Testing login process security of websites

Benjamin Krumnow
Benjamin Krumnow
Initial Project: “Shepherd”

- Marc Sleegers, B.Sc., master student at the Open University
  - **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
Hacking for everyone
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

- Background
- Testing websites
- Wrap up

Diagram:
- Browser
- Web site
- login
  - credentials
  - session cookie
- session
  - session cookie
  - session cookie
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
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

![Diagram](attachment:image.png)
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
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
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!
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

<table>
<thead>
<tr>
<th>500 website set</th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>~ 47 mins</td>
<td>~2,5 h</td>
</tr>
<tr>
<td><strong>Login page detection</strong></td>
<td>206 (41.2%)</td>
<td>369 (73.8%)</td>
</tr>
<tr>
<td><strong>Logins</strong></td>
<td>75 (36%)</td>
<td>94 (25%)</td>
</tr>
</tbody>
</table>
Wrap up
# Summary

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>Fast</td>
<td>relative slow</td>
</tr>
<tr>
<td><strong>Possibilities</strong></td>
<td>Limited to static</td>
<td>Full consumer browser</td>
</tr>
<tr>
<td></td>
<td>elements</td>
<td></td>
</tr>
<tr>
<td><strong>Complexity</strong></td>
<td>Lower</td>
<td>Higher due to dynamics</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>System</th>
<th>Browser</th>
<th>Default setting 3rd-party cookies</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Safari, Chrome, Firefox, Firefox Privacy Mode</td>
<td>Allow from web sites I visit. Non-changeable in UI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allow from web sites I visit. Non-changeable in UI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Session-based stored</td>
</tr>
<tr>
<td>Android</td>
<td>Chrome, Firefox, Firefox Privacy Mode</td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Session-based stored</td>
</tr>
<tr>
<td>Desktop browsers</td>
<td>Safari, Chrome, Firefox, Firefox Privacy Mode</td>
<td>disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Session-based stored</td>
</tr>
</tbody>
</table>
Thank you
References

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

[2] FireSheep
   Eric Butler, 2010

   Mauro Conti, Nicola Dragoni, and Viktor Lesyk. IEEE Communications Surveys & Tutorials,

   https://www.wifipineapple.com/

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

References


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