ROMA MMXVI | APPSEC EUROPE

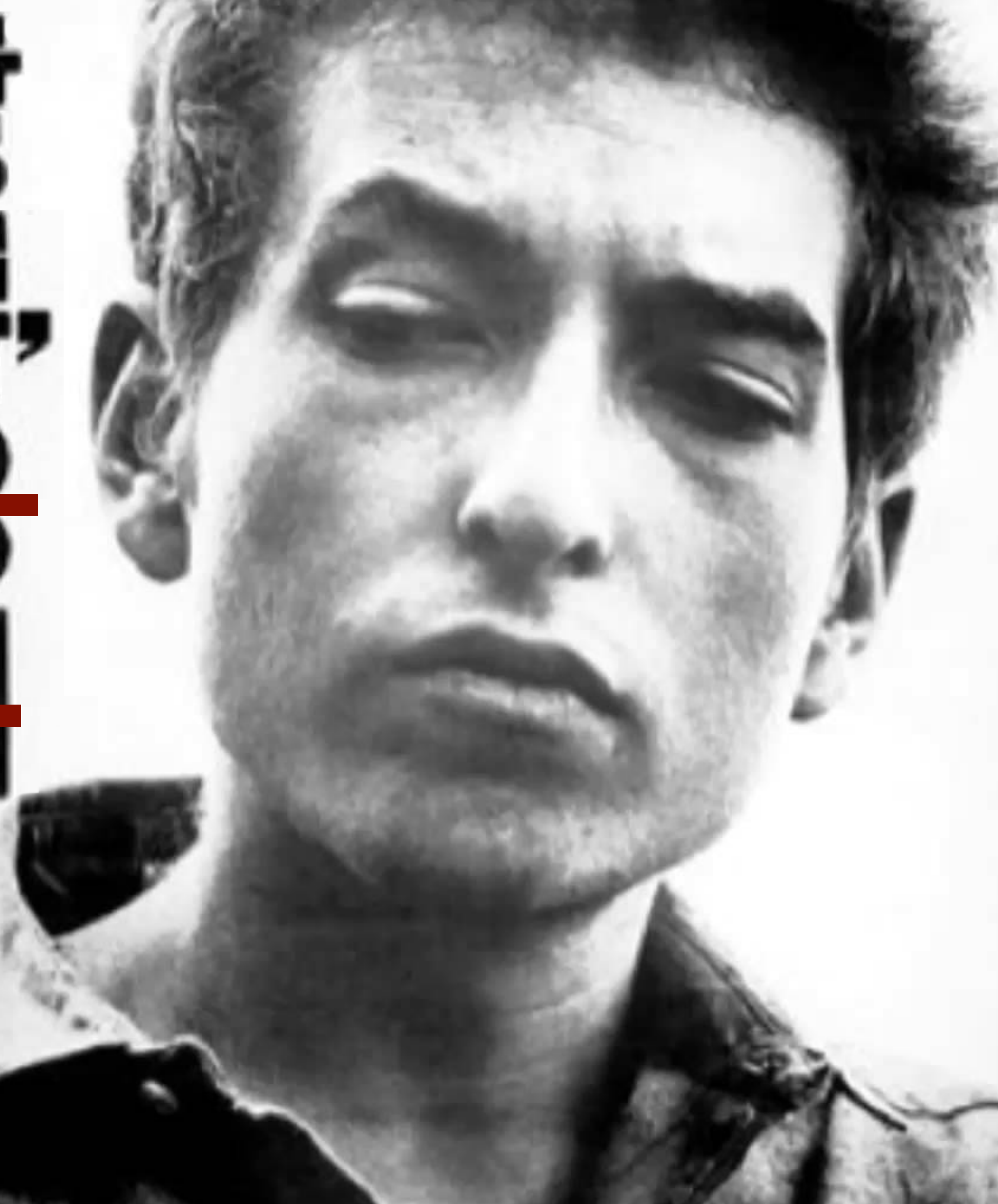
The Timing Attacks They Are A-Changin’

by Tom Van Goethem

THE TIMING ATTACKS THEY ARE A-CHANGIN'

~~BOB DYLAN~~

BY TOM VAN GOETHEM



About me

- Tom Van Goethem
- PhD Student at University of Leuven, Belgium
    - Research on security & privacy on the web
    - Current focus: side-channel attacks in the web/browser
- 🐦: @tomvangoethem

Timing Attacks

- Type of side-channel attack
  - Observe information from the environment to infer secret/ private information
- Known for decades, mainly in cryptography
  - Retrieve cryptographic keys
  - Exploit secure connections (e.g. Lucky13 attack)

Timing Attack Concept

1. Perform action
2. Measure time from `start` to `end`
3. Compare/analyse timing measurement
4. Profit!

Drunkenness Timing Attack

1. Perform action

   → Slap (drunk) person in face

2. Measure time from `start` to `end`

   → `start`: slap in face, `end`: slap back

3. Compare/analyse timing measurement

   → short not drunk, long: DRUNK!

4. Profit!

Drunkenness Timing Attack
The Experiment

Test subject:
Mathias Bynens (@mathias)

Timer: 00.000

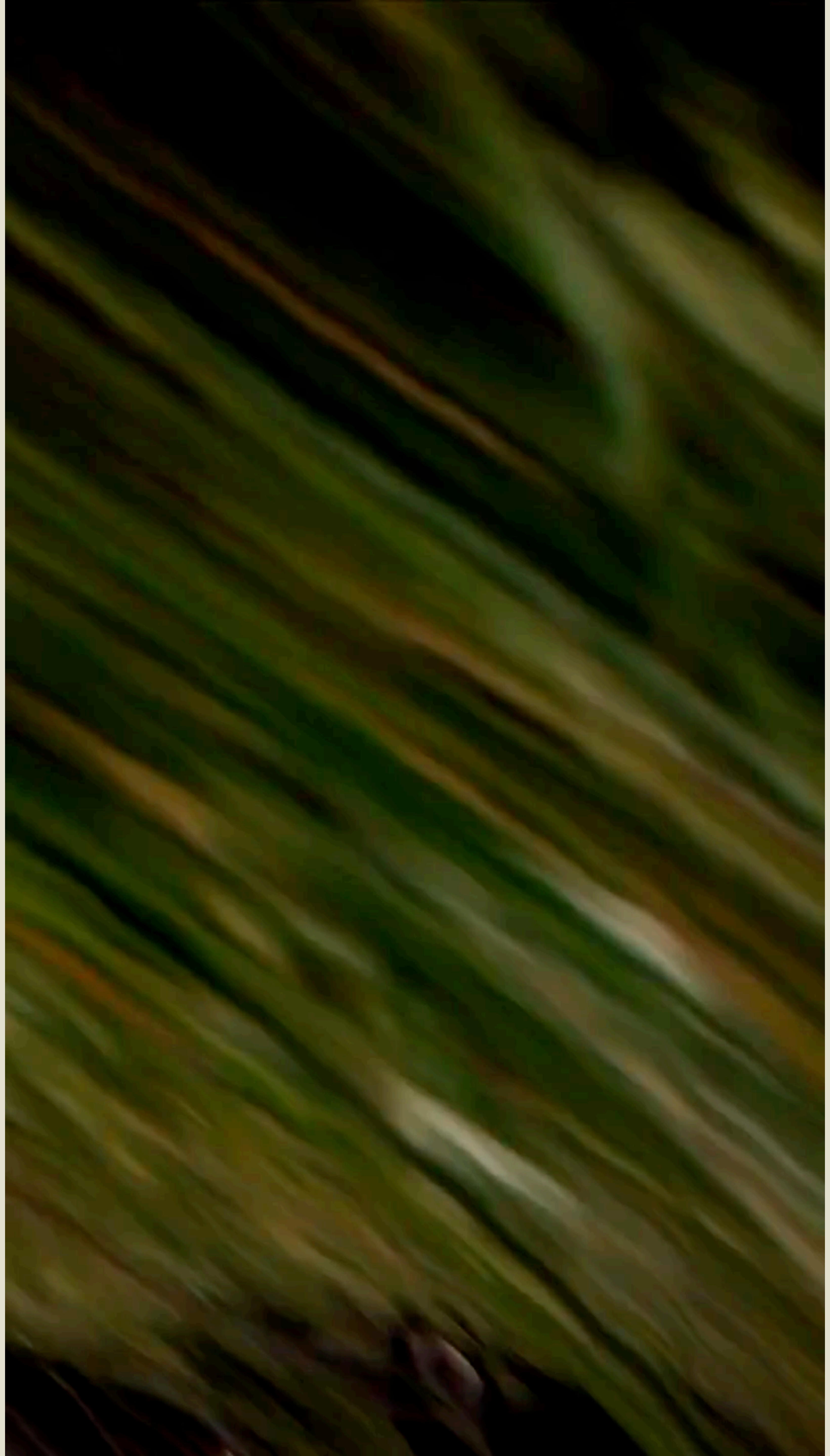
Is Mathias drunk?

Drunkenness Timing Attack
The Experiment

Test subject:
Mathias Bynens (@mathias)

Timer: 00.000

Is Mathias drunk?



Drunkenness Timing Attack
The Experiment

Test subject:
  Mathias Bynens (@mathias)

Timer: 03.209

Is Mathias drunk?

Probably not...

Drunkenness Timing Attack
The Experiment Part 2

Drunkenness Timing Attack
The Experiment Part 2

Test subject:
Mathias Bynens (@mathias)

Timer: 00.000

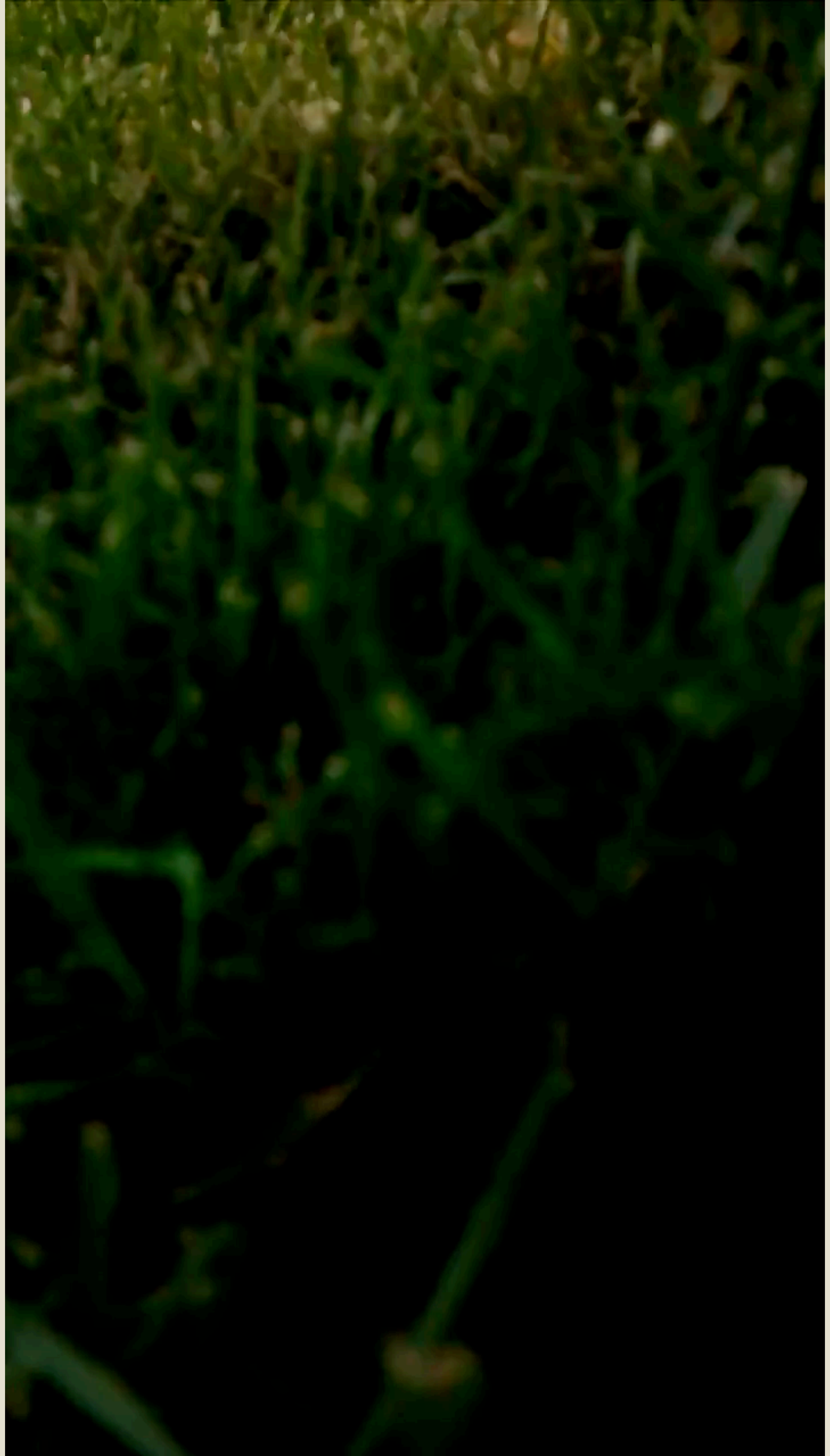
Is Mathias drunk?

Drunkenness Timing Attack
The Experiment Part 2

Test subject:
Mathias Bynens (@mathias)

Timer: 00.000

Is Mathias drunk?



Drunkenness Timing Attack
The Experiment Part 2

Test subject:
Mathias Bynens (@mathias)

Timer: 08.497

Is Mathias drunk?

YES!

(08.497 > 03.209)

Timing Attacks

- Drunkenness Timing Attacks
  - Response time ~ drunkenness
- Cryptographic key (or message) recovery
  - Execution time ~ bit is 1 or bit is 0
- Lucky 13
  - TLS error message time ~ padding information/plaintext
- Web-based Timing Attacks
  - ....

Web-based Timing Attacks

- Known since 2000!
- Pixel Perfect (by Paul Stone, 2013)
  - GPU rendering time ~ pixel color
- JS comparison timing attack (by @sirdarckcat, 2013)
  - JS execution time ~ attacker input + secret data
- Direct timing attack
- Cross-site timing attack

(discovered by Bortz & Boneh, 2007)

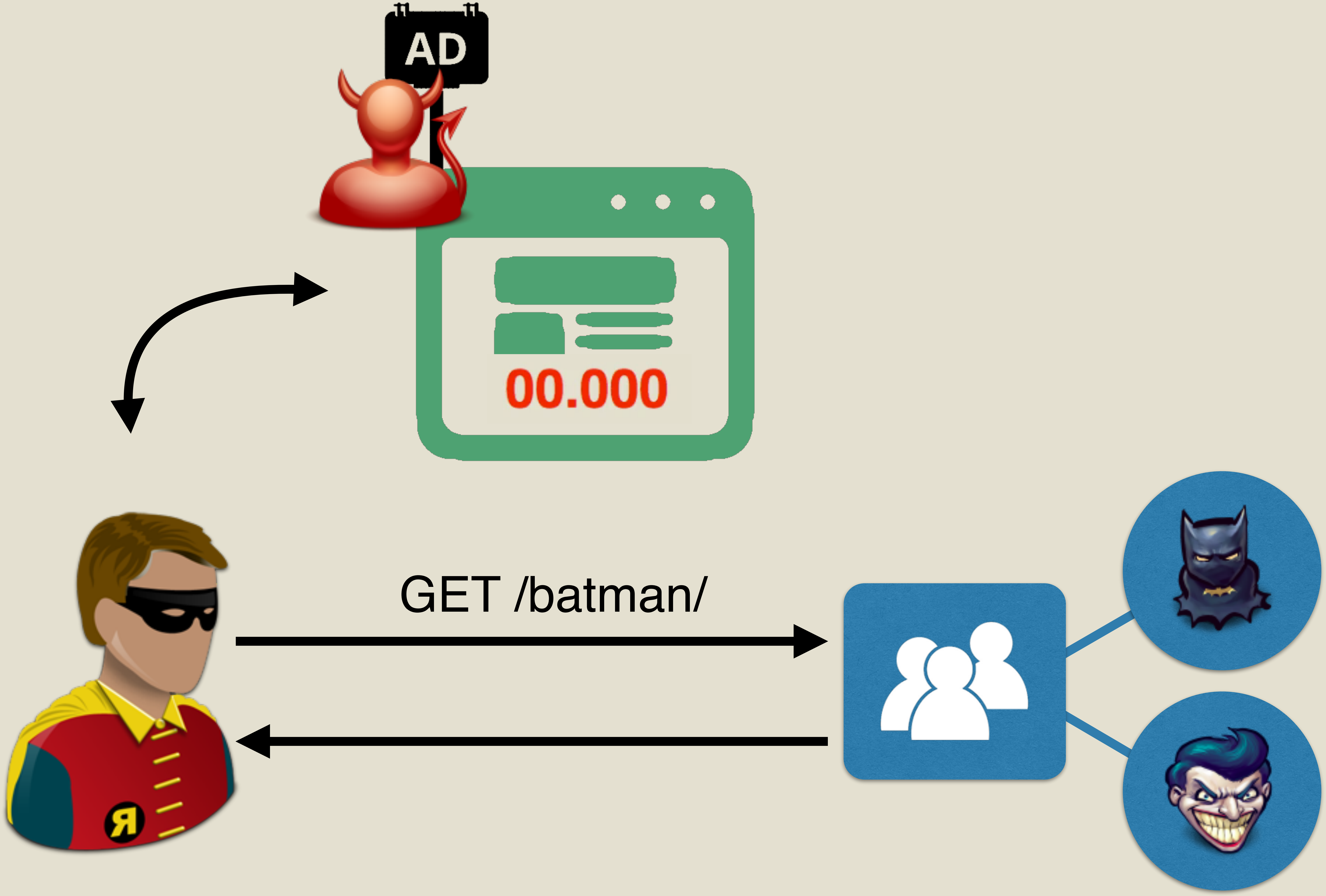
Cross-site Timing Attacks

- Request to the server is sent by the victim
- Cookies are attached to the request
  - Response is specific to the state of the user
  - e.g. logged in → large response / not logged in → small response
- Attacker measures response time with JS

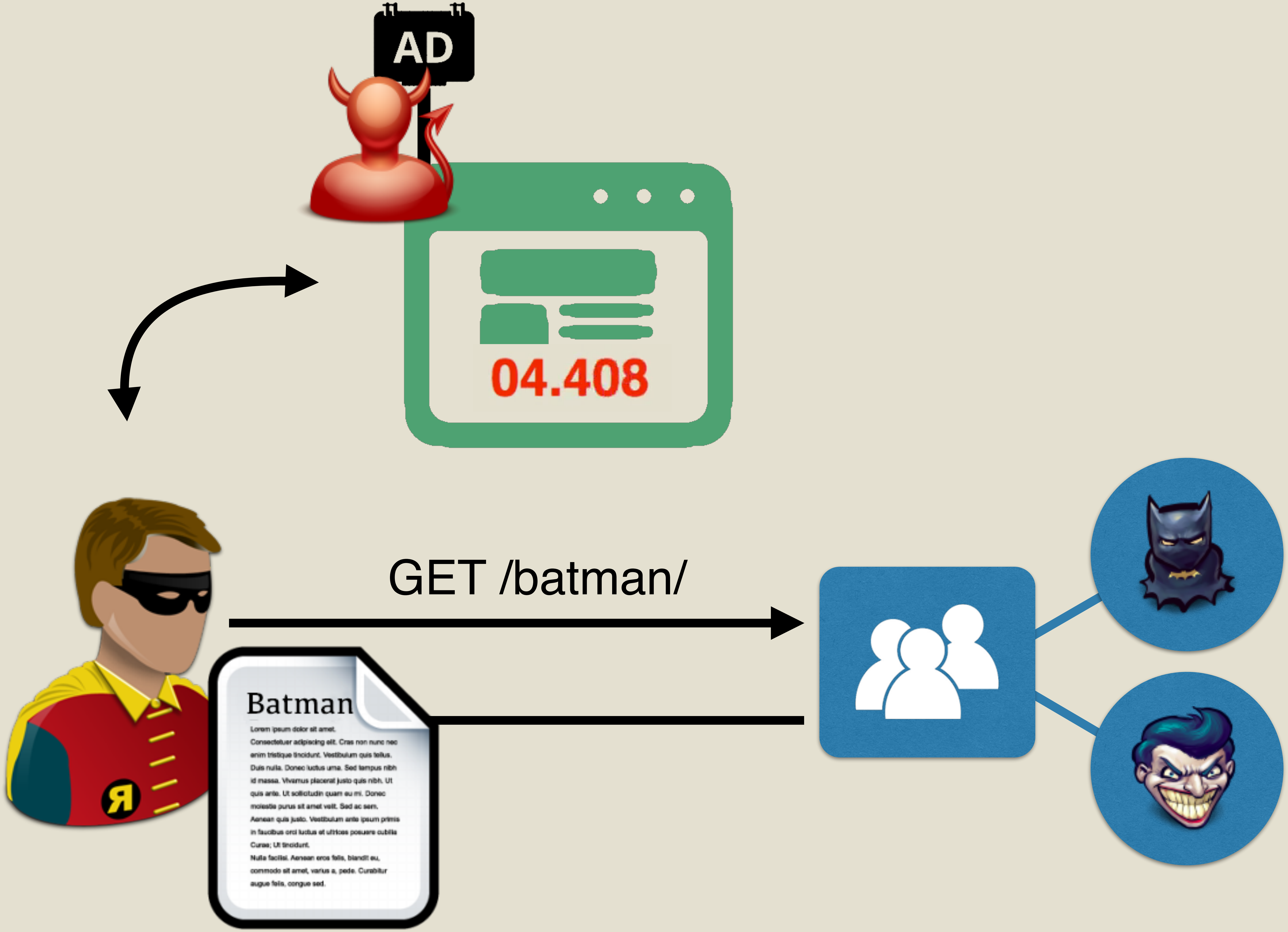
  -
    ```
    img.onerror = function() { stopTimer(); }
    ```
- Violates Same-Origin Policy

Same-Origin Policy

- Main security principle of the web!
- attacker.com can not read anything from example.com
- The length of the resources is also secret
  - With only length information, attacker can infer a lot of private information (see demo later)



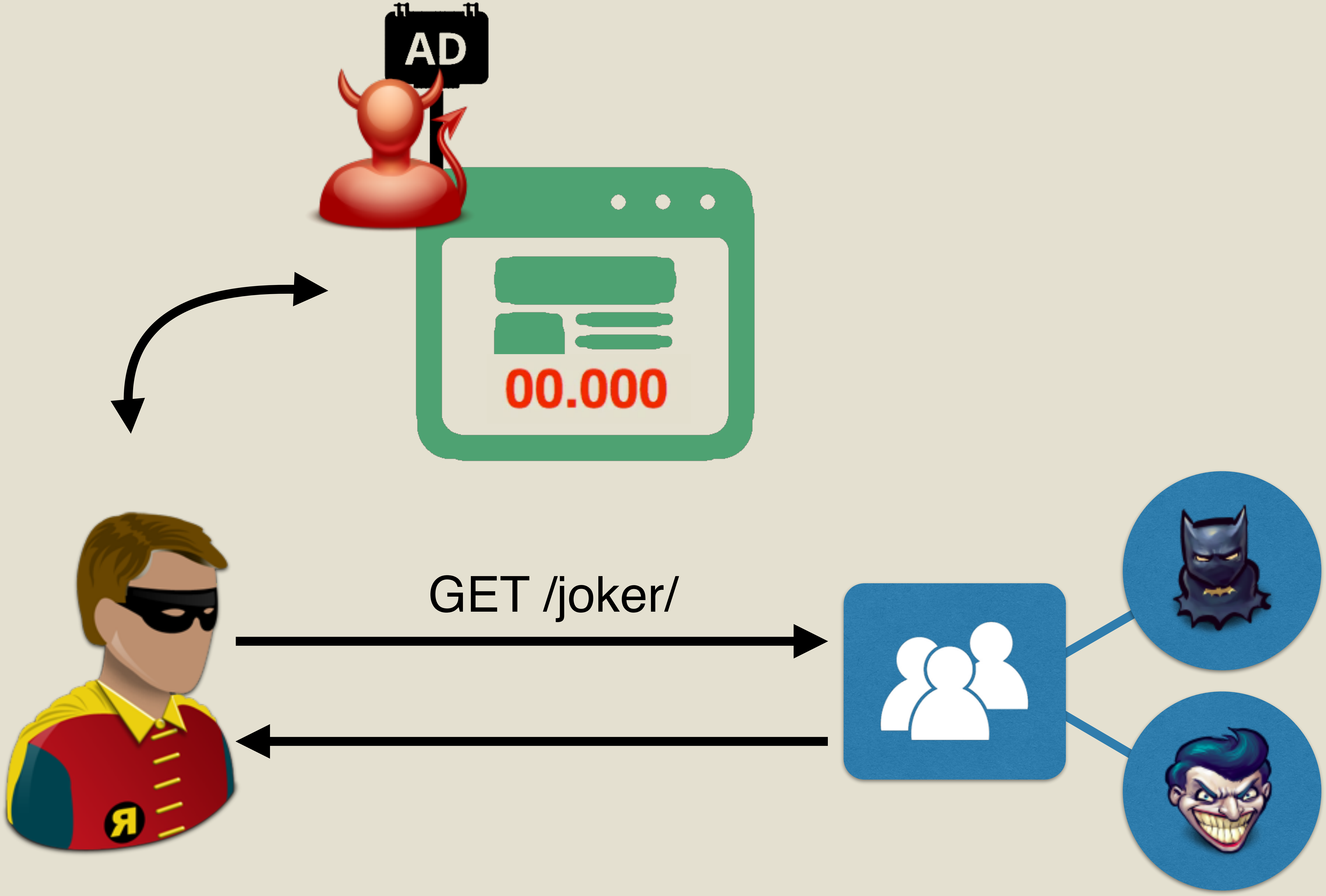
GET /batman/



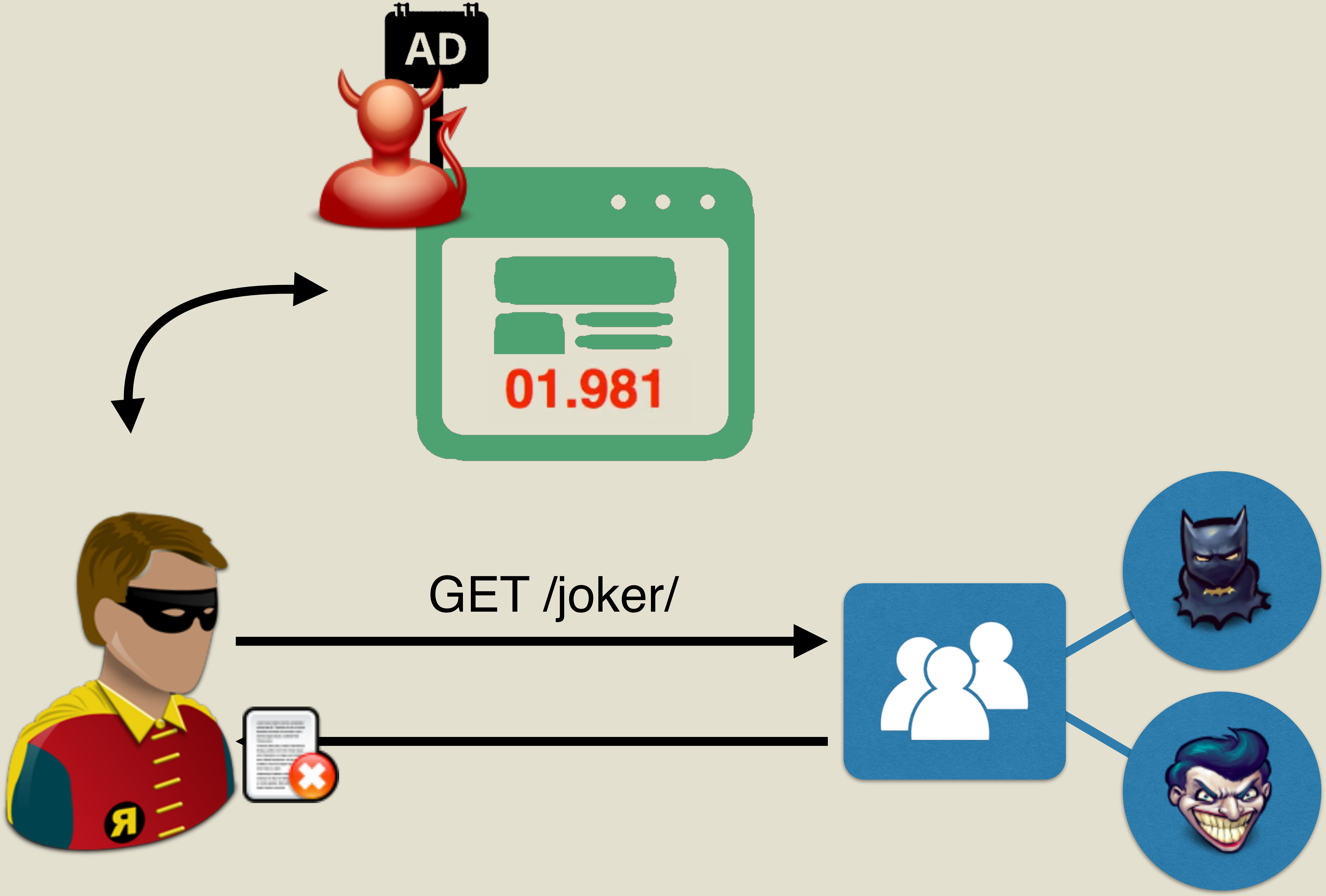
GET /batman/

Batman

Lorem ipsum dolor sit amet.
Consectetuer adipiscing elit. Cras non nunc nec enim tristique tincidunt. Vestibulum quis tellus. Duis nulla. Donec luctus urna. Sed tempus nibh id massa. Vivamus placerat justo quis nibh. Ut quis ante. Ut sollicitudin quam eu mi. Donec molestie purus sit amet velit. Sed ac sem. Aenean quis justo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tincidunt.
Nulla facilisi. Aenean eros felis, blandit eu, commodo sit amet, varius a, pede. Curabitur augue felis, congue sed.

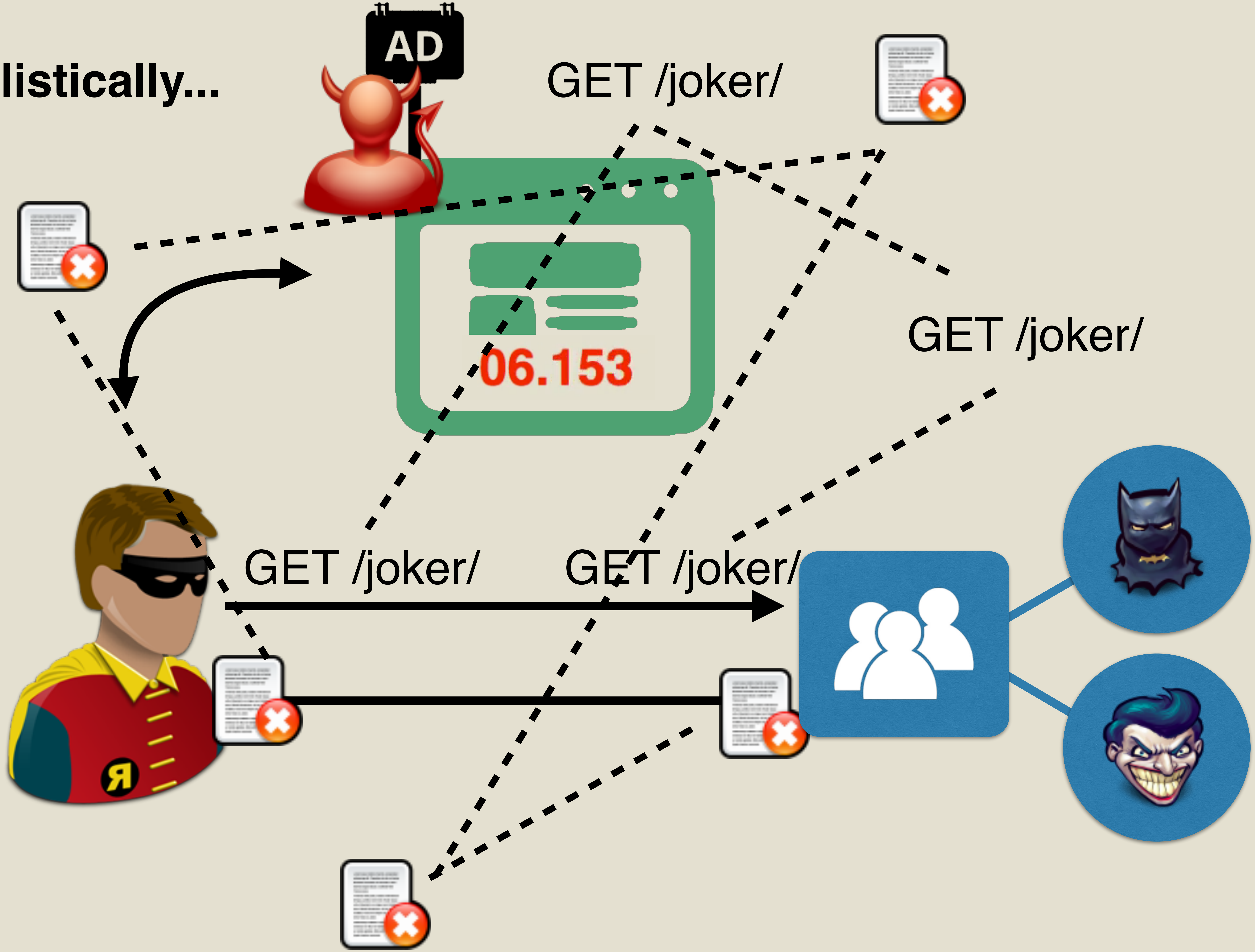


GET /joker/



GET /joker/

More realistically...

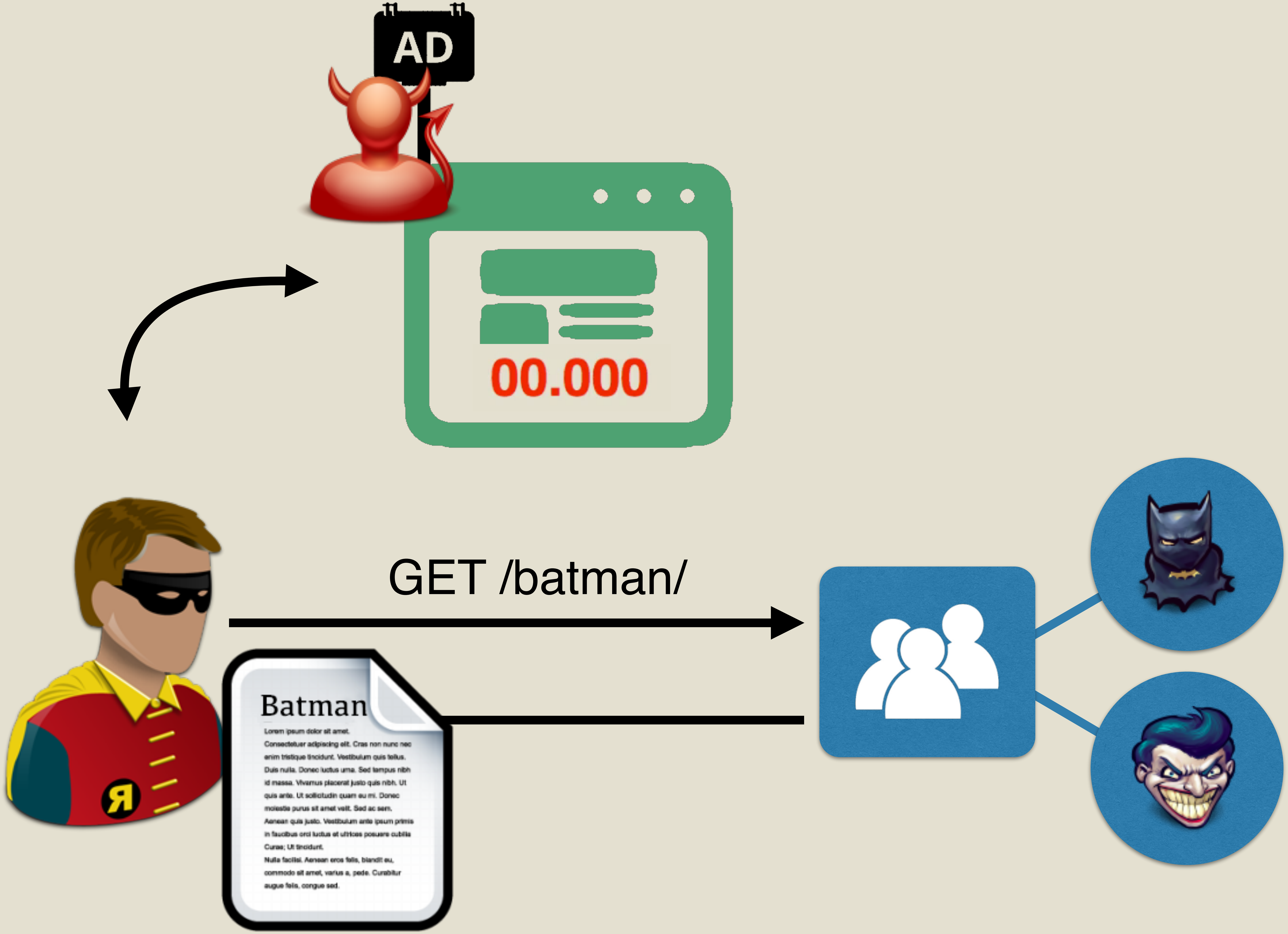


Classic Cross-site Timing Attacks

- Classic timing attacks have several limitations
  - Network irregularities
  - gzip compression
  - Round-trip for each measurement
  - Rate-limiting

Browser-based Timing Attacks

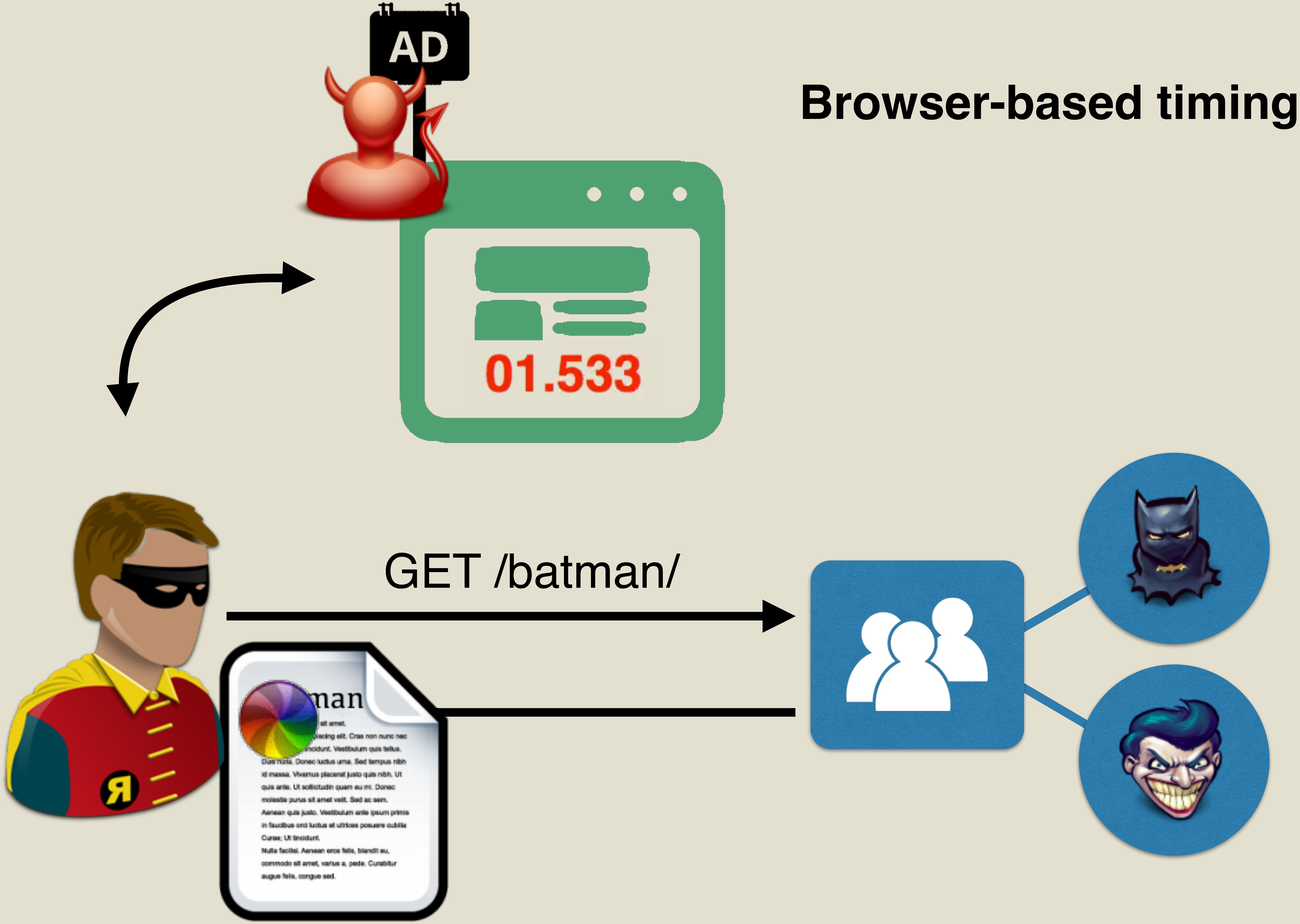
- Timing attacks in browsers overcome these limitations
  - Timing measurement starts *after* resource is downloaded
  - Measurements are more accurate
  - For some attacks: resource is only downloaded once
  - Obtain multiple measurements in short interval



Batman

Lorem ipsum dolor sit amet.
Consectetuer adipiscing elit. Cras non nunc nec enim tristique tincidunt. Vestibulum quis tellus. Duis nulla. Donec luctus urna. Sed tempus nibh id massa. Vivamus placerat justo quis nibh. Ut quis ante. Ut sollicitudin quam eu mi. Donec molestie purus sit amet velit. Sed ac sem. Aenean quis justo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tincidunt.
Nulla facilisi. Aenean eros felis, blandit eu, commodo sit amet, varius a, pede. Curabitur augue felis, congue sed.

Browser-based timing attacks


AD
01.533
GET /batman/
man
sit amet.
piscing elit. Cras non nunc nec
ncidunt. Vestibulum quis tellus.
Duis nulla. Donec luctus urna. Sed tempus nibh
id massa. Vivamus placerat justo quis nibh. Ut
quis ante. Ut sollicitudin quam eu mi. Donec
molestie purus sit amet velit. Sed ac sem.
Aenean quis justo. Vestibulum ante ipsum primis
in faucibus orci luctus et ultrices posuere cubilia
Curae; Ut tincidunt.
Nulla facilisi. Aenean eros felis, blandit eu,
commodo sit amet, varius a, pede. Curabitur
augue felis, congue sed.

Browser-based Timing Attacks

- Side-channels allow measuring time to process resource
  - Parse as specific format (~ CPU processing time)
  - Retrieve from cache (~ disk read time)
  - Store in cache (~ disk write time)

Video Parsing Attack

```javascript
let video = document.createElement('video');

// suspend => download complete
video.addEventListener('suspend',function(){
    start = window.performance.now();
});

// error => parsing complete
video.addEventListener('error',function(){
    end = window.performance.now();
});

video.src = 'https://example.org/resource';
```

Video Parsing Attack

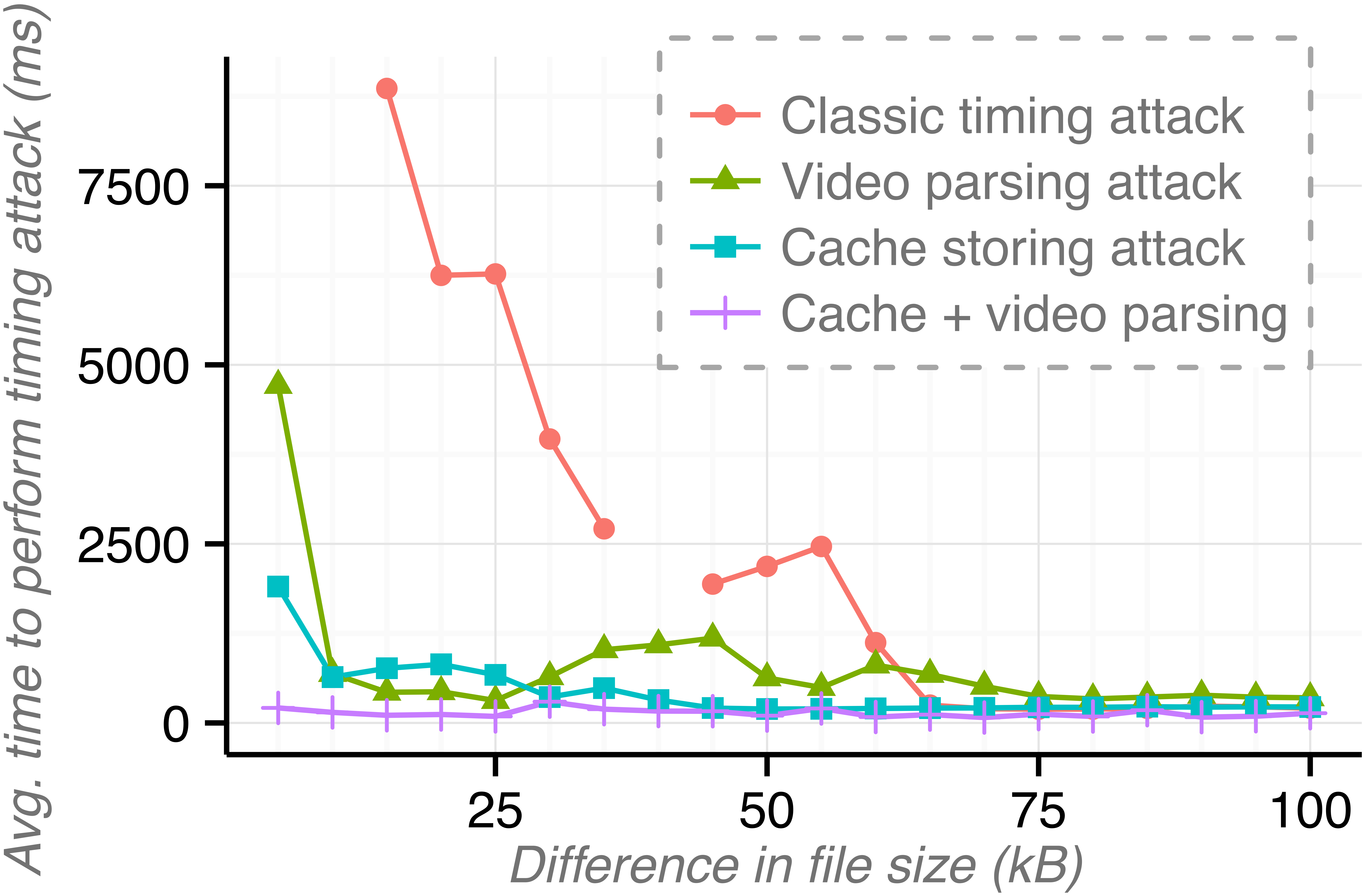
```
let video = document.createElement('video');

// suspend => download complete
video.addEventListener('suspend',function(){
    start = window.performance.now();
});

// error => parsing complete
video.addEventListener('error',function(){
    end = window.performance.now();
});

video.src = 'https://example.org/resource';
```

appcache.manifest

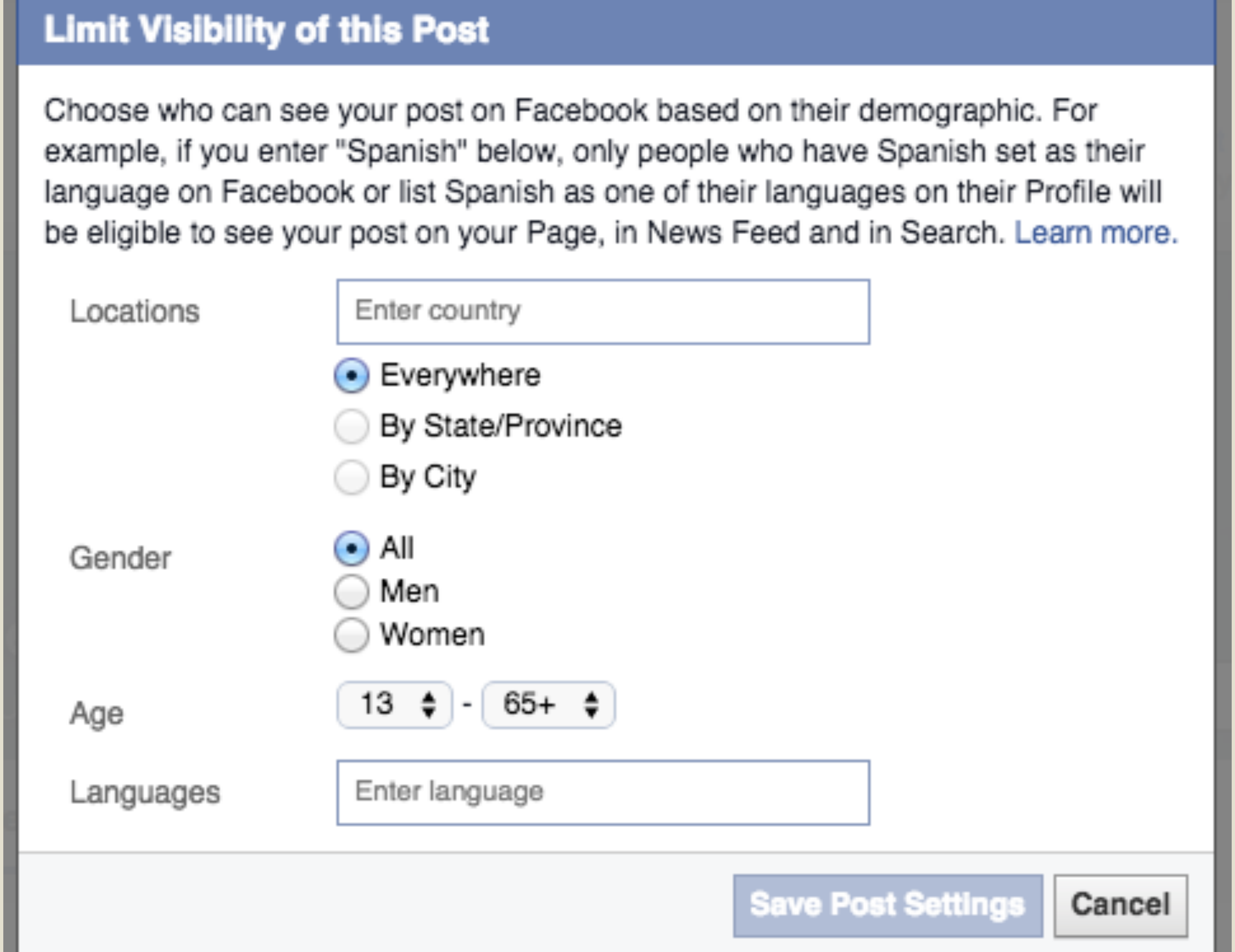
```
CACHE MANIFEST
CACHE:
https://example.org/resource
NETWORK:
*
```

Cache Storing Attack

```
let url = 'https://example.org/resource';
let opts = {credentials: "include", mode: "no-cors"};
let request = new Request(url, opts);
let bogusReq = new Request('/bogus');
fetch(request).then(function(resp) {
    // Resource download complete
    start = window.performance.now();
    return cache.put(foo, resp.clone())
}).then(function() {
    // Resource stored in cache
    end = window.performance.now();
});
```


Classic timing attack
Video parsing attack
Cache storing attack
Cache + video parsing
Avg. time to perform timing attack (ms)
7500
5000
2500
0
25
50
75
100
Difference in file size (kB)

Demo



Limit Visibility of this Post

Choose who can see your post on Facebook based on their demographic. For example, if you enter "Spanish" below, only people who have Spanish set as their language on Facebook or list Spanish as one of their languages on their Profile will be eligible to see your post on your Page, in News Feed and in Search. Learn more.

Locations: Enter country

● Everywhere
○ By State/Province
○ By City

Gender:
● All
○ Men
○ Women

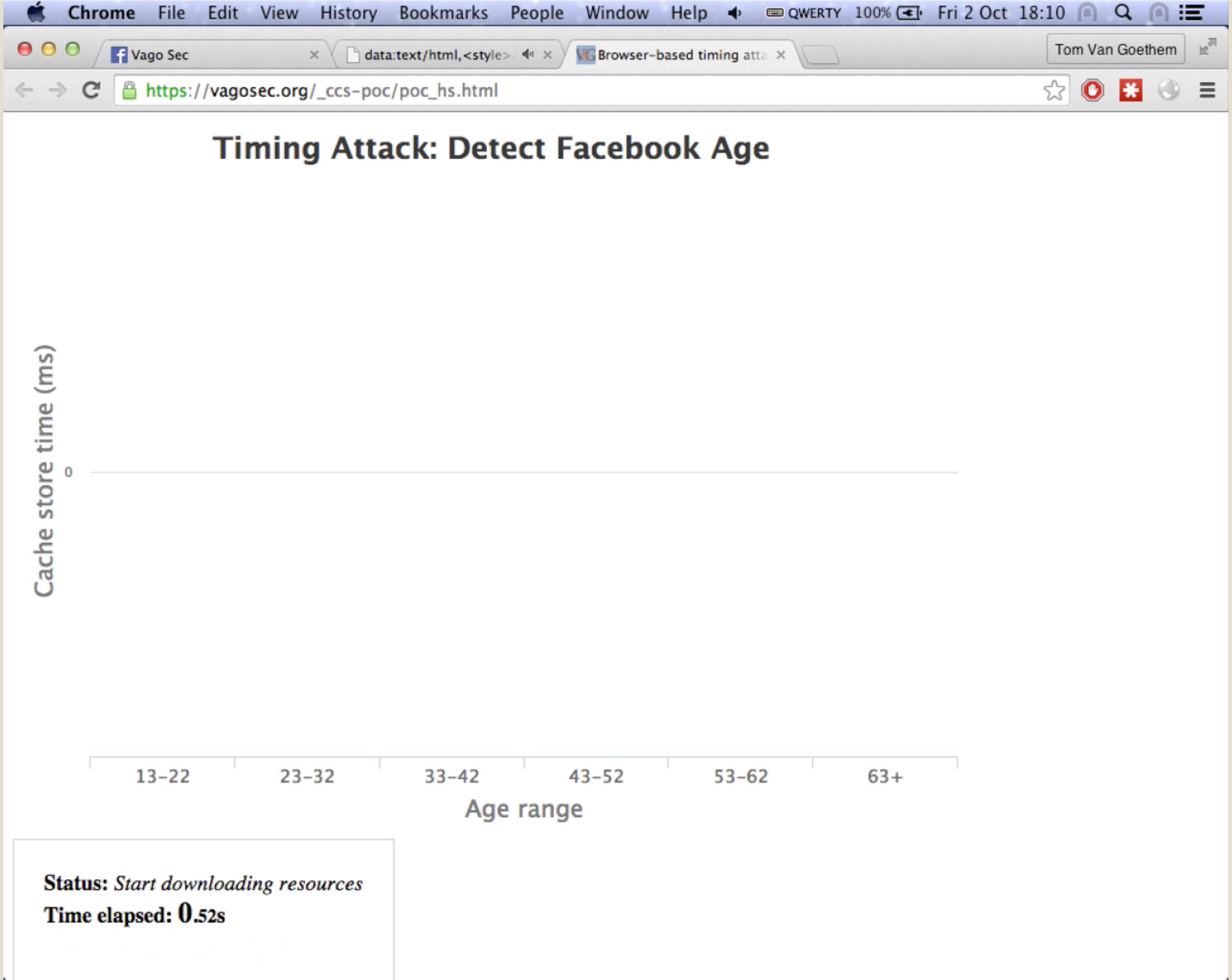
Age: 13 - 65+

Languages: Enter language

Age-discovery Attack

1. Create Facebook posts, each targeted to users of a specific age
2. Discover age-range of the user
   - Fetch corresponding resources
   - Obtain timing measurements
   - Determine age-range according to the value of timing measurements
3. Discover exact age of the user
   - Repeat (2) but for posts targeted to specific age

Step 1

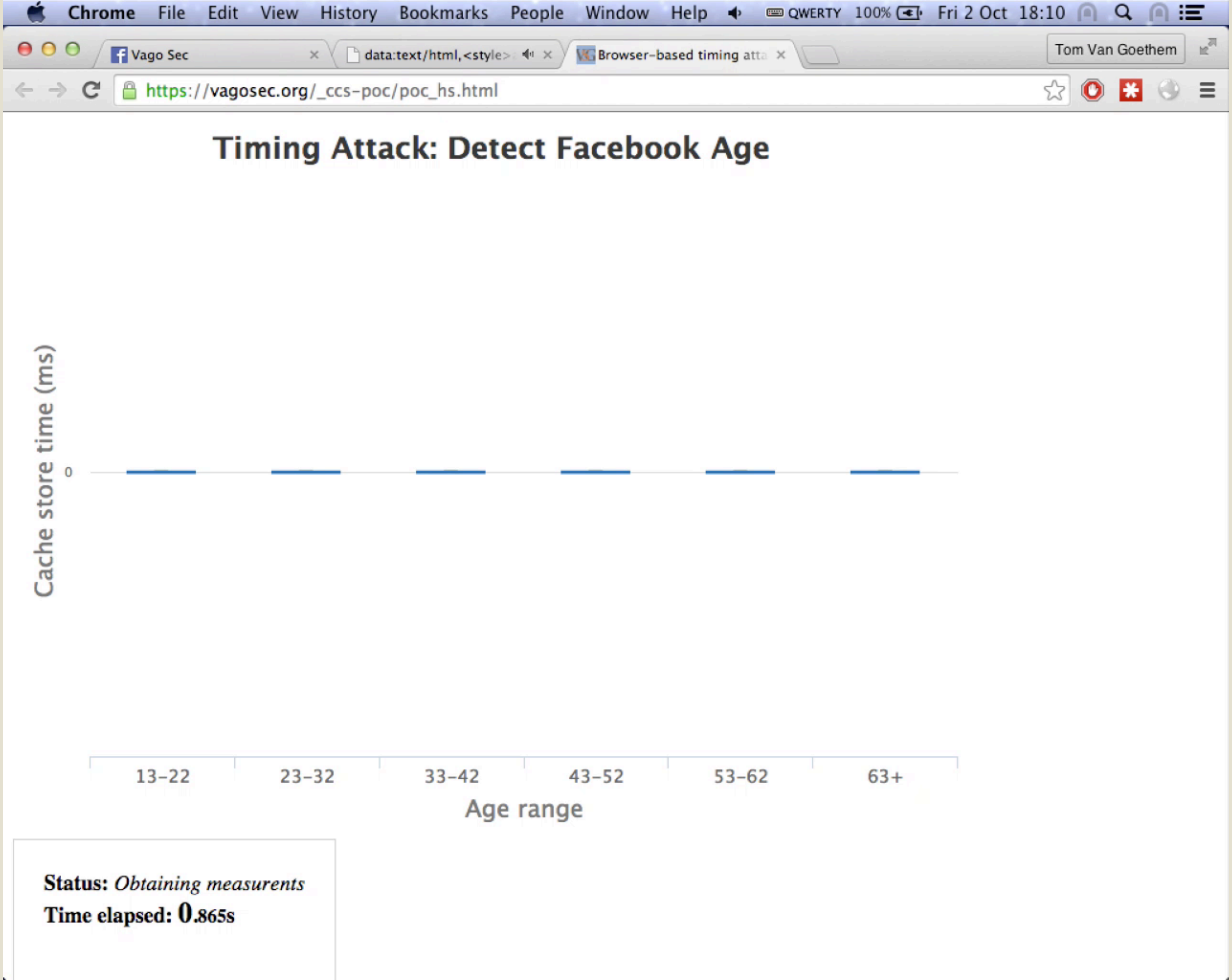


Chrome File Edit View History Bookmarks People Window Help
QWERTY 100% Fri 2 Oct 18:10
Vago Sec
data:text/html,<style>
Browser-based timing atta
Tom Van Goethem
https://vagosec.org/_ccs-poc/poc_hs.html
Cache store time (ms)
0
13-22
23-32
33-42
43-52
53-62
63+
Age range

Timing Attack: Detect Facebook Age

Status: *Start downloading resources*

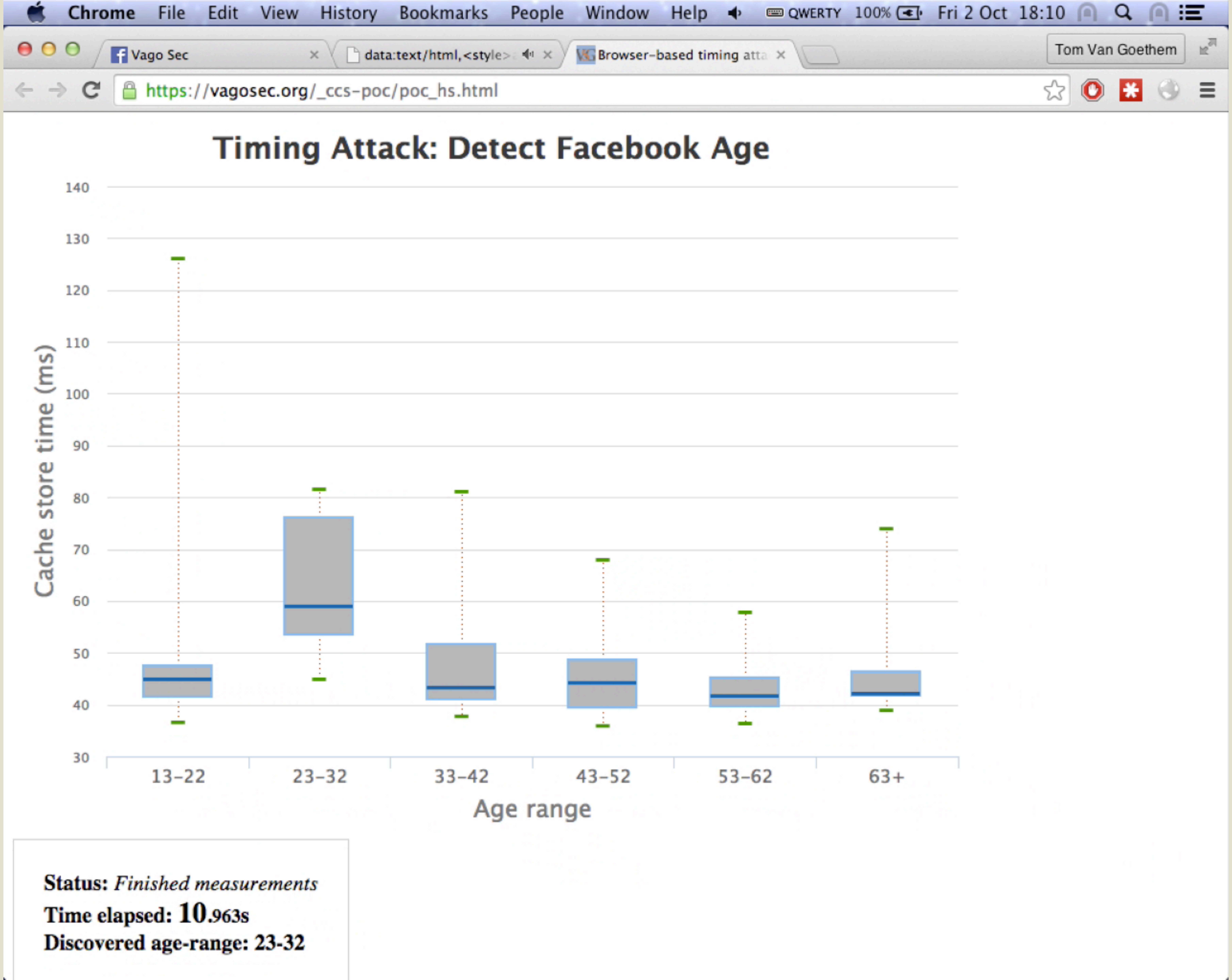
Time elapsed: **0**.52s



Timing Attack: Detect Facebook Age

Status: *Obtaining measurents*

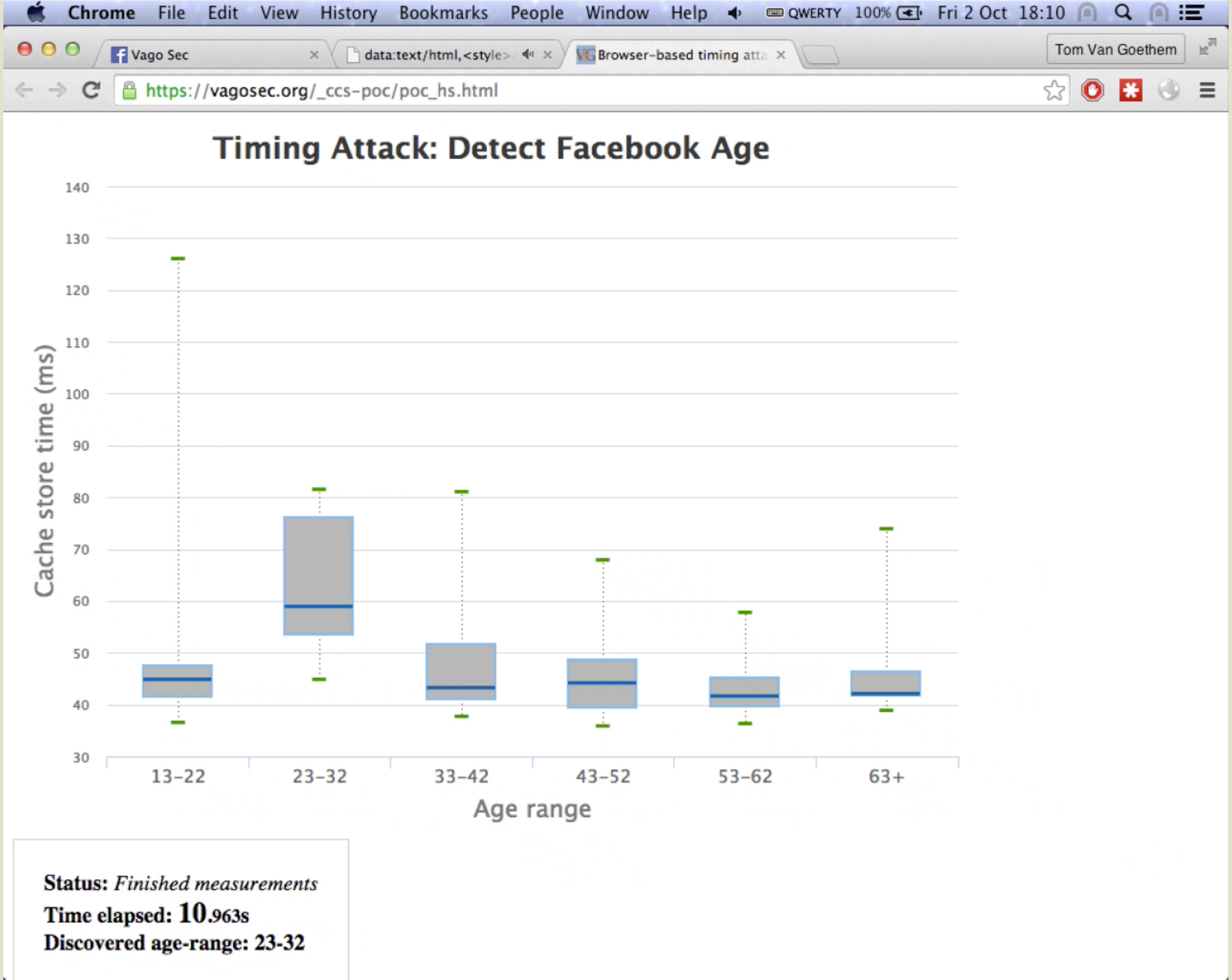
Time elapsed: 0.865s


Chrome File Edit View History Bookmarks People Window Help
QWERTY 100% Fri 2 Oct 18:10
Vago Sec
data:text/html,<style>
Browser-based timing atta
Tom Van Goethem
https://vagosec.org/_ccs-poc/poc_hs.html
Timing Attack: Detect Facebook Age
Cache store time (ms)
140
130
120
110
100
90
80
70
60
50
40
30
13-22
23-32
33-42
43-52
53-62
63+
Age range

Status: *Finished measurements*

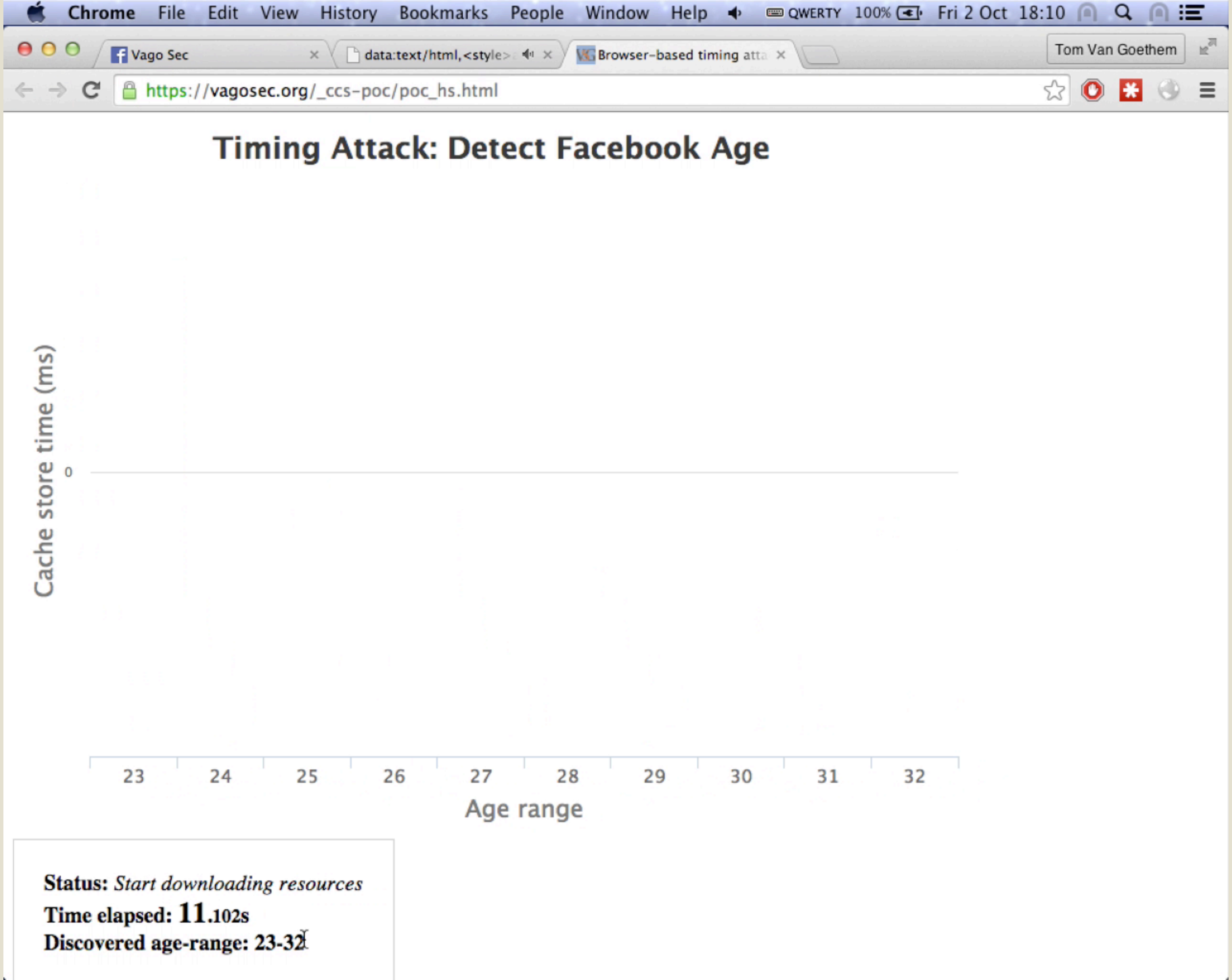
Time elapsed: 10.963s

Discovered age-range: 23-32



Status: *Finished measurements*
Time elapsed: 10.963s
Discovered age-range: 23-32

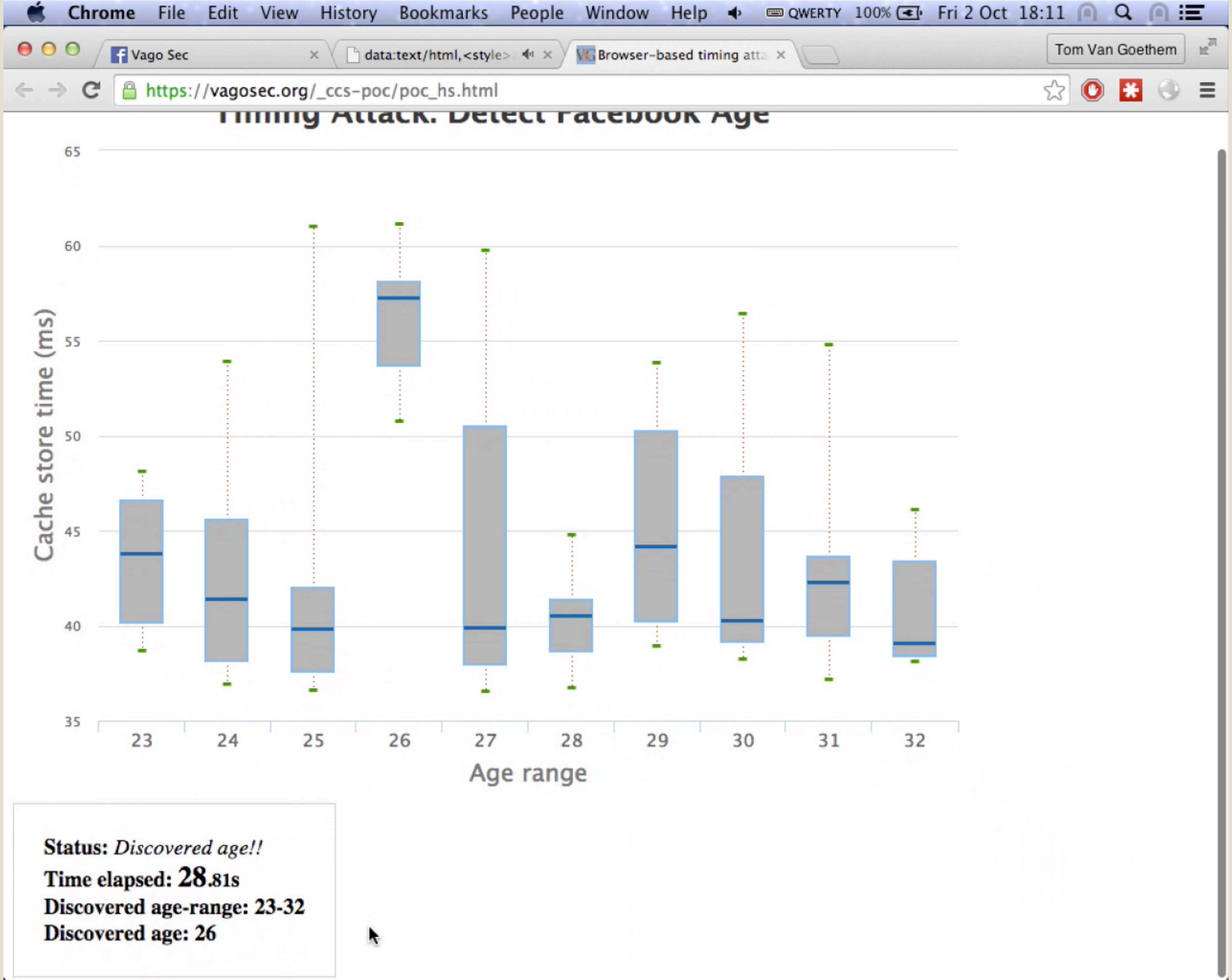
Step 2



Status: *Start downloading resources*

Time elapsed: 11.102s

Discovered age-range: 23-32



Status: *Discovered age!!*

Time elapsed: 28.81s

Discovered age-range: 23-32

Discovered age: 26

Moar Attacks

VULNERABILITIES
VULNERABILITIES EVERYWHERE
Spongebob Squarepants vector trace by kssael. ©Nickelodeon
memegenerator.net

- Facebook: demographics
- LinkedIn: connections, ...
- Twitter: following, identity, ...
- Google: search history
- Amazon: shopping history
- Gmail: inbox search
- ...

Mitigation

- Client-side
  - Non-trivial, side-channels are there by design
  - Reducing timer resolution does not work: ms resolution sufficient
  - Disable 3rd-party cookies?
- Server-side
  - Cause is related to CSRF
  - CSRF token on GET? `SameSite` cookies?

SECURITY
SECURITY

Conclusion

- Novel browser-based timing attacks
  - Timing measurement starts *after* resource is downloaded
  - Faster & more robust than classic timing attacks
  - Side-channels exist by design
- Many popular web applications are vulnerable
- Mitigation is challenging, and definitely not there yet

Questions?

tom.vangoethem@cs.kuleuven.be

: @tomvangoethem

AppCache attack

- Specify which resources need to be cached
  - This include cross-origin resources
- After caching, measure time required to load resources
  - Time between setting `src` and `onerror` on images
- Often too fast for accurate measurement
  - → load resources multiple times
- Resources only need to be downloaded once

Mitigation

(current status)

“We ended up discussing this at length with the Facebook Security Team, and at the time we do not plan to make any changes to our site.”

[geographic location of a user]: “This information is not expected to be private, so we don't consider that aspect a vulnerability.”

[connection status of users]: “If it is possible to exploit in the manner described, we would like to address this.”

“The relationship of a member to a company is not considered private information on our site, so we don't consider that aspect a vulnerability.”

Mitigation
(current status)

https://bugzilla.mozilla.org/show_bug.cgi?id=1211669

“This isn't an issue Firefox can solve on its own; much of it is inherent in the design of the features. After the paper is published we can work with Chrome folks and other browser vendors to see if there are any reasonable ways to address this”

https://code.google.com/p/chromium/issues/detail?id=539390

“It seems like this is very similar to another report about cache-based side channel attacks from "The spy in the sandbox," so I'm marking this as a duplicate.”