



Reverse Engineering (REV3)

## **UE 05 – Firmware Analyse – Protokoll**

Jakob Mayr

WS 2023/2024

## File-Beschaffung

Die benötigte Firmware-Version kann direkt auf der netgear-Seite heruntergeladen werden:

[https://www.downloads.netgear.com/files/GDC/R6400/R6400-V1.0.1.12\\_1.0.11.zip](https://www.downloads.netgear.com/files/GDC/R6400/R6400-V1.0.1.12_1.0.11.zip)

## Analyse Teilkomponenten

Extrahieren des .zip-Archivs:

```
mendacium fedora > build > REV3UE05-Tools
7z x R6400-V1.0.1.12_1.0.11.zip
time:2ms

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,16 CPUs AMD Ryzen 7
5800U with Radeon Graphics (A50F00),ASM,AES-NI)

Scanning the drive for archives:
1 file, 27537183 bytes (27 MiB)

Extracting archive: R6400-V1.0.1.12_1.0.11.zip
--
Path = R6400-V1.0.1.12_1.0.11.zip
Type = zip
Physical Size = 27537183

Everything is Ok

Files: 2
Size: 27543800
Compressed: 27537183
mendacium fedora > build > REV3UE05-Tools
time:83ms
```

Die Ausführung des Befehls `binwalk` mit der Option `--signature` auf die Datei `r6400-V1.0.1.12_1.0.11.chk` liefert folgende Informationen über die Binärdatei der Firmware:

```
mendacium@kali-mendacium: ~/build/REV3
$ binwalk --signature --tera R6400-V1.0.1.12_1.0.11.chk

DECIMAL      HEXADECIMAL  DESCRIPTION
-----
58           0x3A         TRX firmware header, little endian, image size: 27537408 bytes, CRC32: 0x54507561, flags: 0x0, version: 1, header size: 28 bytes, loader offset: 0x1C, linux kernel offset: 0x0
86           0x56         LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size: 5246752 bytes
2141338      0x20AC9A     Squashfs filesystem, little endian, version 4.0, compression:xz, size: 25391589 bytes, 1402 inodes, blocksize: 131072 bytes, created: 2016-05-31 15:57:09

mendacium@kali-mendacium: ~/build/REV3
```

- **TRX Firmware-Header:** Das Binärbild enthält einen TRX Firmware-Header, der häufig bei Firmware-Dateien für Router oder ähnliche Geräte verwendet wird. Er ist in Little-Endian-Format mit einer Größe des Images von 27.537.408 Bytes und enthält einen CRC32-Prüfsummenwert. Ein TRX-Header ist wie folgt aufgebaut:

1	0																1																2																3															
2	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																																
3	+-----+																																																															

Das Firmware-Image hat nicht wirklich einen Bootloader im klassischen Sinne. Der "loader offset" zeigt auf `0x1C` und es handelt sich dabei um einen LZMA-Loader, dieser startet einen selbst-entpackenden Kernel und somit die Firmware. Betrachtet man die Stelle `0x1C` ("loader offset"), so findet man zu Beginn einen String `"U12H332T00_NETGEAR"` (radare2 visual mode). Dieser string gibt die Board ID an:

```

[0x0000001c [Xadvc]0 0% 1264 build/REV3UE05-Tools/R6400-V1.0.1.12.1.0.11.chk] xc 0 rip
- offset - 1c1d 1e1f 2021 2223 2425 2627 2829 2A2B CDEF0123456789AB comment
0x0000001c 0000 0000 72ed 1bc7 15dd 0b2c 5531 3248 .r.....U12H ; rip ; arg3 ; arg1 ; arg3 ; arg2
0x0000002c 3333 3254 3030 5f4e 4554 4745 4152 4844 332T00_NETGEARHD ; arg3
0x0000003c 5230 0030 a401 6175 5054 0000 0100 1c00 R0 0..auPT..... ; arg4
0x0000004c 0000 60ac 2000 0000 0000 5d00 0001 0020 .\.....]....
0x0000005c 0f50 0000 0000 0000 69bc 002e 3568 b600 .P.....l...Sh.
0x0000006c f976 6959 866f 63d5 f3bd 5519 3798 0565 .viY.oc...U.7..e

```

- **LZMA komprimierte Daten:** Der LZMA-komprimierte Teil (Lempel-Ziv-Markov-Kettenalgorithmus) im Header gibt Informationen über die Wortbuchgröße (65536 Bytes) und die unkomprimierte Datengröße des LZMA-Teils (5246752 Bytes). Die Daten sind vermutlich ausführbarer Code für einen Kernel, welche zuerst dekomprimiert und anschließend ausgeführt werden. Dies ist der Start der Firmware.
- **Squashfs-Dateisystem:** Das Binärbild umfasst ein Squashfs-Dateisystem. Squashfs ist ein komprimiertes, schreibgeschütztes Dateisystem für Linux. Die Version, der Kompressionstyp (xz), die Größe, die Anzahl der Inodes, die Blockgröße und das Erstellungsdatum werden ebenfalls angegeben. Das Erstellungsdatum des Squashfs-Dateisystems ist der 31. Mai 2016 um 15:57:09 Uhr.

Extrahieren der LZMA komprimierten Datei und des squashfs-Dateisystems:

```

(mendacium@kali-mendacium)~/build/REV3
$ dd if=R6400-V1.0.1.12.1.0.11.chk of=lzma skip=86 bs=1 count=2141252
2141252+0 records in
2141252+0 records out
2141252 bytes (2.1 MB, 2.0 MiB) copied, 6.36765 s, 336 kB/s

(mendacium@kali-mendacium)~/build/REV3
$ dd if=R6400-V1.0.1.12.1.0.11.chk of=filesystem skip=2141338 bs=1
25396128+0 records in
25396128+0 records out
25396128 bytes (25 MB, 24 MiB) copied, 76.0239 s, 334 kB/s

```

„Mounten“ des Dateisystems in „/mnt“:

```

mendacium@kali-mendacium:~/build/REV3
$ ls
R6400-V1.0.1.12.1.0.11.chk  binmalt  lzma
R6400-V1.0.1.12.1.0.11.chk  filesystem
'R6400-V1.0.1.12.1.0.11.Release Notes.html'  headers

(mendacium@kali-mendacium)~/build/REV3
$ ls /mnt

(mendacium@kali-mendacium)~/build/REV3
$ sudo mount -t squashfs -o loop filesystem /mnt/

(mendacium@kali-mendacium)~/build/REV3
$ ls /mnt
bin dev etc lib media mnt opt proc sbin share sys tmp usr var www

(mendacium@kali-mendacium)~/build/REV3
$

```

Suchen und finden der „httpd“-Binärdatei:

```

(mendacium@kali-mendacium)~/build/REV3
$ sudo find /mnt -name "httpd"
/mnt/usr/sbin/httpd

```

## httpd

Informationen (radare2) über die Binärdatei:

```
[0x000000fc]> i
fd      3
file     httpd
size     0x16ca1c
humansz  1.4M
mode     r-x
format   elf
iorw     false
block    0x100
type     EXEC (Executable file)
arch     arm
baddr    0x8000
binsz    1492530
bintype  elf
bits     32
canary   false
class    ELF32
compiler GCC: (GNU) 3.3.2 20031005 (Debian prerelease) GCC: (Buildroot 2012.02) 4.5.3
flags    0x5000002
noi      eabi5
crypto   false
endian   little
havecode true
intrap   /lib/ld-uClibc.so.0
laddr    0x0
lang     c
linenum  false
lsyms    false
machine  ARM
nx       true
os       linux
pic      false
relocs   false
relro    no
rpath    NONE
sanitize false
static   false
stripped true
subsys   linux
va       true
[0x000000fc]> |
```

Vergleicht man die httpd-Datei mit einer Datei aus der heruntergeladenen Open-Source/GPL R6400-GPL\_V1.0.1.12-1.0.11.zip, sind gewisse Ähnlichkeiten festzustellen:

```
mendacium@mendacium ~$ file ~/httpd
/home/mendacium/httpd: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-uClibc.so.0, stripped
mendacium@mendacium ~$ file router/arm-uclibc/target/usr/sbin/httpd
router/arm-uclibc/target/usr/sbin/httpd: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-uClibc.so.0, stripped
```

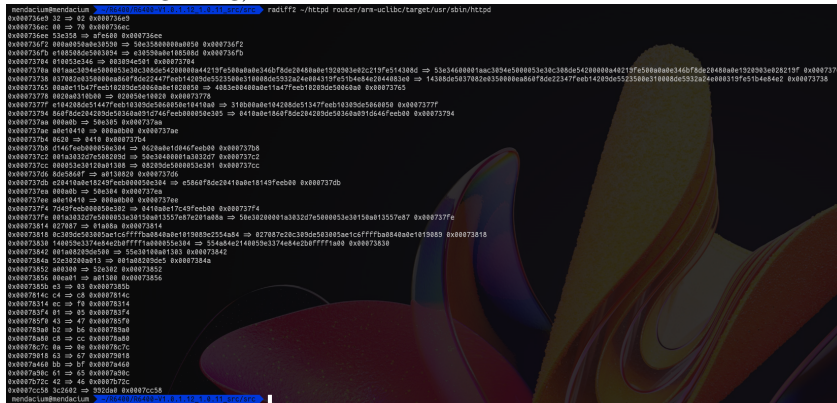
```
[0x000000fc]> i
fd      3
file     router/arm-uclibc/target/usr/sbin/httpd
size     0x16ca1c
humansz  1.4M
minopsz  4
maxopsz  4
lnvopsz  4
pcalign  4
mode     r-x
format   elf
iorw     false
block    0x100
type     EXEC (Executable file)
arch     arm
baddr    0x8000
binsz    1492530
bintype  elf
bits     32
canary   false
class    ELF32
compiler GCC: (GNU) 3.3.2 20031005 (Debian prerelease) GCC: (Buildroot 2012.02) 4.5.3
crypto   false
endian   little
havecode true
intrap   /lib/ld-uClibc.so.0
laddr    0x0
lang     c
linenum  false
lsyms    false
machine  ARM
nx       true
os       linux
pic      false
relocs   false
relro    no
rpath    NONE
sanitize false
static   false
stripped true
subsys   linux
va       true
```

Figure 1: downloaded

```
[0x000000fc]> i
fd      3
file     /home/mendacium/httpd
size     0x16ca1c
humansz  1.4M
minopsz  4
maxopsz  4
lnvopsz  4
pcalign  4
mode     r-x
format   elf
iorw     false
block    0x100
type     EXEC (Executable file)
arch     arm
baddr    0x8000
binsz    1492530
bintype  elf
bits     32
canary   false
class    ELF32
compiler GCC: (GNU) 3.3.2 20031005 (Debian prerelease) GCC: (Buildroot 2012.02) 4.5.3
crypto   false
endian   little
havecode true
intrap   /lib/ld-uClibc.so.0
laddr    0x0
lang     c
linenum  false
lsyms    false
machine  ARM
nx       true
os       linux
pic      false
relocs   false
relro    no
rpath    NONE
sanitize false
static   false
stripped true
subsys   linux
va       true
```

Figure 2: extracted

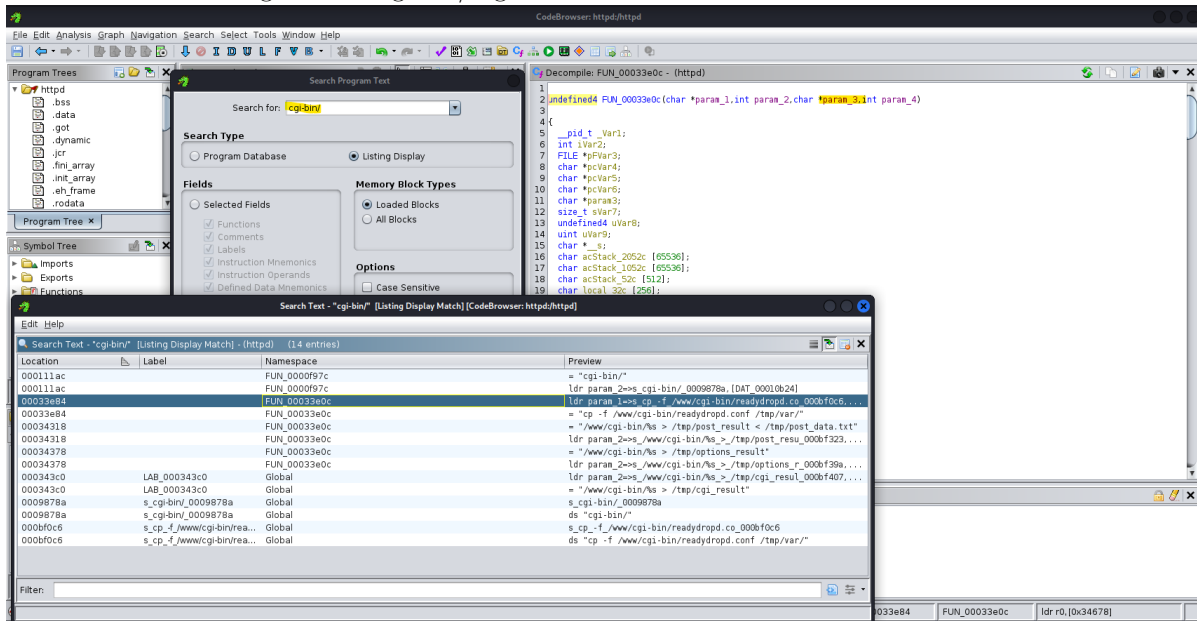
Alle Informationen scheinen gleich zu sein, lediglich die hash-Werte unterscheiden sich. Vergleicht man beide Binärdateien mit "radiff2", so sind ebenfalls wenig Unterschiede festzustellen (womöglich durch die build-Umgebung):



## httpd - Vulnerability

Nach Recherche des Exploits wurde gefunden, dass commands unter cgi-bin ausgeführt werden. Dies ist ein guter Startpunkt um die interessanten Code-Stellen zu finden.

In Ghidra kann zu Beginn nach "cgi-bin/" gesucht werden:



Betrachtet man nun die Suchergebnisse im Decompiler bekommt man ein gewisses Verständnis. Das print-statement in der obig-gezeigten Funktion lässt darauf schließen, dass "param.3" die url ist:

```

1  var_4 = &00000000;
2  if (iVar2 != 0) {
3      printf("\r\n#####%s(%d)url=%s,method=%d\r\n", "netgear_commonCgi", 0x3b, param_3, param_4)
4  }

```

Eine genaue Analyse wie "param.3" schlussendlich ausgeführt wird, wurde nicht gemacht. Ein Umbenennen und Verfolgen der Variablen in Ghidra würde sehr sicher darauf hinführen, dass ein system-Aufruf mit "acStack\_52c" durchgeführt wird. Der Wert in acStack\_52 steht in Zusammenhang mit dem input der url und wird ohne weitere Überprüfungen und Authentifizierung ausgeführt.

Systemaufruf:

```
┌  
sprintf(acStack_52c, pcVar4, acStack_ad);  
system(acStack_52c);  
memset(acStack_ad, 0, 0x40);
```

Dekompilierte Funktion:

```
1 undefined4 FUN_00033e0c(char *param_1,int param_2,char *param_3,int param_4)
2
3 {
4     __pid_t _Var1;
5     int iVar2;
6     FILE *pFVar3;
7     char *pcVar4;
8     char *pcVar5;
9     char *pcVar6;
10    char *param3;
11    size_t sVar7;
12    undefined4 uVar8;
13    uint uVar9;
14    char *__s;
15    char acStack_2052c [65536];
16    char acStack_1052c [65536];
17    char acStack_52c [512];
18    char local_32c [256];
19    char local_22c [256];
20    undefined auStack_12c [64];
21    char acStack_ec [64];
22    char acStack_ac [64];
23    undefined auStack_6c [32];
24    undefined auStack_4c [16];
25    char acStack_3c [16];
26    int aiStack_2c [2];
27
28    _Var1 = fork();
29    if (_Var1 != 0) {
30        if (0 < _Var1) {
31            waitpid(_Var1,aiStack_2c,0);
32        }
33        return 0;
34    }
35    _Var1 = fork();
36    if (_Var1 != 0) goto LAB_00034638;
37    iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
38    if (iVar2 != 0) {
39        printf("\r\n#####s(%d)url=%s,method=%d\r\n","netgear_commonCgi",0x3b,
40            param_3,param_4);
41    }
42    pFVar3 = fopen("/tmp/var/readdropd.conf","r");
43    if (pFVar3 == (FILE *)0x0) {
44        system("cp -f /www/cgi-bin/readdropd.conf /tmp/var/");
45        iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
46        if (iVar2 != 0) {
47            puts("\r\n#####copy readdropd.conf\r\n");
48        }
49    }
50    else {
51        fclose(pFVar3);
52    }
53    pcVar4 = strstr(param_3,"cgi-bin");
54    if (pcVar4 != (char *)0x0) {
55        iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
56        if (iVar2 != 0) {
57            printf("\r\n#####s(%d)\r\n","netgear_commonCgi",0x4c);
58        }
59        pcVar5 = strchr(pcVar4,0x3f);
60        if (pcVar5 == (char *)0x0) {
61            iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
62            if (iVar2 != 0) {
63                printf("\r\n#####s(%d)\r\n","netgear_commonCgi",99);
64            }
65            pcVar6 = strchr(pcVar4,0x2f);
```

```
66     __s = pcVar6 + 1;
67     param3 = strchr(__s,0x2f);
68     memset(acStack_ac,0,0x40);
69     pcVar5 = pcVar6;
70     if (pcVar6 != (char *)0x0) {
71         pcVar5 = (char *)0x1;
72     }
73     if (param3 == (char *)0x0 || pcVar6 == (char *)0x0) {
74         if (param3 == (char *)0x0) {
75             uVar9 = (uint)pcVar5 & 1;
76         }
77         else {
78             uVar9 = 0;
79         }
80         if (uVar9 != 0) {
81             strcpy(acStack_ac,__s);
82         }
83     }
84     else {
85         strncpy(acStack_ac,__s,(size_t)(param3 + (-1 - (int)pcVar6)));
86         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
87         if (iVar2 != 0) {
88             printf("\r\n#####tmp1=%s,tmp2=%s,tmp3=%s,cgi=%s\r\n",pcVar4,pcVar6,
126 param3,
127         acStack_ac);
128     }
129     pcVar4 = local_32c;
130     strcpy(pcVar4,param3);
131     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
132     if (iVar2 != 0) {
133         pcVar5 = "\r\n#####s(%d)path_info=%s\r\n";
134         uVar8 = 0x6e;
135         goto LAB_000340e8;
136     }
137 }
138 }
139 else {
140     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
141     if (iVar2 != 0) {
142         printf("\r\n#####s(%d)\r\n","netgear_commonCgi",0x50);
143     }
144     pcVar5 = strchr(pcVar4,0x3f);
145     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
146     if (iVar2 != 0) {
147         printf("\r\n#####s(%d)tmp1=%s,tmp2=%s\r\n","netgear_commonCgi",0x53,
148 pcVar4,
149         pcVar5 + 1);
150     }
151     strcpy(local_22c,pcVar5 + 1);
152     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
153     if (iVar2 != 0) {
154         printf("\r\n#####s(%d)query_string=%s\r\n","netgear_commonCgi",0x56,
155 local_22c);
156     }
157     pcVar6 = strchr(param_3,0x2f);
158     if (pcVar6 != (char *)0x0) {
159         pcVar4 = acStack_ac;
160         memset(pcVar4,0,0x40);
161         strncpy(pcVar4,pcVar6 + 1,(size_t)(pcVar5 + (-1 - (int)pcVar6)));
162         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
163         if (iVar2 != 0) {
164             pcVar5 = "\r\n#####s(%d)cgi_name=%s\r\n";
165             uVar8 = 0x5d;
166             LAB_000340e8:
167             printf(pcVar5,"netgear_commonCgi",uVar8,pcVar4);
168         }
169     }
170 }
171 }
```



```
132 if (param_4 == 0) {
133     pcVar4 = "GET";
134     LAB_00034150:
135     strcpy(acStack_3c, pcVar4);
136 }
137 else {
138     if (param_4 == 1) {
139         pcVar4 = "POST";
140         goto LAB_00034150;
141     }
142     if (param_4 == 2) {
143         pcVar4 = "OPTIONS";
144         goto LAB_00034150;
145     }
146 }
147 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
148 if (iVar2 != 0) {
149     printf("\r\n#####s(%d)request_method=%s\r\n", "netgear_commonCgi", 0x82,
150         acStack_3c);
151 }
152 if (local_32c[0] != '\0') {
153     setenv("PATH_INFO", local_32c, 1);
154 }
155 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
156 if (iVar2 != 0) {
157     pcVar4 = getenv("PATH_INFO");
158     printf("\r\n#####s(%d)PATH_INFO=%s\r\n", "netgear_commonCgi", 0x88, pcVar4);
159 }
160 setenv("LD_LIBRARY_PATH", "/usr/lib", 1);
161 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
162 if (iVar2 != 0) {
163     pcVar4 = getenv("LD_LIBRARY_PATH");
164     printf("\r\n#####s(%d)LD_LIBRARY_PATH=%s\r\n", "netgear_commonCgi", 0x8c,
165         pcVar4);
166 }
167 setenv("REQUEST_METHOD", acStack_3c, 1);
168 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
169 if (iVar2 != 0) {
170     pcVar4 = getenv("REQUEST_METHOD");
171     printf("\r\n#####s(%d)REQUEST_METHOD=%s\r\n", "netgear_commonCgi", 0x90,
172         pcVar4);
173 }
174 if (local_22c[0] != '\0') {
175     setenv("QUERY_STRING", local_22c, 1);
176 }
177 iVar2 = strcmp(acStack_3c, "POST");
178 if (iVar2 == 0) {
179     pFVar3 = fopen("/tmp/post_result", "r");
180     if (pFVar3 != (FILE *)0x0) {
181         fclose(pFVar3);
182         system("rm -f /tmp/post_result");
183         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
184         if (iVar2 != 0) {
185             puts("\r\n#####del post #####\r\n");
186         }
187     }
188     system("rm -f /tmp/post_data.txt");
189     sleep(1);
190     pFVar3 = fopen("/tmp/post_data.txt", "w");
191     if (pFVar3 != (FILE *)0x0) {
192         fputs(param_1, pFVar3);
193         fclose(pFVar3);
194     }
195     pcVar4 = "/www/cgi-bin/%s > /tmp/post_result < /tmp/post_data.txt";
196 }
197 else {
198     iVar2 = strcmp(acStack_3c, "OPTIONS");
199     if (iVar2 == 0) {
200         pFVar3 = fopen("/tmp/options_result", "r");
```

```
198     if (pFVar3 != (FILE *)0x0) {
199         fclose(pFVar3);
200         system("rm -f /tmp/options_result");
201         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
202         if (iVar2 != 0) {
203             puts("\r\n#####del option #####\r\n");
204         }
205     }
206     pcVar4 = "/www/cgi-bin/%s > /tmp/options_result";
207 }
208 else {
209     pFVar3 = fopen("/tmp/cgi_result", "r");
210     if (pFVar3 != (FILE *)0x0) {
211         fclose(pFVar3);
212         system("rm -f /tmp/cgi_result");
213         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
214         if (iVar2 != 0) {
215             puts("\r\n#####delete /tmp/cgi_result #####\r\n");
216         }
217     }
218     pcVar4 = "/www/cgi-bin/%s > /tmp/cgi_result";
219 }
220 }
221 sprintf(acStack_52c, pcVar4, acStack_ac);
222 system(acStack_52c);
223 memset(acStack_ec, 0, 0x40);
224 memset(auStack_12c, 0, 0x40);
225 memset(auStack_6c, 0, 0x20);
226 memset(auStack_4c, 0, 0x10);
227 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "1");
228 if (iVar2 != 0) {
229     printf("\r\n#####%s(%d)\r\n", "netgear_commonCgi", 200);
230 }
231 iVar2 = strcmp(acStack_3c, "POST");
232 if (iVar2 == 0) {
233     pcVar4 = "/tmp/post_result";
234 }
235 else {
236     iVar2 = strcmp(acStack_3c, "OPTIONS");
237     if (iVar2 == 0) {
238         pcVar4 = "/tmp/options_result";
239     }
240     else {
241         pcVar4 = "/tmp/cgi_result";
242     }
243 }
244 pFVar3 = fopen(pcVar4, "r");
245 if (pFVar3 != (FILE *)0x0) {
246     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "1");
247     if (iVar2 != 0) {
248         printf("\r\n#####%s(%d)\r\n", "netgear_commonCgi", 0xd3);
249     }
250     while (pcVar4 = fgets(acStack_1052c, 0xffff, pFVar3), pcVar4 != (char *)0x0) {
251         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "1");
252         if (iVar2 != 0) {
253             printf("\r\n#####%s(%d)\r\n", "netgear_commonCgi", 0xd7);
254         }
255         pcVar4 = strstr(acStack_1052c, "Status:");
256         if (pcVar4 == (char *)0x0) {
257             strcat(acStack_2052c, acStack_1052c);
258         }
259         else {
260             strcpy(acStack_ec, pcVar4 + 7);
261             pcVar4 = strchr(acStack_ec, 10);
262             if (pcVar4 != (char *)0x0) {
263                 *pcVar4 = '\0';
264             }
265             iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
266             if (iVar2 != 0) {
```

```
267     printf("\r\n#####s(%d)status=%s\r\n", "netgear_commonCgi", 0xdf,
268     acStack_ec);
269     sprintf(acStack_2052c, "HTTP/1.1%s\r\n", acStack_ec);
270     }
271     }
272     fclose(pFVar3);
273 }
274 strcat(acStack_2052c, "\r\n");
275 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "1");
276 if (iVar2 != 0) {
277     printf("\r\n#####s(%d)http_hdr=%s\r\n", "netgear_commonCgi", 0x114,
278     acStack_2052c);
279 }
280 sVar7 = strlen(acStack_2052c);
281 FUN_0000f27c(param_2, acStack_2052c, sVar7);
282 iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg", "2");
283 if (iVar2 != 0) {
284     printf("\r\n#####s(%d)\r\n", "netgear_commonCgi", 0x118);
285 }
286 LAB_00034638:
287 /* WARNING: Subroutine does not return */
288 exit(0);
289 }
```