UNIVERSITÄT
BERN

Phishing on Demand

Improvements & Survey

Yannick Hänni

Software Composition Seminar
01 June 2021

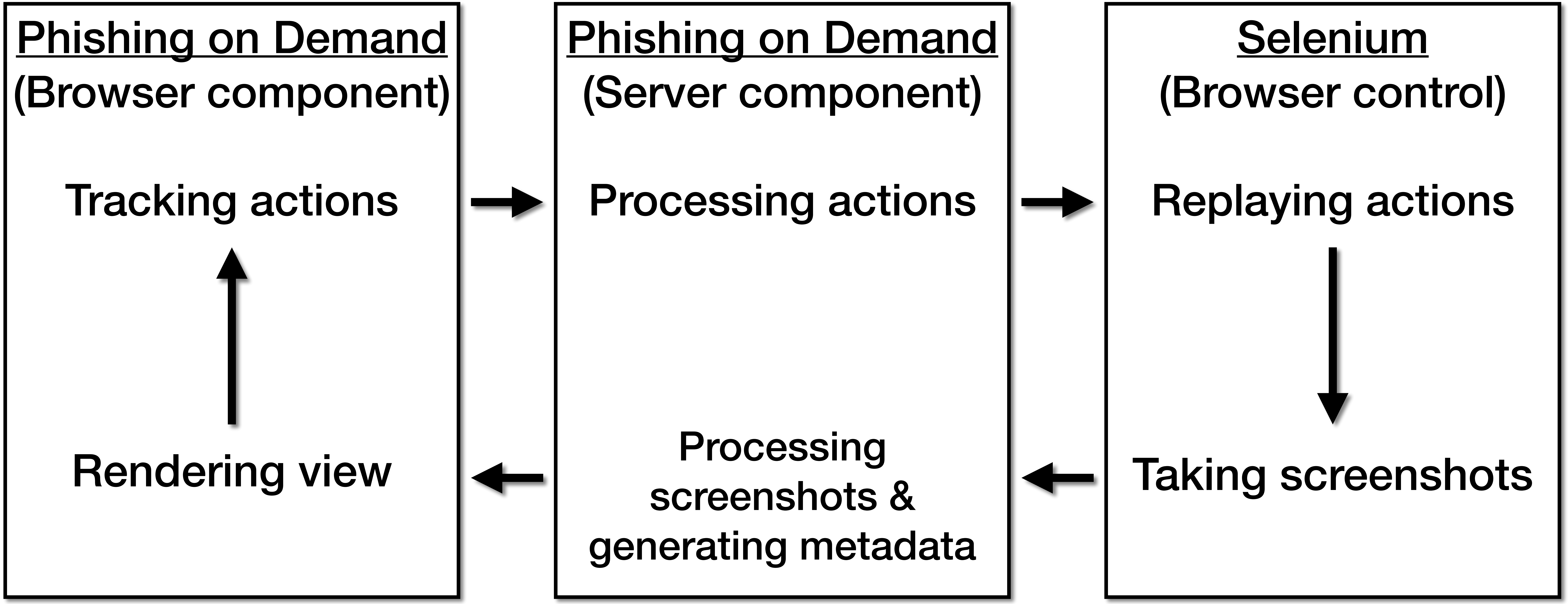
What is “Phishing on Demand” ?

A portable phishing framework.

No programming knowledge required.

Supports Mac OS, Windows, and Linux.

Architecture



Phishing on Demand
(Browser component)

Tracking actions

Rendering view

Phishing on Demand
(Server component)

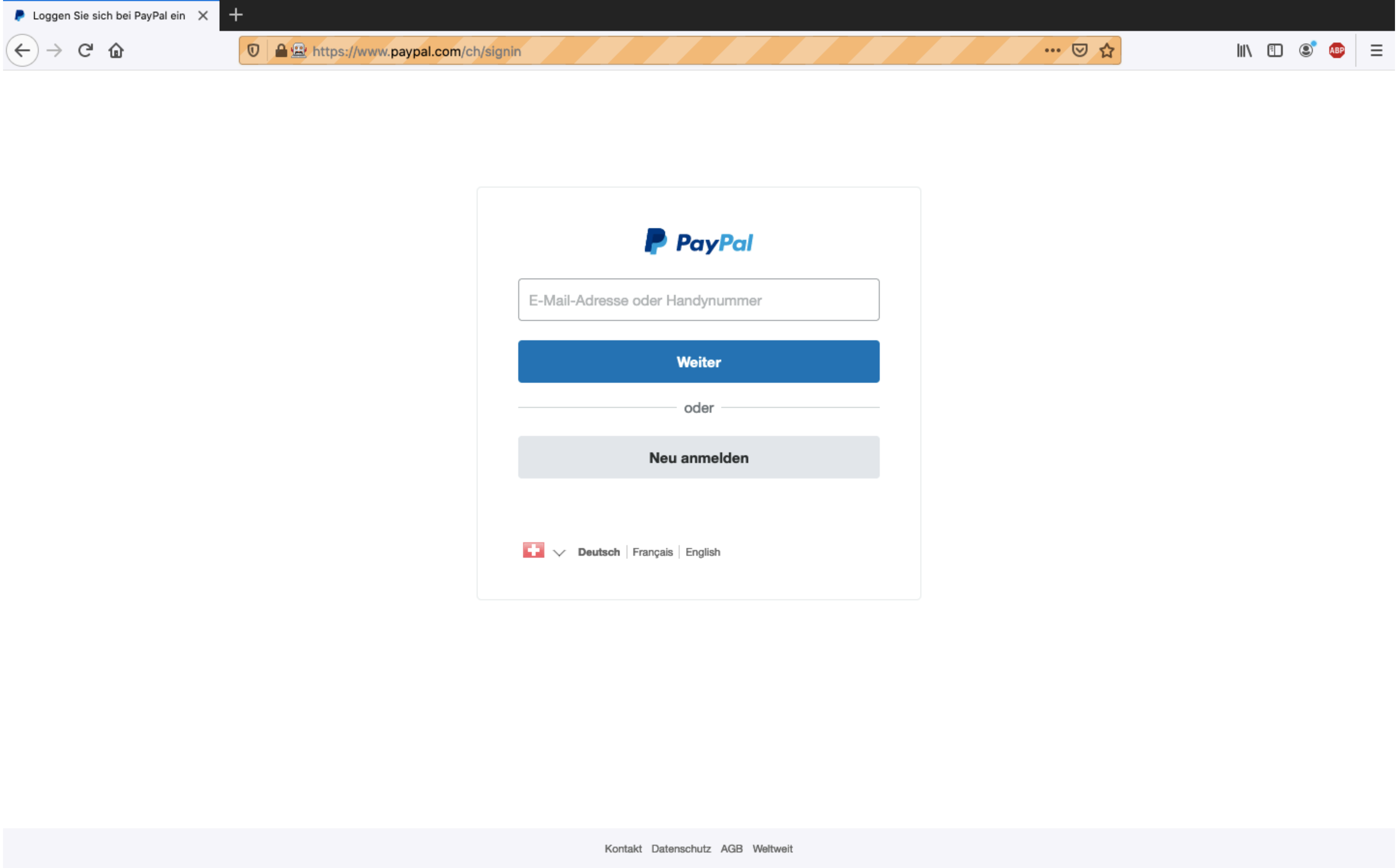
Processing actions

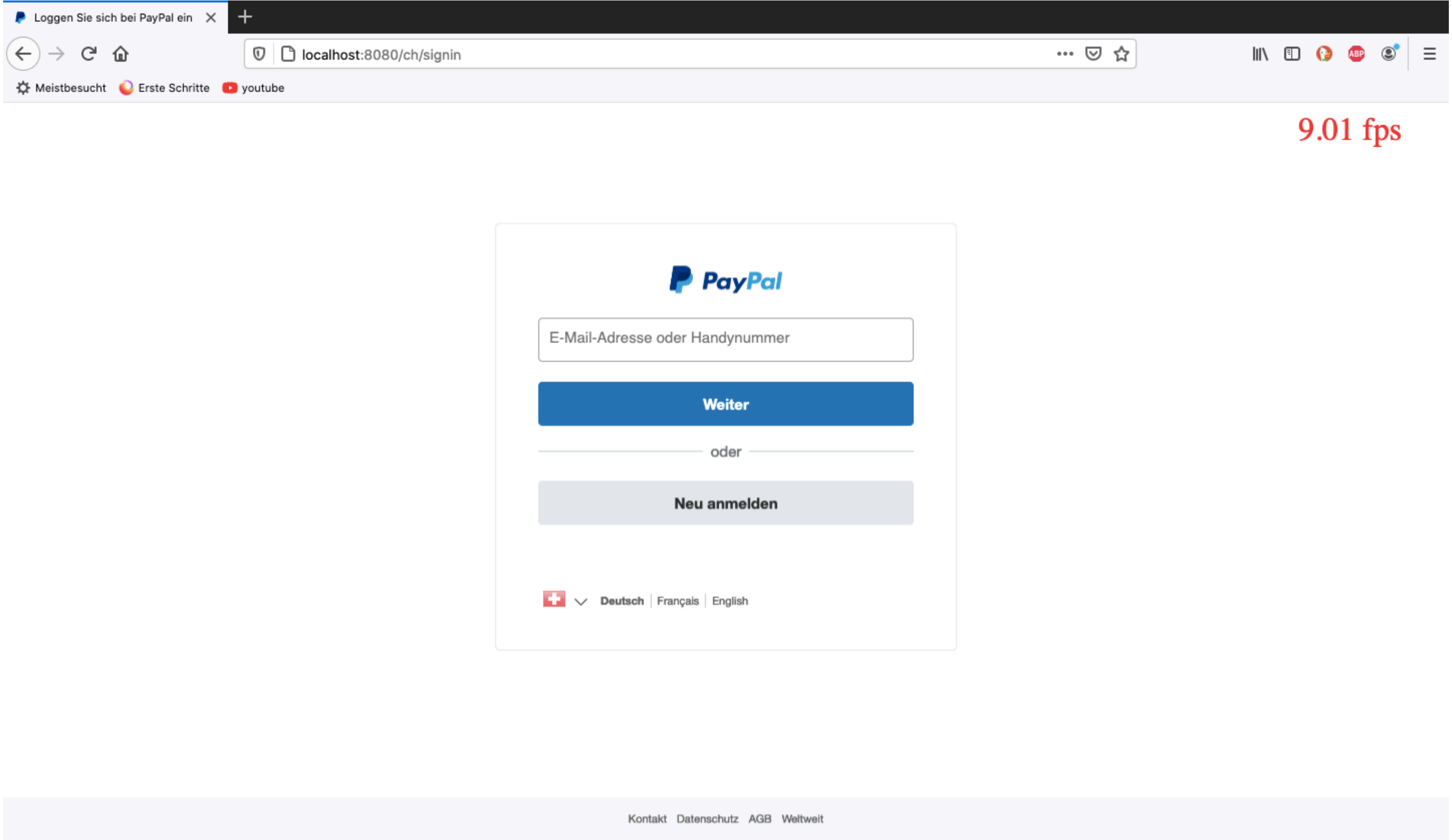
Processing screenshots & generating metadata

Selenium
(Browser control)

Replaying actions

Taking screenshots


Loggen Sie sich bei PayPal ein
https://www.paypal.com/ch/signin
PayPal
E-Mail-Adresse oder Handynummer
Weiter
oder
Neu anmelden
Deutsch
Français
English
Kontakt
Datenschutz
AGB
Weltweit


Loggen Sie sich bei PayPal ein
localhost:8080/ch/signin
Meistbesucht
Erste Schritte
youtube
9.01 fps
PayPal
E-Mail-Adresse oder Handynummer
Weiter
oder
Neu anmelden
Deutsch
Français
English
Kontakt
Datenschutz
AGB
Weltweit

New functionality: Browser history

Website - Die Post
localhost:8080/de/
✓ Website - Die Post
Post schweiz - Google Suche
paypal - Google Suche
wikipedia - Google Suche
Google
Google
Neuer Tab
ube
Jobs
Über uns
Hilfe und Kontakt
efe versenden
Pakete versenden
Empfangen
Standorte
Geschäftslösungen
Sendungsnummer, Suchbegriff eingeben

Neue Phishing-Welle im Namen der Post

Achtung aktuelles Betrugsmuster: Die Post wickelt keine Zahlungen über post.ch.paysget.info ab.

Mehr erfahren

Challenges

Caching

State change detection

Correct rerouting

Cross site tracking

What we have tried…

Using the JavaScript history manipulation method

`popstate(event e)`

Which button was pressed?

Workaround corrupts browser history!

What we have tried…

Using the favicon URL to determine the current state

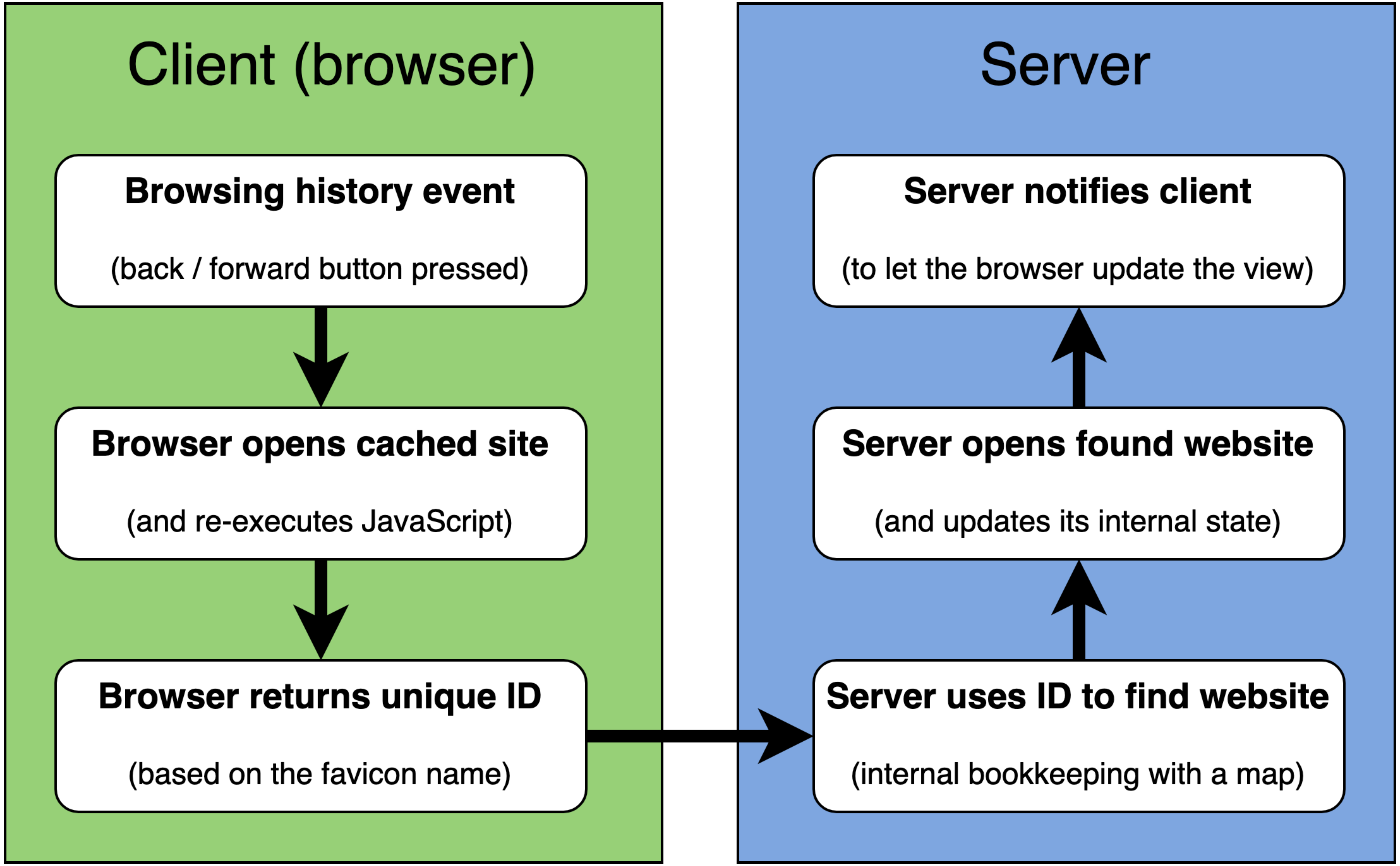
```
document.querySelector("link[rel*='icon'").href
```

… but there was a new problem: browser caching.

How to avoid browser caching ?

Random favicon file names!

Final workflow



Client (browser)
Browsing history event
(back / forward button pressed)
Browser opens cached site
(and re-executes JavaScript)
Browser returns unique ID
(based on the favicon name)
Server
Server notifies client
(to let the browser update the view)
Server opens found website
(and updates its internal state)
Server uses ID to find website
(internal bookkeeping with a map)

Result

Working navigation buttons

Authentic browser history

Cross site tracking

No caching issues

Performance loss

Minor favicon inconsistencies

Further improvements and future work

Scrolling

Webdriver issues

Code cleanup

Popup window placement

Horizontal scrollbars on some pages

Reaction and loading times

FPS

Demo

Survey design

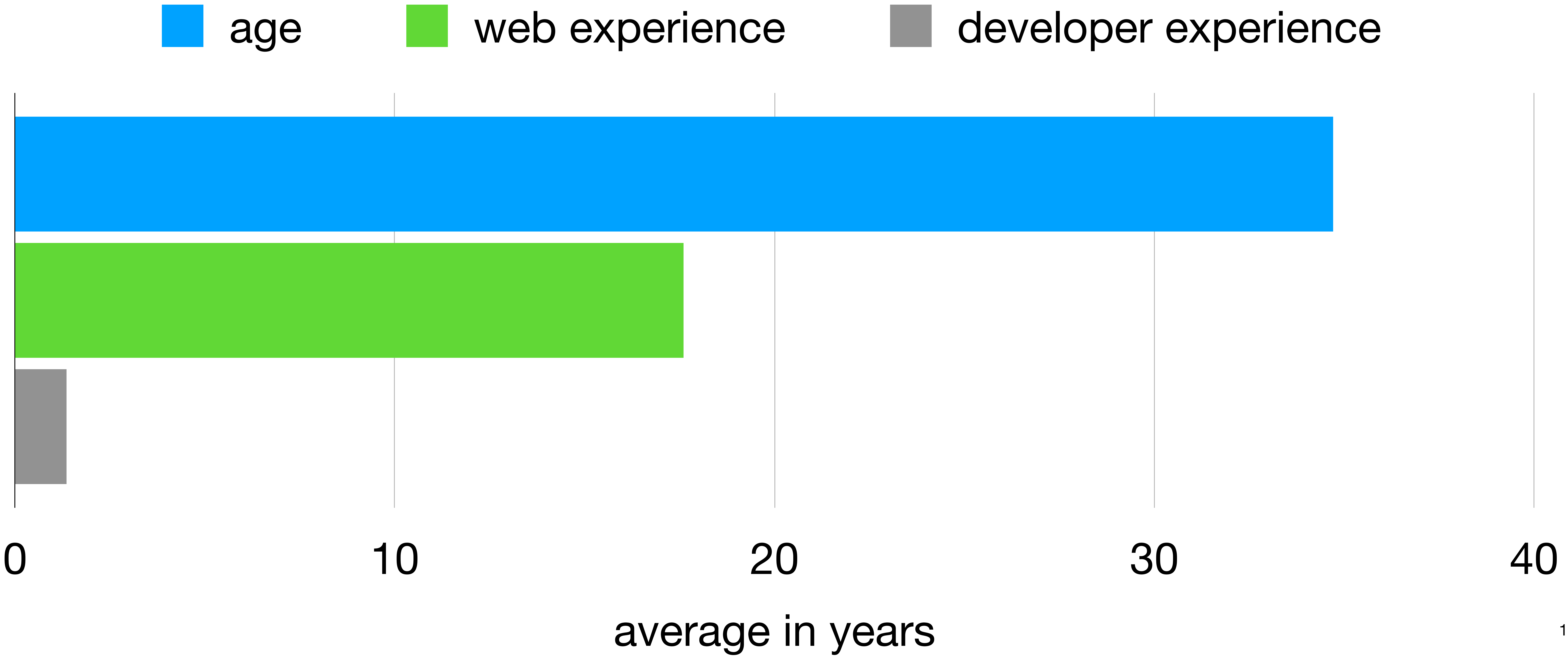
Personal information

Website usability tests

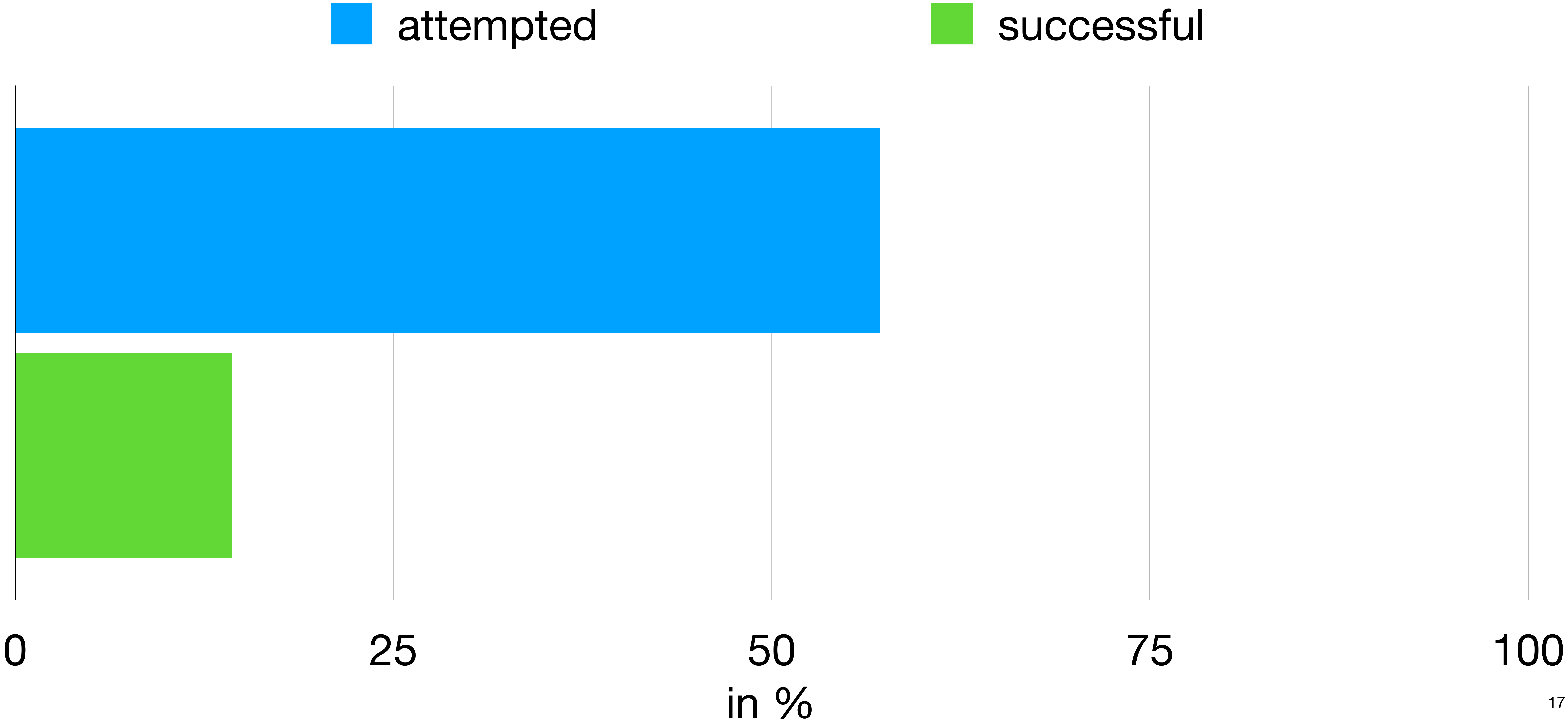
Phishing experience

Demography

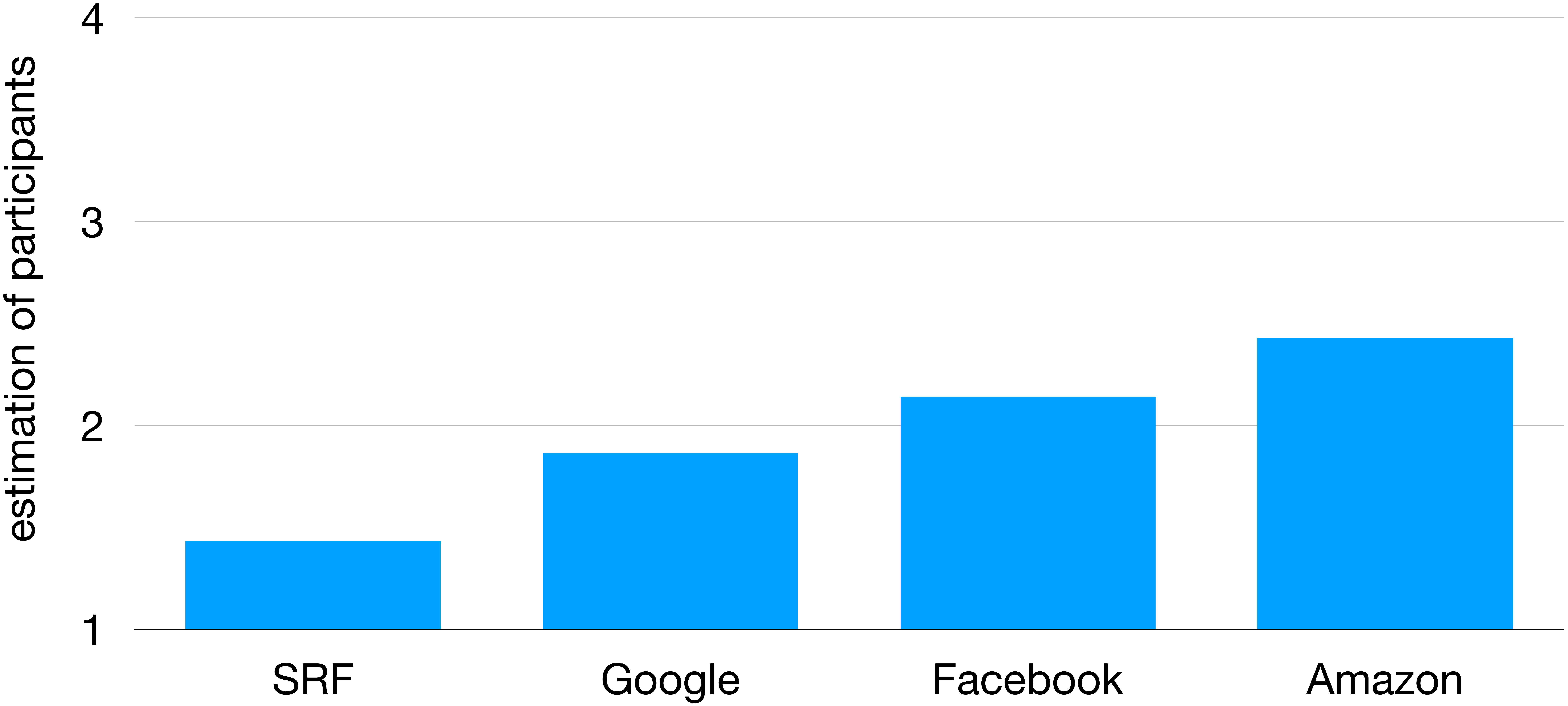
7 participants



Have participants been phished before?



Probability of phishing on website



Did participants notice the phishing ?

No!

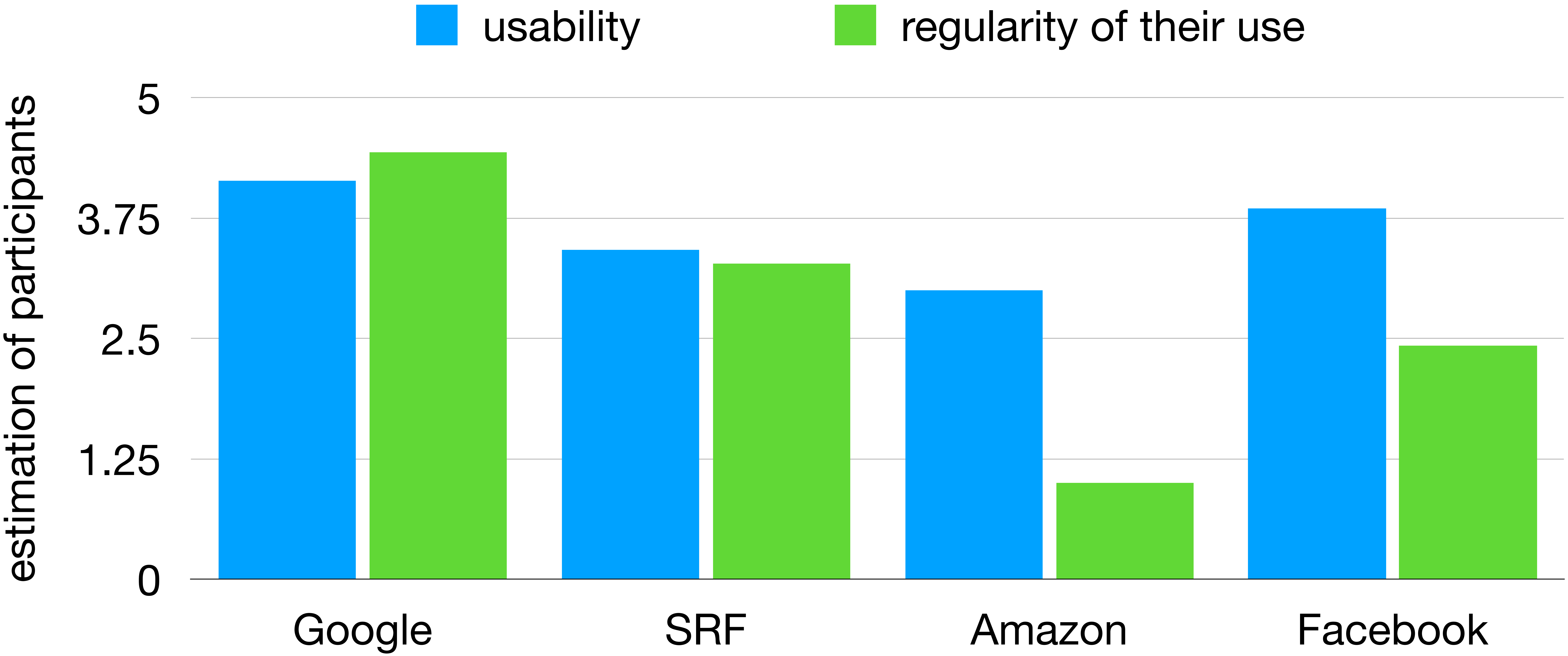
They noticed instead:

Lower performance

Repeated login

Scrollbars

Other observations



Conclusion

No more critical issues

Less minor issues

Better performance

Convincing results