

浅谈信息安全主战场的转移



LinE@CloverSec Labs

1

• Whoami

2

• 物联网的套路

3

• 方法论

4

• 举个栗子

5

• 技能点/兵器库

我是谁

```
[LinE@CloverSec]:~# whoami
```

```
ID : LinE , Real name : YuanWei , Age : 23 years old
```

```
[LinE@CloverSec]:~# groupinfo
```

```
LinE in the cloversec group
```

```
[LinE@CloverSec]:~# cat LinE_Info.txt
```

- 小伙伴们喜欢叫我懒妞 or 烂。。。 T_T
- 打小就想知道那绿油油的电路板到底能做什么，对那神秘的世界心怀敬畏，然后不知道拆坏了多少玩意（学习成本嘛）
- 家里大大小小的电器基本上都遭到过我的摧残，从而锻炼了自己良好的维修能力（呃，专业修电脑）
- 从初三开始接触黑客，玩鸽子，搞入侵，探内网，看代码，写程序，到后来的搞电工，做网络，玩无线，改路由，做四轴，一发不可收拾，然后入了物联网的坑 233333
- 个人惯用 Debian，字符界面的那种，因为指法娴熟，手速惊人，遂荣幸的成为了，呃，网管（明明是OP好么！！！！）
- 身高185，体重114，喜欢玩，喜欢结交各类好汉（不是.py的那种），24K纯直男不搞基谢谢
- 曾参加某次众测关于智能路由漏洞的挖掘，SSCTF线下出了一道NFC破解的题，嵌入式漏洞挖掘挑战赛挖掘到5个高危
- 一次机缘巧合结识了坤哥，有幸加入了四叶草，成为了CloverSec Lab的IoT方向负责人，欢迎各种技术非技术骚扰~

1

• Whoami

2

• 物联网的套路

3

• 方法论

4

• 举个栗子

5

• 技能点/兵器库

物联网的套路



物联网的套路



1

• Whoami

2

• 物联网的套路

3

• 方法论

4

• 举个栗子

5

• 技能点/兵器库

How To Do It

让我们聊聊方法论



如何从一无所知到开启上帝模式

黑盒阶段

1. 了解功能，使用范围，使用方法，以及能做什么
2. 拆机看PCB，从各种组件上了解其架构，寻找调试接口（TTL/JTAG）
3. 加电，进行常规性检测（扫端口，看服务等）
4. 截取信号进行分析，看它发送了什么，这些信号都是做什么的
5. 弄到固件，拆包分析，对其中的关键程序进行逆向
6. 重点关注Ping/Telnet等功能，尝试命令执行，进入白盒阶段
7. 自制添加了后门的固件，尝试刷入，进入白盒阶段
8. 其他脑洞大开的想法，做法

白盒阶段

1. 以高权限登入设备，对自己的一些想法进行验证。
2. 对外通信内容进行分析，构造Payload，跑一下
3. 连接调试接口，看终端打印信息
4. 利用QEMU进行动态调试，下断试错等
5. 从终端到云端（如果有的话）
6. 站在上帝视角，寻找更多问题，物联网不只是pwn it就完了
7. 一个小玩意引发的血案（基于物联网设备的内网漫游）

1

• Whoami

2

• 物联网的套路

3

• 方法论

4

• 举个栗子

5

• 技能点/兵器库

搞硬件
不仅要能硬起来，也要能软下去

举个栗子

现在，有这么一个路由器，官方未给出固件，也没有在线更新，怎么搞到固件？

固件存放在flash中，要拿固件，搞他硬盘（Flash）

硬搞：拆机，取芯片，扔编程器读取，拿到整个flash的内容

软搞：拆机，连调试，从U-boot这里断下启动过程，利用U-Boot的查看内存功能搞定

所谓硬起来

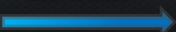
硬搞：拆机，取芯片，扔编程器读取，就像dd一样
拿到整个flash的内容



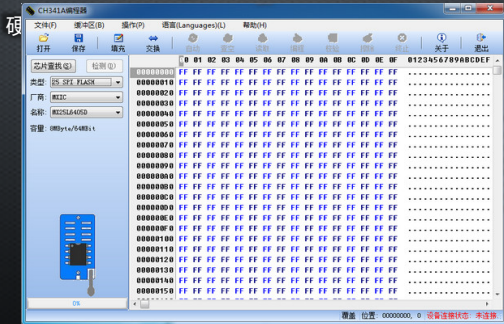
NOR Flash



Nand Flash



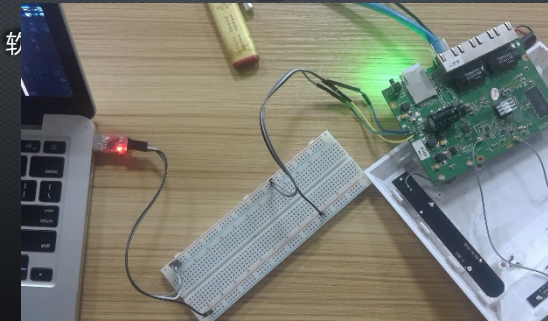
所谓硬起来



然后软下去

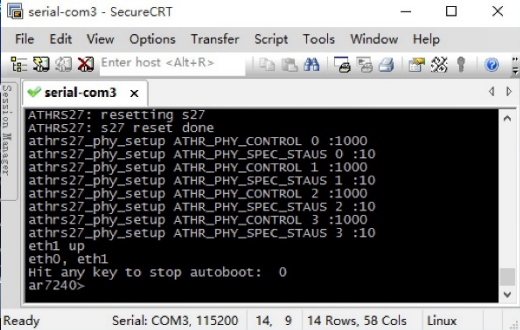
软搞：拆机，连调试，从U-boot这里断下启动过程
利用U-boot的查看内存功能搞定

然后软下去



然后软下去

软



```
serial-com3 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com3 x
ATHRS27: resetting s27
ATHRS27: s27 reset done
athrs27_phy_setup ATHR_PHY_CONTROL 0 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 0 :10
athrs27_phy_setup ATHR_PHY_CONTROL 1 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 1 :10
athrs27_phy_setup ATHR_PHY_CONTROL 2 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 2 :10
athrs27_phy_setup ATHR_PHY_CONTROL 3 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 3 :10
eth1 up
eth0, eth1
Hit any key to stop autoboot: 0
ar7240>
```

Ready Serial: COM3, 115200 14, 9 14 Rows, 58 Cols Linux

```
serial-com3 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com3 x
eth0, eth1
Hit any key to stop autoboot: 0
ar7240>
ar7240>
ar7240>
ar7240>
ar7240> ?
? - alias for 'help'
boot - boot default, i.e., run 'bootcmd'
bootd - boot default, i.e., run 'bootcmd'
bootm - boot application image from memory
cp - memory copy
erase - erase FLASH memory
help - print online help
md - memory display
mm - memory modify (auto-incrementing)
mtest - simple RAM test
mw - memory write (fill)
rm - memory modify (constant address)
ping - send ICMP ECHO_REQUEST to network host
printenv - print environment variables
progmac - Set ethernet MAC addresses
reset - Perform RESET of the CPU
run - run commands in an environment variable
setenv - set environment variables
tftpboot - boot image via network using TFTP protocol
version - print monitor version
ar7240>
```

```
serial-com3 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com3 x
ar7240> printenv
bootargs=console=ttyS0,115200 root=31:01 rootfstype=jffs2 rw init=/sbin/
init mtdparts=ath-nor0:64k(u-boot),6976k(rootfs),1024k(uImage),64k(mib0)
,64k(ART)
bootcmd=bootm 0x9f6e0000
bootdelay=4
baudrate=115200
ethaddr=0x00:0xaa:0xbb:0xcc:0xdd:0xee
ipaddr=192.168.1.1
serverip=192.168.1.10
dir=
lu-tftp 0x80060000 ${dir}tuboot.bin&&erase 0x9f000000 +${filesize}&&cp.b $
fileaddr 0x9f000000 $filesize
lf-tftp 0x80060000 ${dir}tozedcpe-5g-jffs2&&erase 0x9f010000 +0x6d0000&&
cp.b $fileaddr 0x9f010000 $filesize
lk-tftp 0x80060000 ${dir}vmlinux.lzma.uimage&&erase 0x9f6e0000 +${filesiz
e&&cp.b $fileaddr 0x9f6e0000 $filesize
stdin=serial
stdout=serial
stderr=serial
ethact=eth0

Environment size: 665/65532 bytes
ar7240>
```

#

```

Environment size: 665/65532 bytes
ar7240> md 80060000 64
80060000: 8e030000 24422000 54e20007 8e020000    ... $B .T.....
80060010: 02002821 0c017a68 02203021 54400014    .. (!..zh. 0!T@..
80060020: 00001021 8e020000 3c068000 3c03802b    ...!...<...<..+
80060030: 7e244b00 00461021 8c65dfa0 00042080    ~$K..F.!..e....
80060040: 00021302 26430040 00021140 ae630000    ....&C.@...@.C..
80060050: 00a21021 00451027 00021143 00021300    ...!.E.'...C....
80060060: 00461021 0801801c 00441021 00001021    .F.!.....D.!...!
80060070: 8fbf0024 8fb30020 8fb2001c 8fb10018    ...$. ... ..
80060080: 8fb00014 03e00008 27bd0028 23bdf0d0    ... .. (#...
80060090: afb30028 00e09821 afb10020 00c08821    ...C...!... ..!
800600a0: afb0001c 00a08021 afbf002c afb20024    .....!.....$
800600b0: 8cc20010 8c920000 30420001 1440003b    .....0B...@.;
800600c0: 2404ffea 3c02802a 244307e0 90620039    $.<...*$C...b.9
800600d0: 30420004 14400005 00000000 9062002d    0B...@.....b.-
800600e0: 30420008 14400004 27a60010 0c003be4    0B...@.'.....;
800600f0: 02202021 27a60010 02002821 0c017ff1    . !'.....(!....
80060100: 02402021 0404fff4 10400028 00408021    .@ !.....@.C.@.!
80060110: 88420000 2403ffbf 00411024 14400023    .B..$. ...A.$@.#
80060120: 2404fff0 8e220000 30428000 10000002    $. ... " 0B.....
80060130: 02201821 8e23000c 24620004 c0430000    . !.#.. $b...C..
80060140: 24630001 e0430000 10600213 00000000    $c...C... ..
80060150: 8e420048 02202021 24420001 0c019765    .B.H. ! $B.....e
80060160: ae420048 3c02802b 8c42dfa0 02220023    .B.H<...+.B...".#
80060170: 00021143 00021300 00531025 30430040    ...C.....S.%0C.@
80060180: 10600009 aa020000 3a040004 2406ffbf    .`.....:..$.
ar7240>

```

在

```
Environment size: 665/65532 bytes
ar7240> md 80060000 64
80060000: 8e030000 24422000 54e20007 8e020000    ....$B .T.....
80060010: 02002821 0c017a68 02203021 54400014    ..(!..zh. 0!T@..
80060020: 00001021 8e020000 3c068000 3c03802b    ...!....<...<..+
80060030: 7e244b00 00461021 8c65dfa0 00042080    ~$K..F.!..e....
80060040: 00021302 26430040 00021140 ae630000    ....&C.@...@.C..
80060050: 00a21021 00451027 00021143 00021300    ...!.E.'...C....
```

4

```
[line@LinE-MacBook-Pro]: /Users/line/route
→ file uImage.hex [2016-06-02 15:52:37]
uImage.hex: u-boot legacy uImage, Linux Kernel Image, Linux/MIPS, OS Kernel Image (lzma), 961251 bytes, Fri Jan 30 11:22:09 2015, Load Address: 0x80002000, Entry Point: 0x801E66D0, Header CRC: 0x0BF7628B, Data CRC: 0xA60A8B32
[line@LinE-MacBook-Pro]: /Users/line/route
→ █ [2016-06-02 15:52:41]
```

```
80060130: 02201821 8e23000c 24620004 c0430000    . .!.#..$b...C..
80060140: 24630001 e0430000 10600213 00000000    $c...C...`.....
80060150: 8e420048 02202021 24420001 0c019765    .B.H. !$B.....e
80060160: ae420048 3c02802b 8c42dfa0 02220023    .B.H<...+.B...".#
80060170: 00021143 00021300 00531025 30430040    ...C.....S.%0C.@
80060180: 10600009 aa020000 3a040004 2406ffbf    .`.....:....$...
ar7240> █
```

再举个栗子

现在，有这么一个机器，通过无线连接到云端，接受云端的指令
然而机器自身没有发现问题，怎么知道机器与云端的通信内容？

通信必然要通过网络，要拿流量，直接抓

从硬件层面：使用Throwing Star LAN Tap等类似设备直接搭线攻击

从软件层面：使用WooyunWiFi或者类似的拥有流量抓取功能的路由（基于OpenWRT）截取cap

再举个栗子

现在，有这么一个机器，通过无线
然而机器自身没有发现问题，怎



接受云端的指令
端的通信内容？

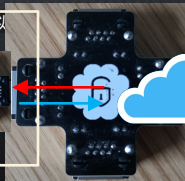


通信必然要通过网络，网络接口抓

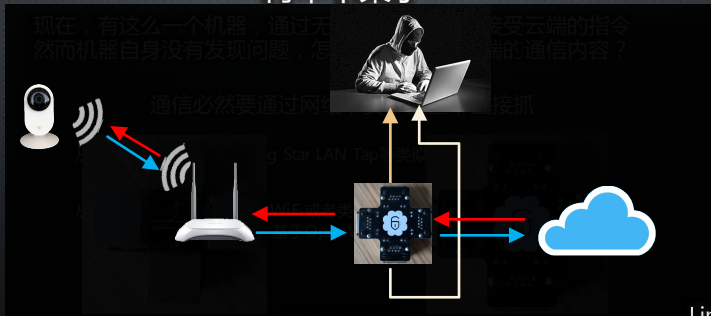


g Star LAN Tap等类似

WiFi或者类
基于OpnW



再举个栗子



再举个栗子

现在，有这么一个设备，通过遥控器进行控制，遥控器使用433MHz
如何知道遥控器发送了什么？

想要知道，直接抓~，用HackRF去捕捉信号

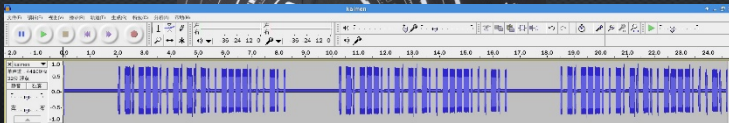


得到的原始信号长这样
这里有个小诀窍，在抓取的时候建议偏离中心信号一点，比如432.7MHz
可以避免信号尖峰影响

再举个栗子

现在，有这么一个设备，通过遥控器进行控制，遥控器使用433MHz
如何知道遥控器发送了什么？

想要知道，直接抓~，用HackRF去捕捉信号

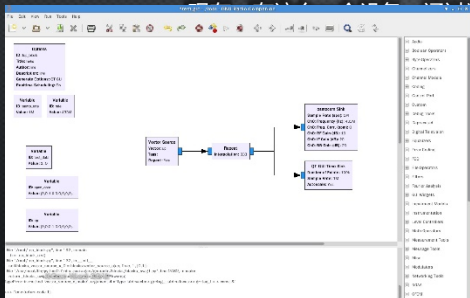


得到的原始信号长这样

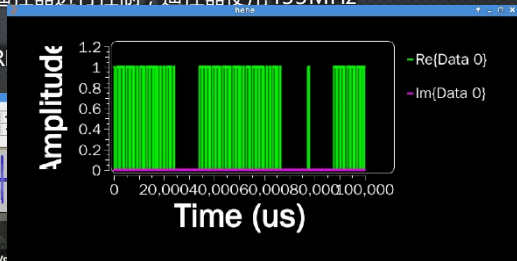
这里有个小诀窍，在抓取的时候建议偏离中心信号一点，比如432.7MHz
可以避免信号尖峰影响

再举个栗子

遥控器进行控制，遥控器使用433MHz



可以避开信号大峰影响



偏置中心信号点，比如432.71MHz

1

• Whoami

2

• 物联网的套路

3

• 方法论

4

• 举个栗子

5

• 技能点/兵器库

技能加点

- Linux系统得会玩吧，这可是物联网的基石。
- 嵌入式/Arduino开发板总要玩过吧
- 常见单片机（例如51，STM32）总得把玩一番
- 逆向工程好歹得会吧，不光是x86，还有ARM和MIPS架构的
- Android总得了解吧，调试应用啊，抓个包啊，逆个APK啊，ADB调试，RemoteADB
- 网络架构，通信方面总得入个门吧
- 常见的嵌入式解决方案得有个了解
 - 路由器解决方案（OpenWRT？DD-WRT？开源方案？自主研发？）
 - 摄像头解决方案（嵌入式Linux还是Android）
 - 手持终端解决方案（Android？Linux？自研？）
- 电子电路入个门，需要的时候焊个板子吧~
- 最后，对新鲜事物的接受能力、动手能力和学习能力！！！！很重要！！！！

我的兵器库

软件篇

虚拟机：VMware Workstation，Virtual Box，Parallels Desktop

终端软件：SecureCRT，SecureFX

逆向软件：OllyDbg，IDA

动态调试：QEMU虚拟机+逆向软件

固件分析：Binwalk，squashfs-tools，7zip，010editor，C32ASM

代码编辑：Sublime Text，NotePad++，Visual Studio Code

编程语言：Python，C，Java

流量分析：tcpdump，wireshark

无线射频：GNURadio，aircrack-ng，

Web调试：Burp Suite，Chrome开发者工具

我的兵器库 硬件篇

趁手的螺丝刀，我用的南旗的22in1，拆机片，撬棒，吹风机等等
趁手的电烙铁，我用的自己DIY的白菜白光，配刀头，尖头，马蹄头发热芯，焊锡，助焊剂
USB转TTL，USB转串口，USB转JTAG，USB转各类奇葩接口
Arduino学习套件，外围模块，面包板，洞洞板，杜邦线，各类电子相关的配件
HackRF One，UberTooth One，Proxmark3，Throwing Star LAN Tap，Teensy
刷了OpenWRT的路由器（WiFiPineApple），8187/3070网卡+各类天线，树莓派3
逻辑分析仪，芯片编程器等等
一双无所不能的手，一个停不下来的大脑，一颗永不言弃的心

Hacking For Fun And Enjoy Just Do It



CloverSec
安全实验室

KCon West 2016

Designed By LinE @ CloverSec Labs

Thanks



CloverSec
安全实验室

KCon West 2016

Designed By LinE @ CloverSec Labs