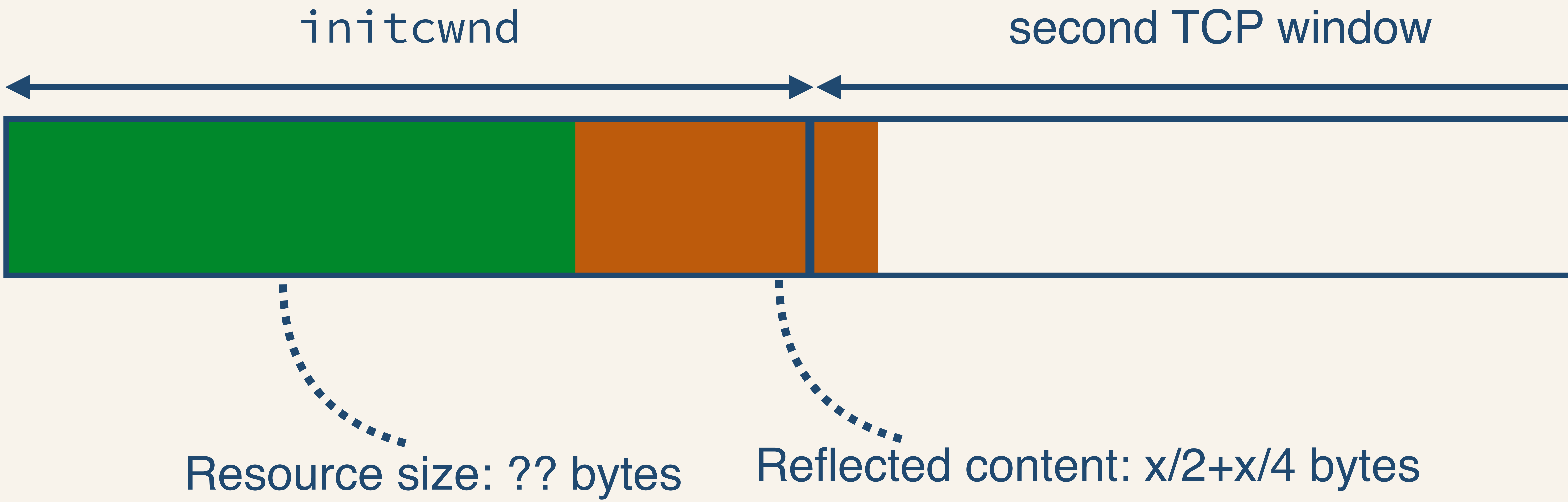
Discover Exact Response Size



Discover Exact Response Size



Discover Exact Response Size

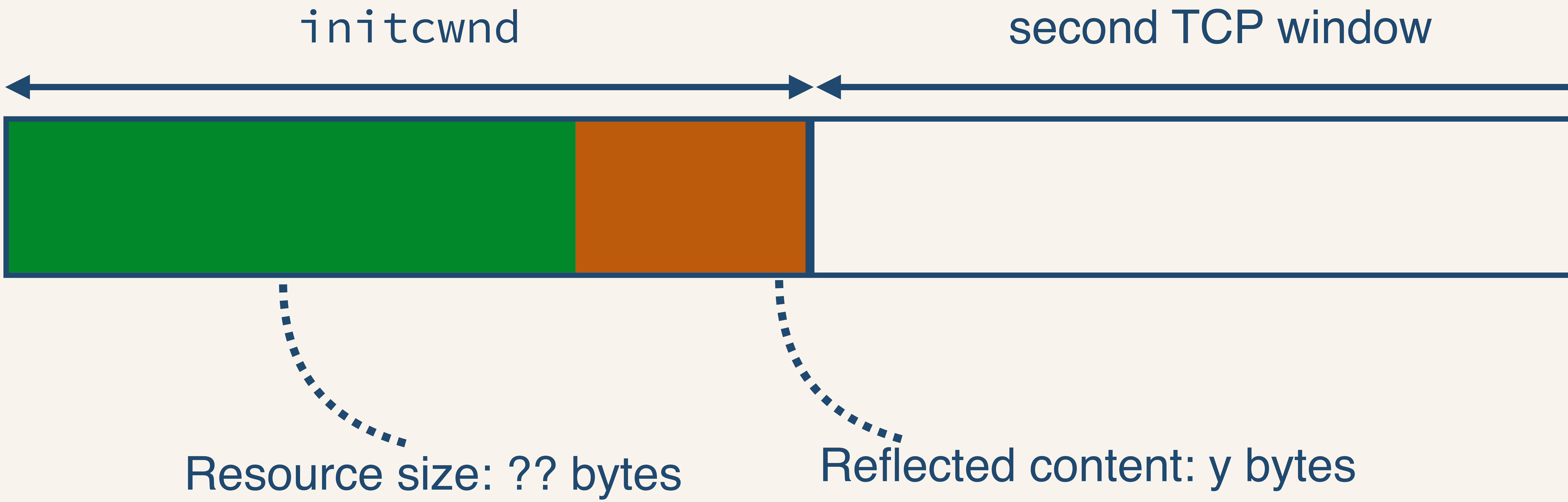


After *log(n)* checks, we find:

y bytes of reflected content = 1 TCP window

y+1 bytes of reflected content = 2 TCP windows

→ resource size = `initcwnd` - y bytes

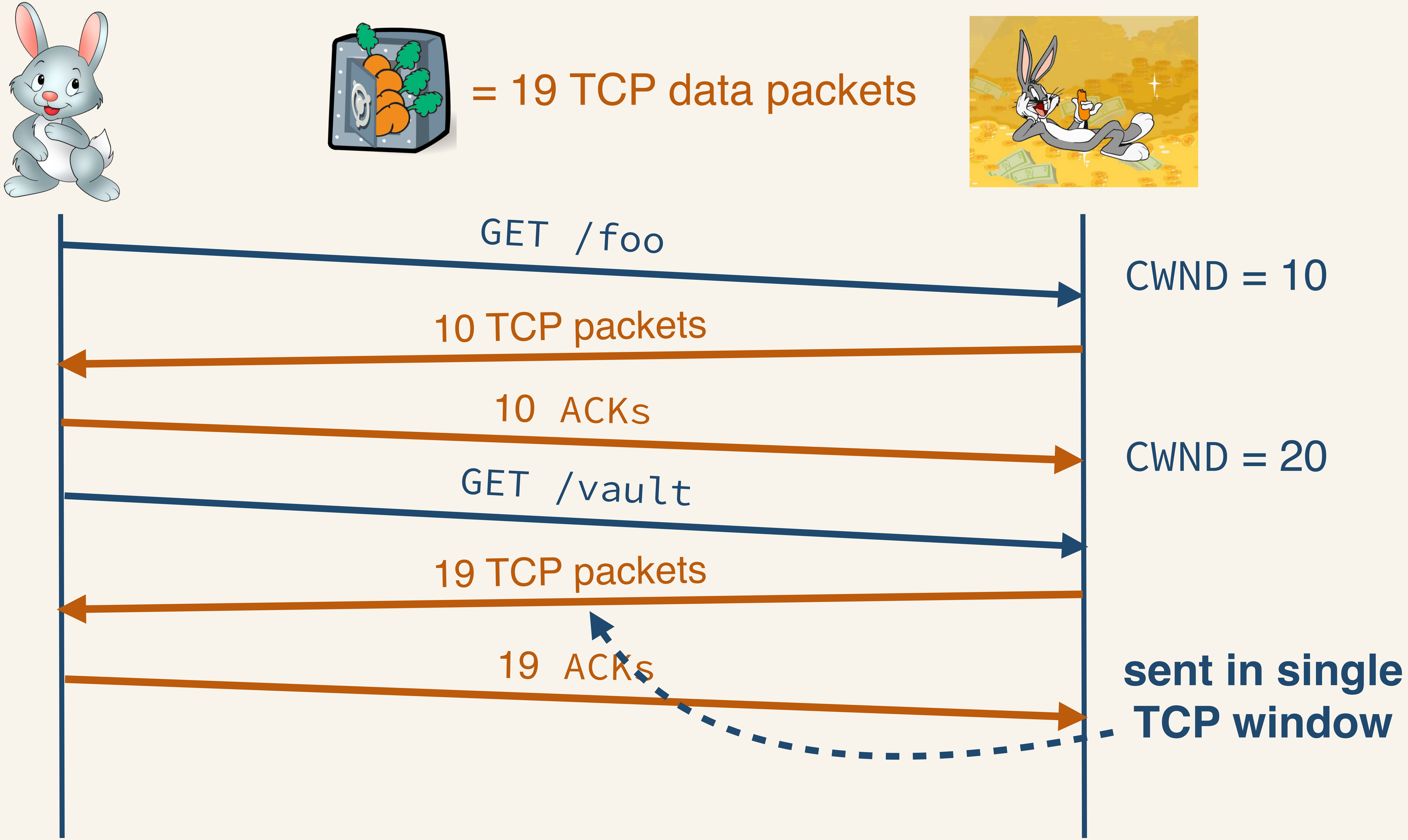


HEIST

- Step 1: find out if response fits in a single TCP window
- Step 2: discover exact response size
- Step 3: do the same for large responses (> `initcwnd`)

Determine size of large responses

- Large response = bigger than initial TCP window
- `initcwnd` is typically set to 10 TCP packets
  - ~14kB
- TCP windows grow as packets are acknowledged
- We can arbitrarily increase window size


= 19 TCP data packets
GET /foo
CWND = 10
10 TCP packets
10 ACKs
CWND = 20
GET /vault
19 TCP packets
19 ACKs
sent in single TCP window

HEIST

HEIST

- Step 1: find out if response fits in a single TCP window
- Step 2: discover exact response size
- Step 3: do the same for large responses (> `initcwnd`)
- Step 4: if available, leverage HTTP/2

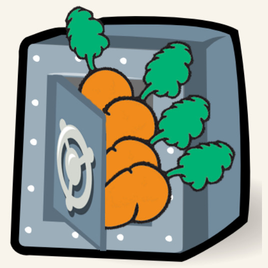
Leveraging HTTP/2

- HTTP/2 is the new HTTP version
  - Preserves the semantics of HTTP
- Main changes are on the network level
  - Only a single TCP connection is used for parallel requests

Leveraging HTTP/2

- Determine exact response size *without* reflected content in the same response
- Use (reflected) content in other responses on the same server
  - Note that BREACH still requires (a few bytes of) reflective content in the same resource


= 6 TCP packets
/reflect?x=... = 3 TCP packets
GET /reflect?x=...
CWND = 10
GET /vault
Promise resolves
9 TCP packets
9 ACKs
responseEnd
contains both /reflect and /vault



= 6 TCP packets

`/reflect?x=...` = 5 TCP packets


40 50 60 70

DEMO

Other targets

- Compression-based attacks
  - gzip compression is used by virtually every website
- Size-exposing attacks
  - Uncover victim's demographics from popular social networks
  - Reveal victim's health conditions from online health websites
  - ....
- Hard to find sites that are not vulnerable

Countermeasures

- Browser layer
  - Prevent side-channel leak *(infeasible)*
  - **Disable third-party cookies *(complete)***
- HTTP layer
  - Block illicit requests *(inadequate)*
  - Disable compression *(incomplete)*
- Network layer
  - Randomize TCP congestion window *(inadequate)*
  - Apply random padding *(inadequate)*

Conclusion

- Collection of techniques to discover network response size **in the browser**, for all authenticated cross-origin resources
- Side-channel originates from subtle interplay between multiple layers
- Allows for compression-based and size-exposing attacks
- HTTP/2 makes exploitation easier
- Many countermeasures, few that actually work

Questions?

Mathy Vanhoef
@vanhoefm
mathy.vanhoef@cs.kuleuven.be

Tom Van Goethem
@tomvangoethem
tom.vangoethem@cs.kuleuven.be