Testing login process security of websites

Benjamin Krumnow

Technology
Arts Sciences
TH Köln

Benjamin Krumnow

[redacted]

Technology
Arts Sciences
TH Köln

Initial Project: “Shepherd”

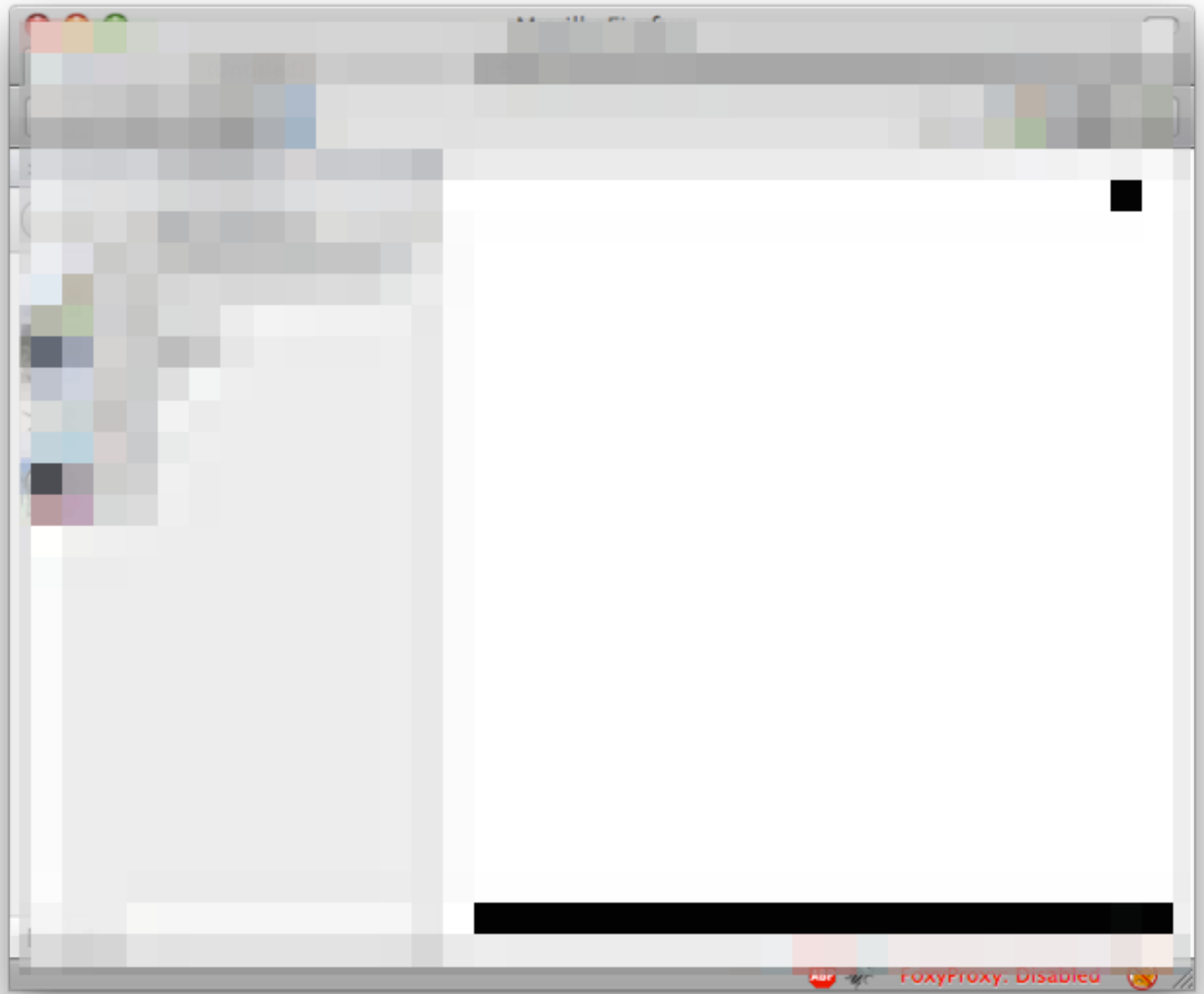
- Marc Sleegers, B.Sc., master student at the Open University
- Bachelor Thesis, March 2017 [1]
  - **Counting Sheep - Analysing online authentication security**
    - Mentor: dr. ir. H.L. Hugo Jonker
    - Coordinator: dr. ir. H. Harrie Passier
    - Examiner: prof. dr. T. Tanja Vos, prof. dr. M.C.J.D Marko van Eekelen

Agenda

1. **Background**
    - Firesheep
    - Attack vectors in 2010 and now
2. **Testing tools for research**
    - Tools for scanning
    - Comparison between static and dynamic scans
3. **Wrap-up**

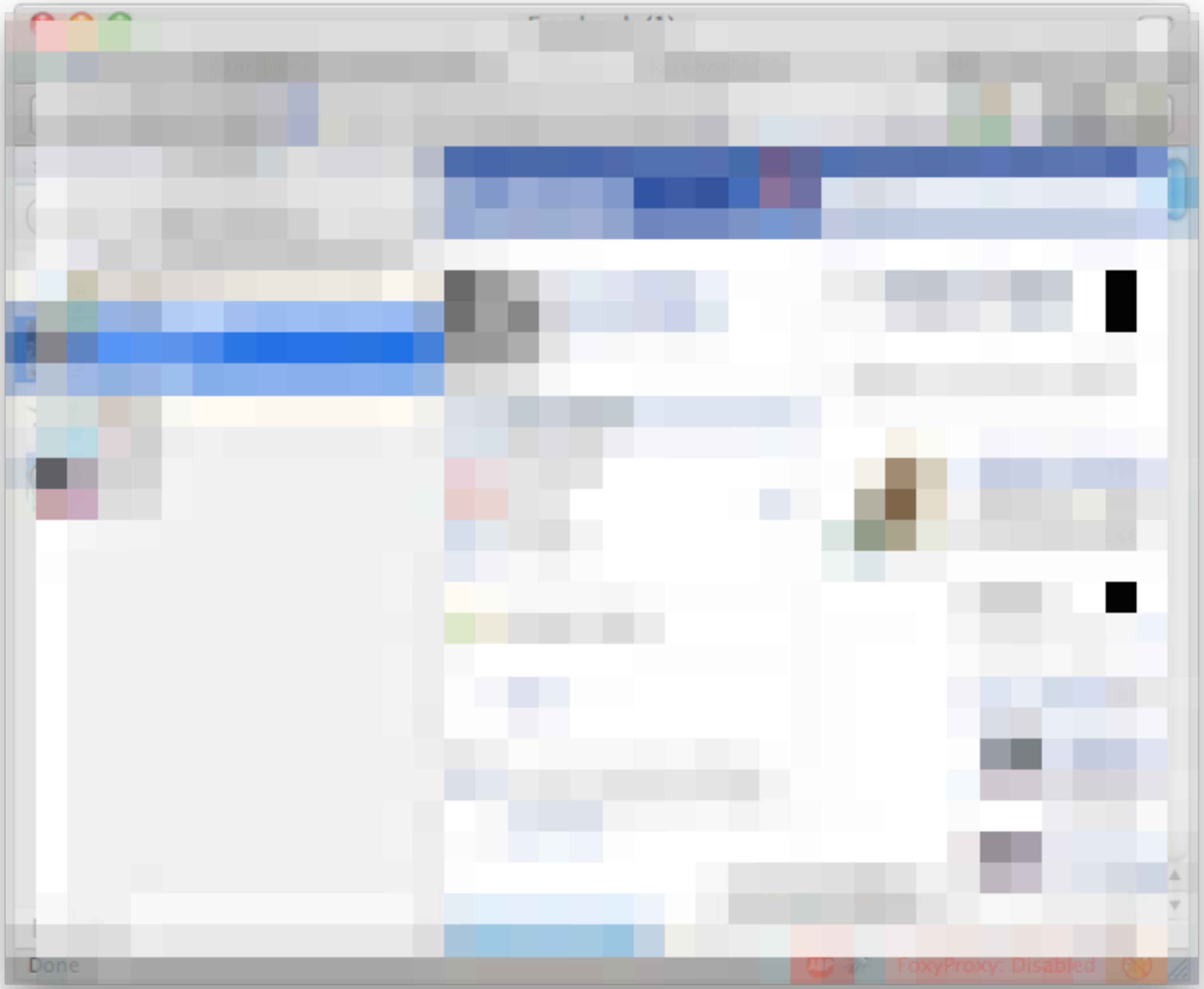
Motivation: Firesheep

Firesheep add-on in 2010



[2]

Hacking for everyone



[2]

Pressure on popular services

- It was easy to do for everyone, due to a browser add-on
  - Out of scope sides demand to write a script
- Huge media attention

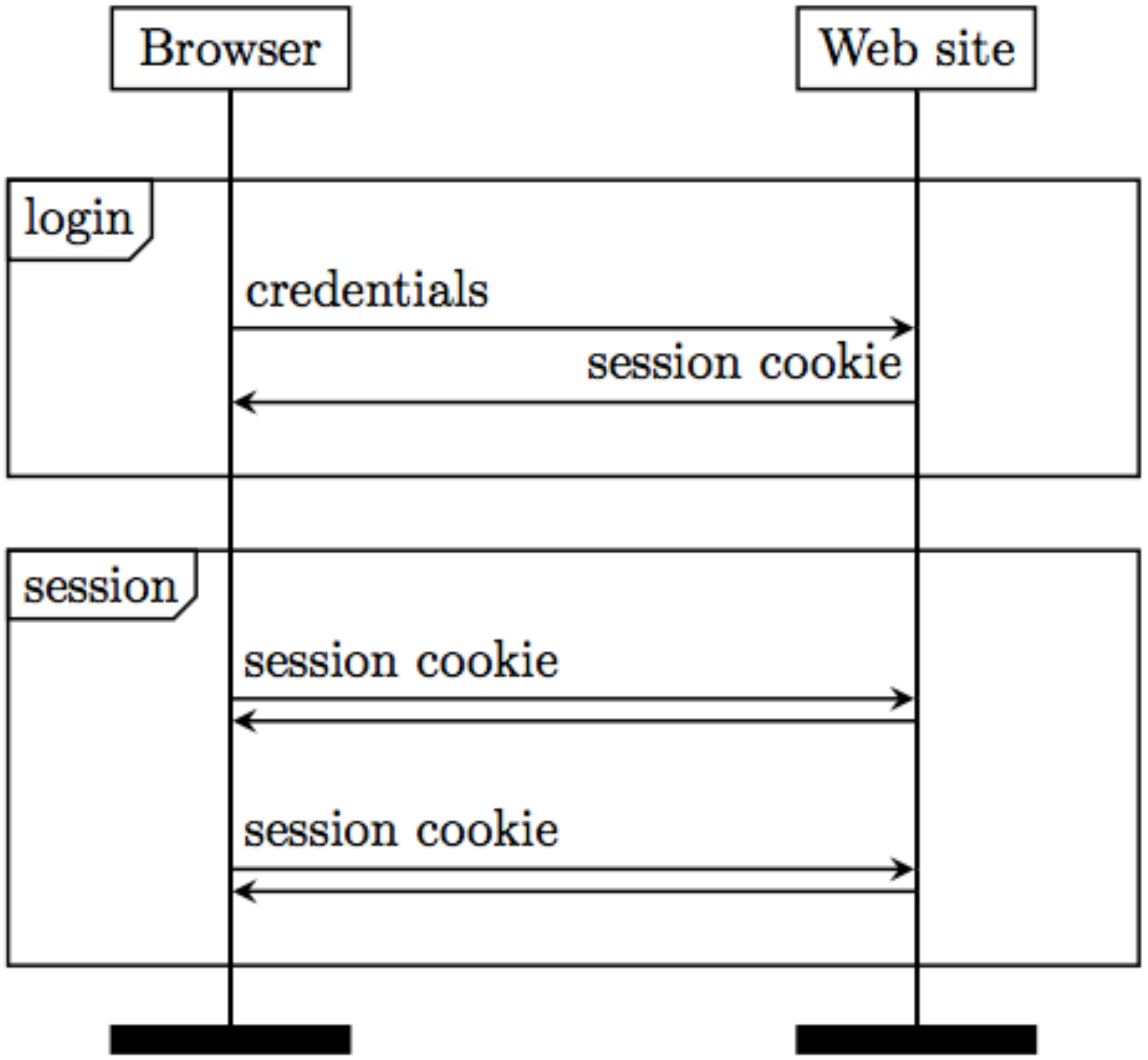
- Affected Facebook, Google…and they fixed it:
  - Deployment of TLS (SSL)

- Security in WiFi Networks
  - WPA, WPA2

The attack

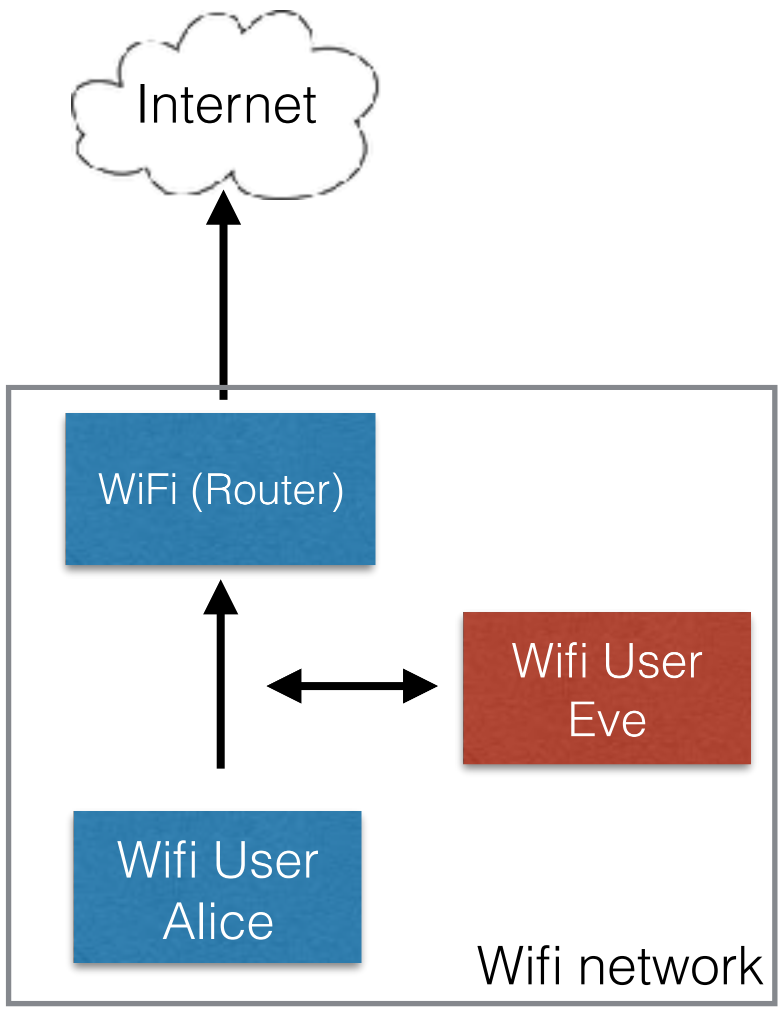
“Cookie stealing”

Login processes



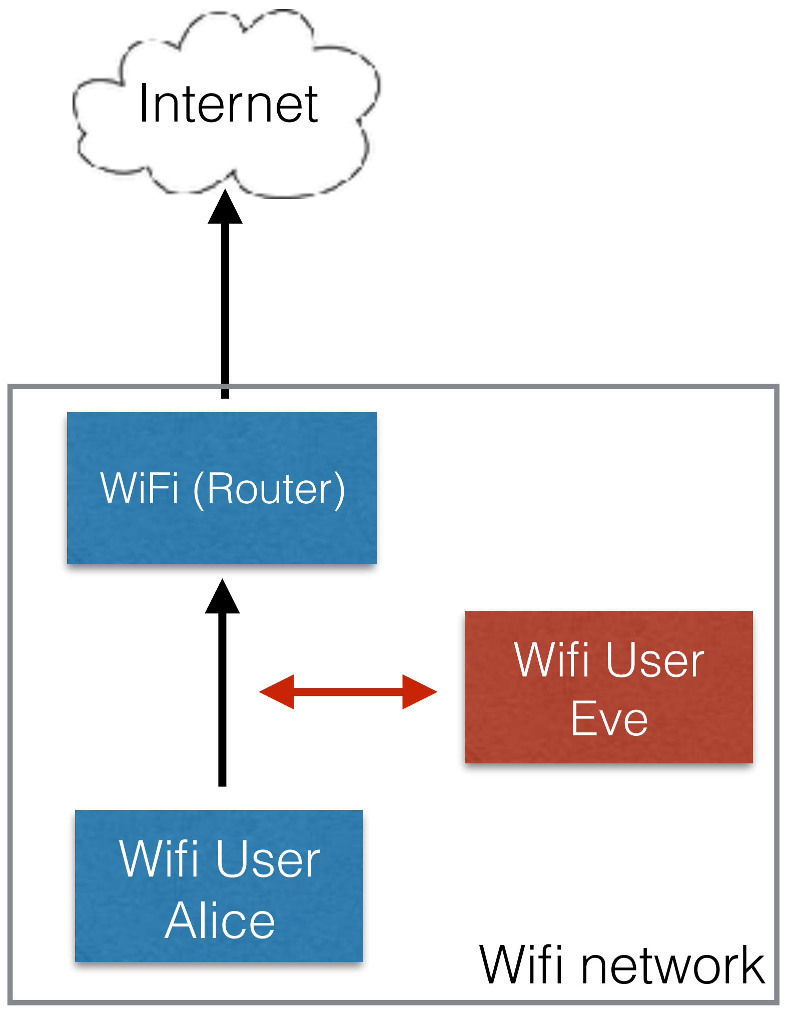
How to eavesdrop on WiFi in 2010

- Due to unencrypted and WEP WIFIs, promiscuous mode was often enough



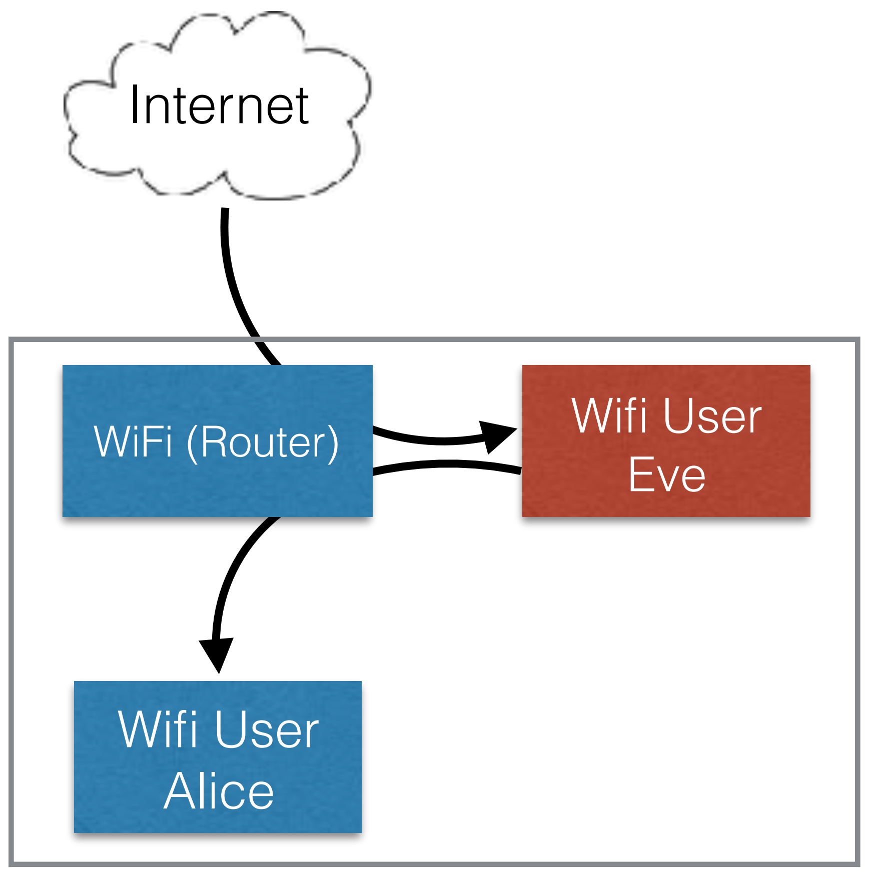
…and in 2017

- Deployment of WPA and WPA2
- Encrypted connections between access point/router and wifi users



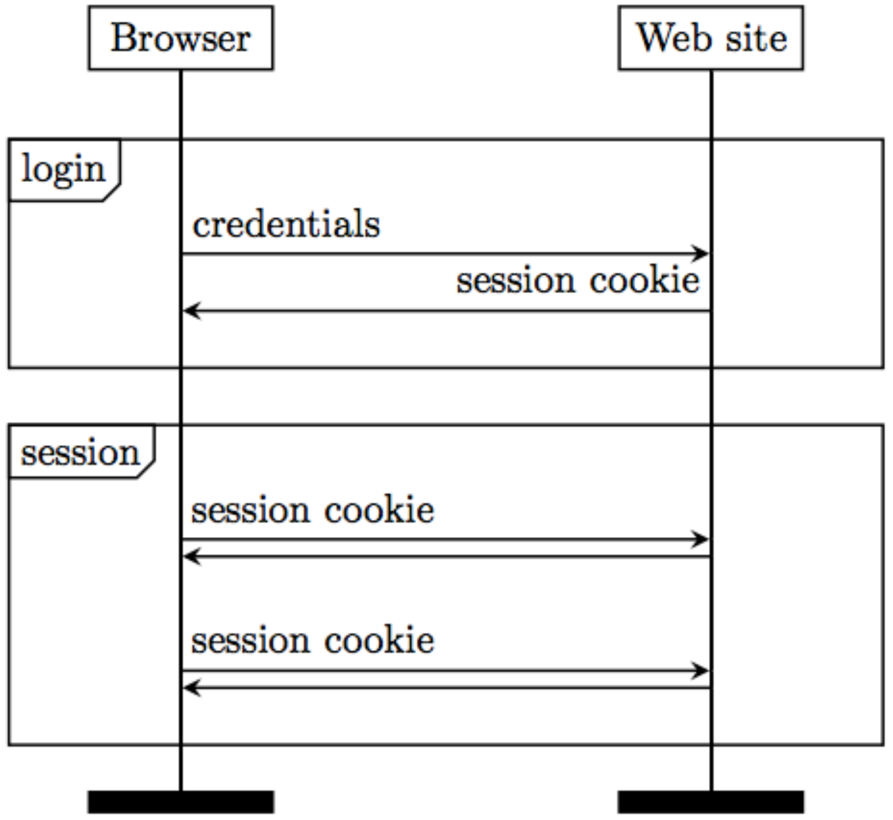
Becoming a MITM

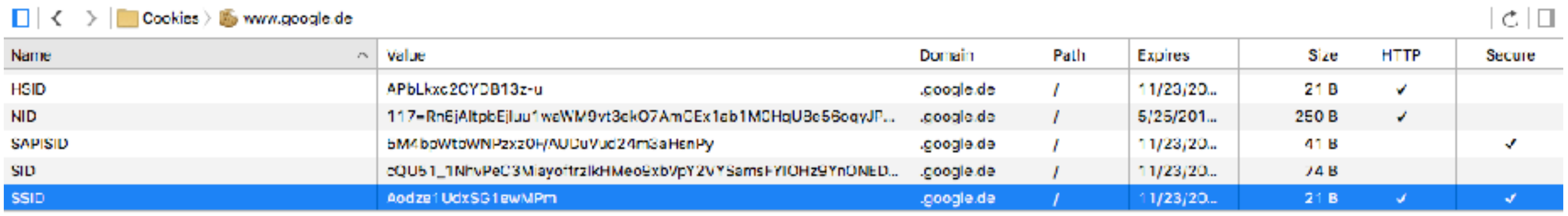
- Malicious access points
  - WIFI Pineapple Auditing Tool [4]
- Network attacks [3], e.g.
  - DHCP-based attacks
  - ARP spoofing
  - Still TLS/SSL encryption in place



Attacks in 2017

1. HTTP only
2. HTTPS first, then falling back to HTTP
3. HTTPS, but the secure flag is not set
   - Transmitting the cookies also via HTTP requests




Cookies › www.google.de

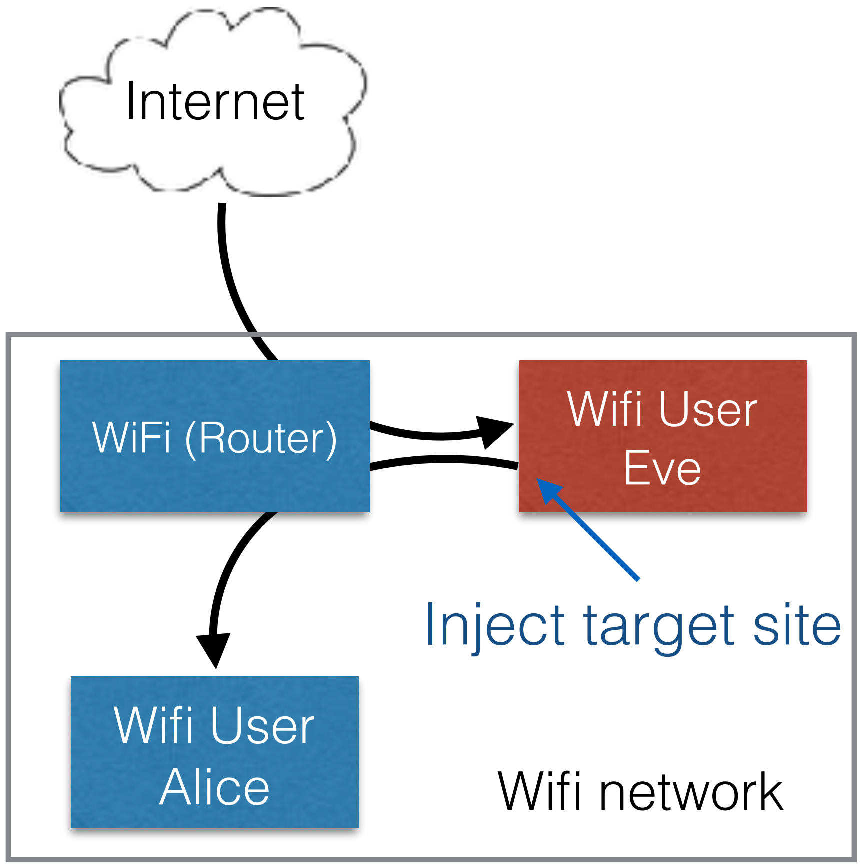
Name	Value	Domain	Path	Expires	Size	HTTP	Secure
HSID	APbLkxc2CYDB13z-u	.google.de	/	11/23/20...	21 B	✓	
NID	117=Rn6jAltpbEjIuu1weWM9vt3ckO7AmCEx1ab1MCHqUBe56oqyJP...	.google.de	/	5/25/201...	250 B	✓	
SAPISID	5M4bpWtdWNPzxz0H/AUDuVud24m3aHenPy	.google.de	/	11/23/20...	41 B		✓
SID	cQU51_1NFvPeC3MiayoffrzlkHMeo8xbVpY2VYSamsFYIOHz9YnONED...	.google.de	/	11/23/20...	74 B		
SSID	Aodze1UdxSG1ewMPm	.google.de	/	11/23/20...	21 B	✓	✓

How to make another client’s browser access a vulnerable site?

Cross-Site Request Forgery - Attacks

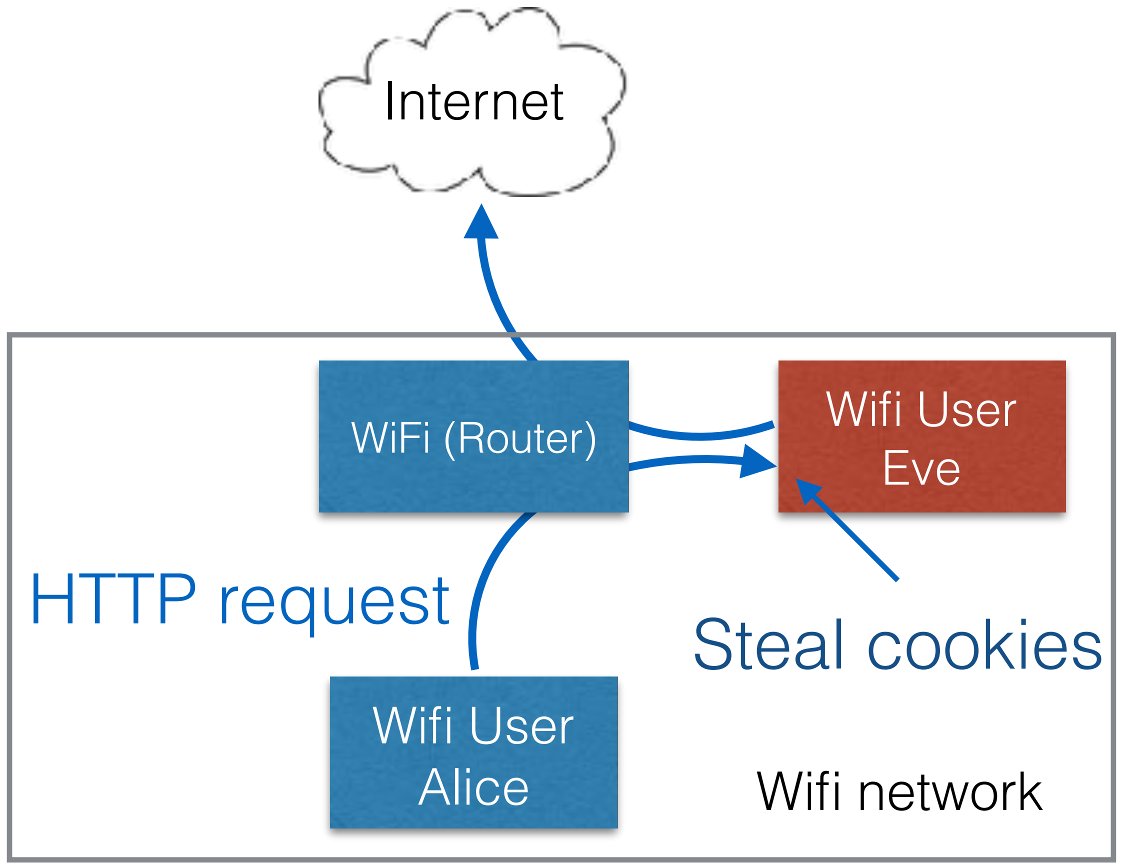
- Waiting for any HTTP traffic with a head element
- Injecting one or more URL(s) to target site(s)
- <link type="text/css" href="**http**://target_url/style.css">

- No interference by noScript or HTTPS-everywhere
- 3rd-party-cookies must be allowed
  - Except Safari, this is the default setting



Cross-Site Request Forgery - Attacks

- Injection triggers request by the victim’s browser via **HTTP**
- The request contains all the cookies for the target site, which is called
- Eve steals Alice’s cookies



Testing Websites

Testing Motivation

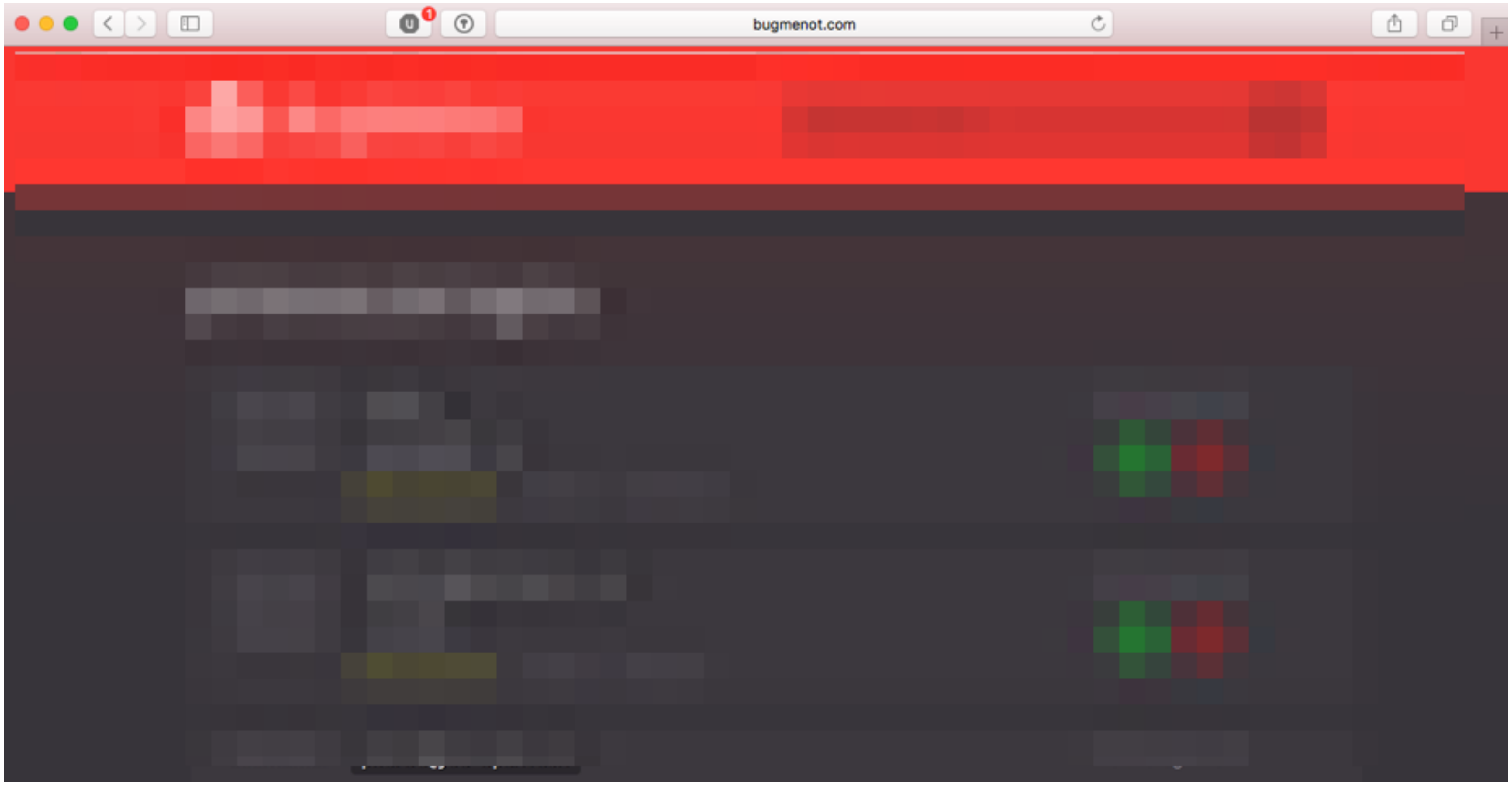
- Goals:
  - Long known vulnerabilities in the web
  - How far has security of website login processes evolved?
    - Test validity of attacks
    - Measure the widespread of related vulnerabilities
      - **Testing tools!**
        - Selenium, PhantomJS, CasperJS,…

Methodology

1. Need to login on websites to evaluate vulnerability
    a. acquire credentials for websites
2. Build an automatic vulnerability scanner
    a. make a choice of implementation
    b. find login pages
    c. submit credentials to login
    d. evaluate login state
    e. check for existence of vulnerabilities

Where to obtain login credentials from?

Acquiring credentials



[5]

Acquiring credentials

- Restrictions
  - Paid-content accounts
  - Age verification
  - Opt-out
  - fraud risk associated sites

- Terms of use
  - "*You agree never to access any form of networked device while not wearing happy pants.*" [5]

Scanners

3 classes of tools

1. Static tools
   - Downloading the HTML(, javascript, css,…) file of a site
   - Parse HTML
   - Browser-based functionality from websites (such as JavaScript) will not be executed!
2. Headless Browser
   - Dynamic, executes JavaScript
   - Some lack functionality, e.g. PhantomJS [6]
   - Error prone
   - new development here: headless Chrome
   - Performance gain?

3 classes of tools

3. Full-functioning consumer browsers with automatisation tools
    - Dynamic, behaves like your real browser
    - Selenium (browsers are interchangeable, even headless)
    - Interactions are executed within the browser
    - Might be slower?

Two Scanner Solutions

- Python-based scanner
    - Download the HTML file of a site and parse with BeautifulSoup
    - Website’s script will not be executed!
    - No waiting for elements to be loaded
    - Performance!
- Python-based Selenium scanner
    - Load website within a browser and perform operations for that specific website
    - Interaction can be done via JavaScript or within Python
    - Far more possibilities
    - Side effects due to the dynamics in website
    - Slow because of waiting time

Detecting login pages

1. Scan for login fields (<input type=”password”…/>)
    - Landing page
    - <a hrefs*=“keyword”>, keywords = “login, signin, …”
        - Translations
    - Brute force -> www.example.org/login
    - Sub levels of href

Detecting login pages

2. Scanning with the dynamic version
    - Each single page load takes time!
    - Brace yourself! Traversing sites can be difficult
        - TimeOutException
        - StaleElementReferenceException
        - ElementNotVisibleException
        - OutOfBoundException
        - Popups, iFrames, Alerts

[7]

Detecting login pages #2

2. Scanning with the dynamic version
    - Range of logins
        - Social logins (many implications)
        - two-step logins
    - Clickable / interactive elements
        - Difficult because every element can be clickable
        - Tradeoff due to the scanner's speed!

[8]

Detect login forms

- Static
  - improved algorithm of the scrapy framework - Rating
  - Login forms only
- Dynamic
  - Visible elements!
  - Higher range, due to not form-based login elements

Logging in

- Static
  - Submit a form
- Dynamic
  - Type credentials, be aware of changes

- Evaluate successful login
  - Cookies, 200 status code, visible login elements —> False positives
  - Re-accessing the site with login cookies

Results

- Results of the Bachelor thesis (February 2017)
    - Credentials from the Alexa Top 500.000 (46,548 credentials of BNM)
    - 30.376 (~65%) “login pages” detected
    - 4.976 (~16%) successful logins
    - 3.996 (~80%) vulnerable sites

- Static improved version (November 2017)
    - Credentials from the Alexa Top 1M (59.626 credentials of BNM)
    - 9.330 (~32%) login pages detected
    - 6,741 (~34%) successful logins
    - 4.946 (~73%) vulnerable sites

Comparison

500 website set	Static	Dynamic
Time	~ *47 mins*	~2,5 h
Login page detection	206 (41,2%)	369 (73,8%)
Logins	75 (36%)	94 (25%)

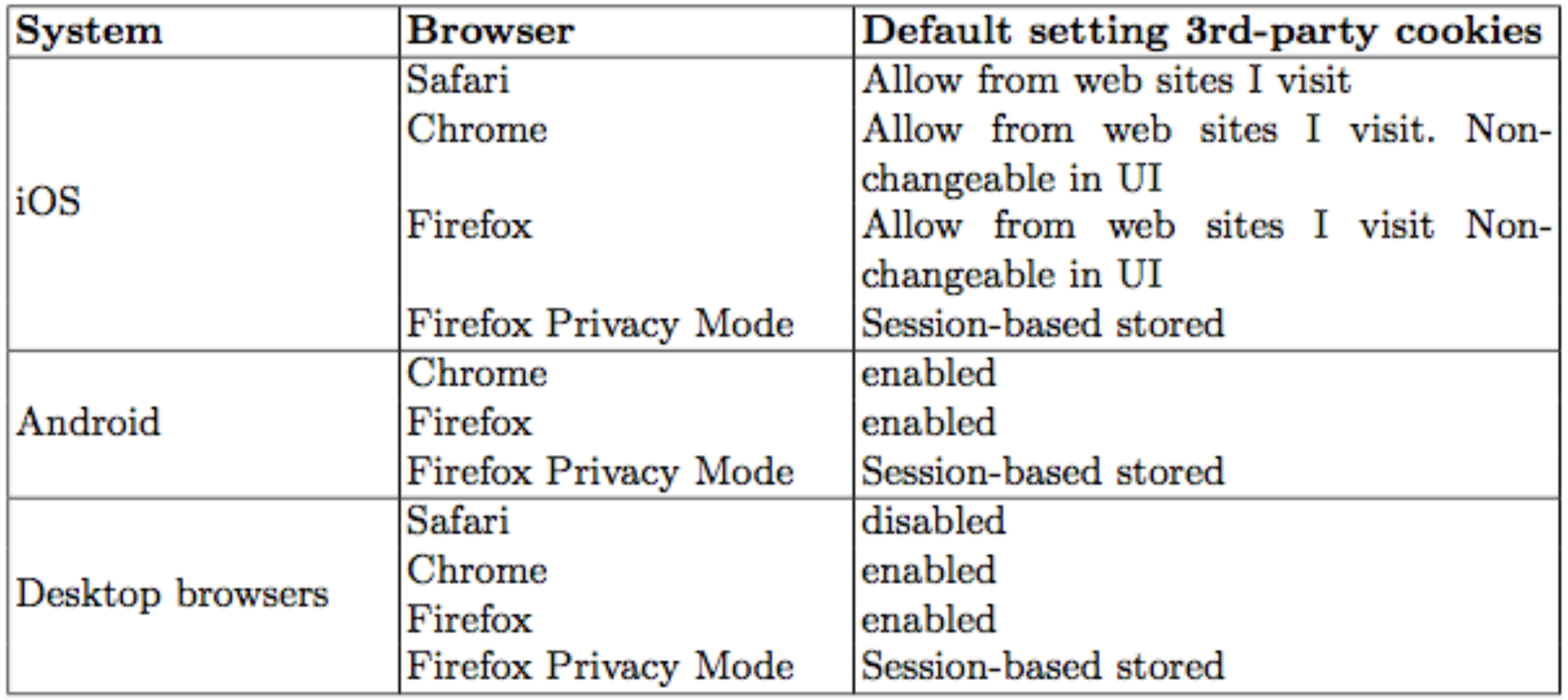
Wrap up

Summary

	Static	Dynamic
Performance	Fast	relative slow
Possiblities	Limited to static elements	Full consumer browser
Complexity	Lower	Higher due to dynamics

Securing your website

- Protect yourself (and your users)
  - Set secure flag on cookies
  - Deploy HSTS on your own servers
  - Deactivate 3rd-party cookies (not possible on iOS)
  - Use private browsing mode or delete cookies after each session



System	Browser	Default setting 3rd-party cookies
iOS	Safari	Allow from web sites I visit
	Chrome	Allow from web sites I visit. Non-changeable in UI
	Firefox	Allow from web sites I visit Non-changeable in UI
	Firefox Privacy Mode	Session-based stored
Android	Chrome	enabled
	Firefox	enabled
	Firefox Privacy Mode	Session-based stored
Desktop browsers	Safari	disabled
	Chrome	enabled
	Firefox	enabled
	Firefox Privacy Mode	Session-based stored

Thank you

References

[1] Counting Sheep - Analysing online authentication security
Marc Sleegers, March 2017

[2] FireSheep
Eric Butler, 2010
https://codebutler.github.io/firesheep/tc12/, last seen 23th of March 2017.

[3] A survey of man in the middle attacks.
Mauro Conti, Nicola Dragoni, and Viktor Lesyk. IEEE Communications Surveys & Tutorials, 18(3):2027– 2051, 2016.

[4] The WIFI Pineapple Wireless Auditing Platform
https://www.wifipineapple.com/

[5] BugMeNot
http://bugmenot.com/terms.php

[6] A.: Online tracking: A 1-million-site measurement and analysis
Engelhardt, S., Narayanan. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1388–1401 (2016)

References

[7] Wikipedia Login page
https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Main+Page, last seen 24th of November 2017.

[8] Skyscanner Login page
https://www.skyscanner.net/, last seen 24th of November

Questions