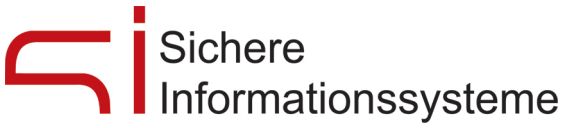


Reverse Engineering (REV3)

UE 05 – Firmware Analyse – Protokoll

Jakob Mayr

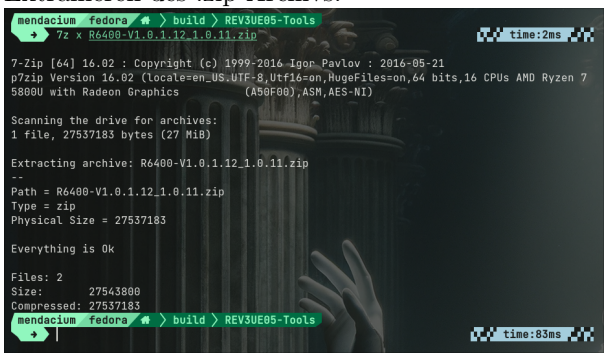
WS 2023/2024

File-Beschaffung

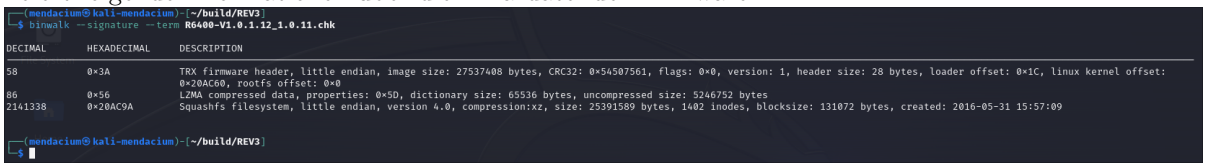
Die benötigte Firmware-Version kann direkt auf der netgear-Seite heruntergeladen werden:
`https://www.downloads.netgear.com/files/GDC/R6400/R6400-V1.0.1.12_1.0.11.zip`

Analyse Teilkomponenten

Extrahieren des .zip-Archivs:



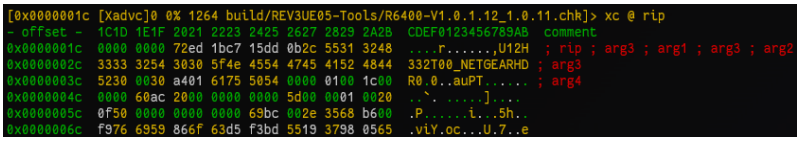
Die Ausführung des Befehls `binwalk` mit der Option `--signature` auf die Datei `r6400-V1.0.1.12_1.0.11.chk` liefert folgende Informationen über die Binärdatei der Firmware:



- **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:

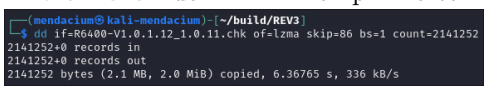
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|                      magic number ('HDR0')                    |
+---------------------------------------------------------------+
|                   length (header size + data)                 |
+---------------+---------------+-------------------------------+
|                       32-bit CRC value                        |
+---------------+---------------+-------------------------------+
|           TRX flags           |          TRX version          |
+-------------------------------+-------------------------------+
|                       Partition offset[0]                     |
+---------------------------------------------------------------+
|                       Partition offset[1]                     |
+---------------------------------------------------------------+
|                       Partition offset[2]                     |
+---------------------------------------------------------------+
```

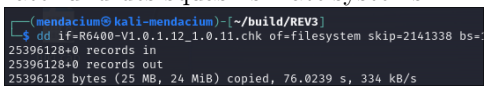
  Das Firmware-Image hat nicht wirklich einen Bootloader im klassischen Sinne. Der "loader offset" zeigt auf **0x1C** und es handelt sich dabei um einen LZMA-Loader, dieser startet einen selbstentpackenden Kernel und somit die Firmware. Betrachtet man die Stelle **0x1C** ("loader offset"), so findet man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



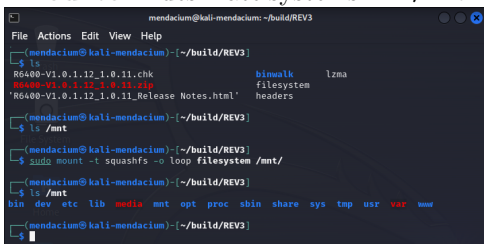
- **LZMA komprimierte Daten:** Der LZMA-komprimierte Teil (Lempel-Ziv-Markov-Kettenalgorithmus) im Header gibt Informationen über die Wortbuchgröße (65536 Bytes) und die unkomprimierte Datengröße des LZMA-Teils (5246752 Bytes). Die Daten sind vermutlich ausführerbarer Code für einen Kernel, welche zuerst 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:

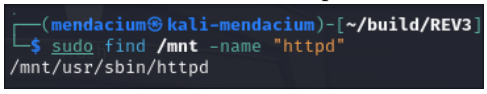




"**Mounten**" des Dateisystems in "`/mnt`":



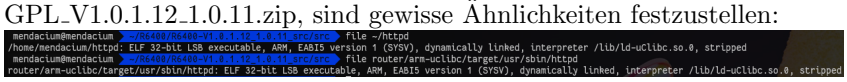
Suchen und finden der "httpd"-Binärdatei:



httpd

Informationen (radare2) über die Binärdatei:

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:



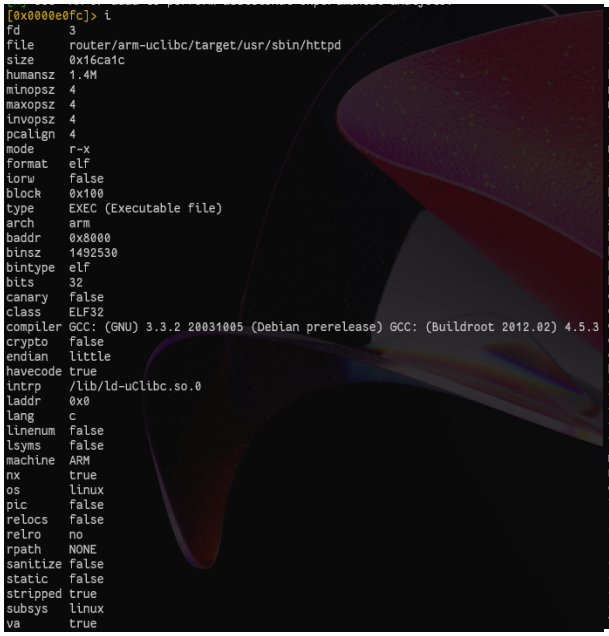


Figure 1: downloaded

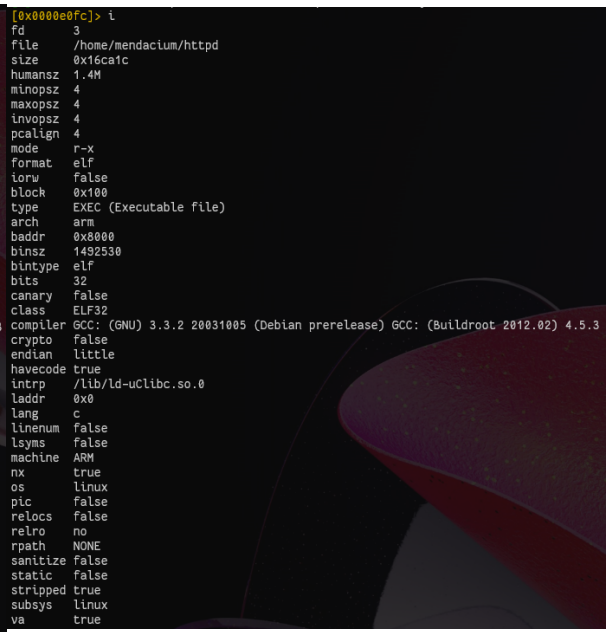
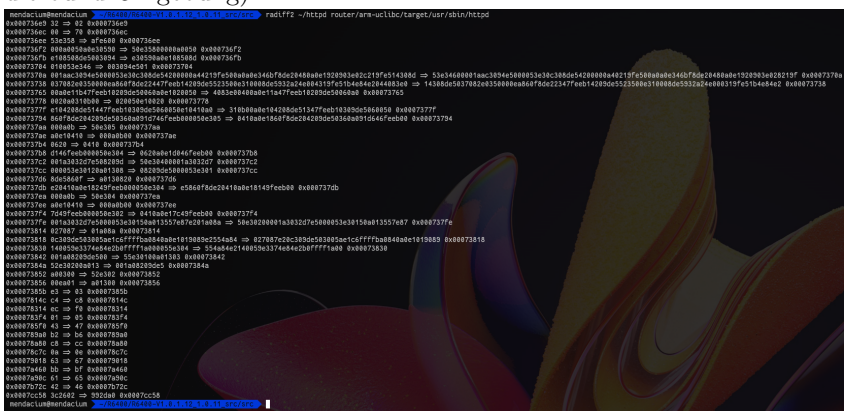


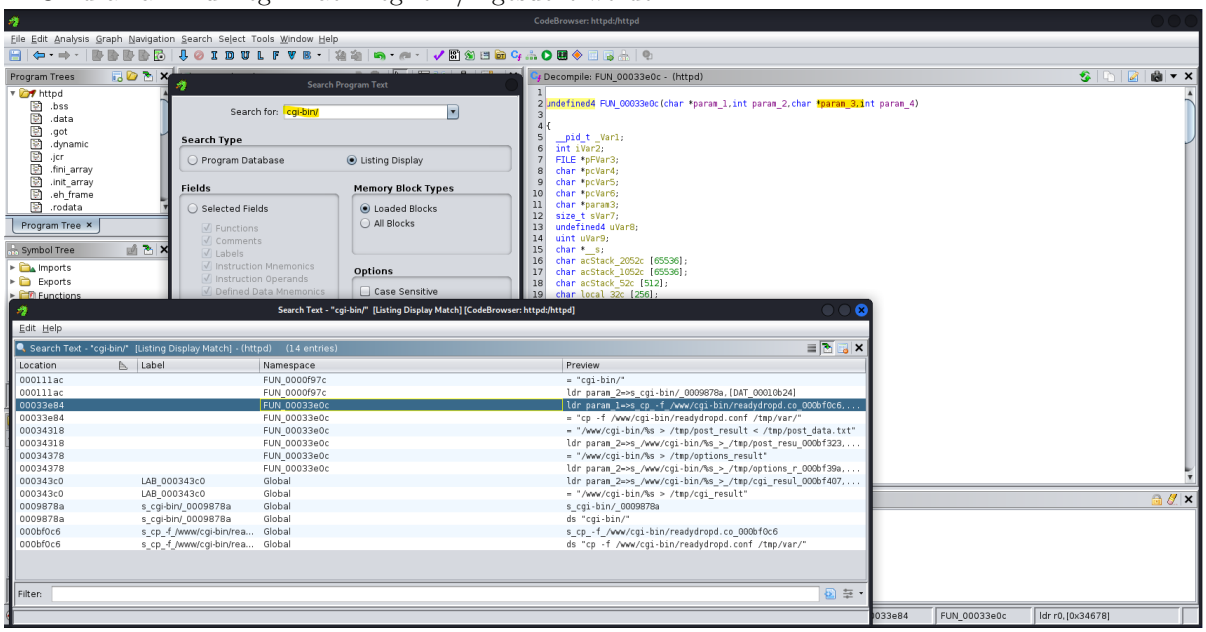
Figure 2: extracted

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 Begin nach "cgi-bin/" gesucht werden:



Betrachtet man nun die Sucherergebnisse im Decompiler bekommt man ein gewisses Verständnis. Das print-statement in der obig-gezeigten Funktion lässt darauf schließen, dass "`param_3`" die url ist:

```
if (iVar2 != 0) {
  printf("\r\n###############%s(%d)url=%s,method=%d\r\n","netgear_commonCgi",0x3b,param_3,param_4)
  ;
}
```

Eine genaue Analyse wie "`param_3`" schluss-endlich 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_ac);
system(acStack_52c);
```

Dekompilierte Funktion:

```
undefined4 FUN_00033e0c(char *param_1,int param_2,char *param_3,int param_4)

{
  __pid_t _Var1;
  int iVar2;
  FILE *pFVar3;
  char *pcVar4;
  char *pcVar5;
  char *pcVar6;
  char *param3;
  size_t sVar7;
  undefined4 uVar8;
  uint uVar9;
  char *__s;
  char acStack_2052c [65536];
  char acStack_1052c [65536];
  char acStack_52c [512];
  char local_32c [256];
  char local_22c [256];
  undefined auStack_12c [64];
  char acStack_ec [64];
  char acStack_ac [64];
  undefined auStack_6c [32];
  undefined auStack_4c [16];
  char acStack_3c [16];
  int aiStack_2c [2];

  _Var1 = fork();
  if (_Var1 != 0) {
    if (0 < _Var1) {
      waitpid(_Var1,aiStack_2c,0);
    }
    return 0;
  }
  _Var1 = fork();
  if (_Var1 != 0) goto LAB_00034638;
  iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
  if (iVar2 != 0) {
    printf("\r\n################%s(%d)url=%s,method=%d\r\n","netgear_commonCgi",0x3b,
    param_3,param_4)
    ;
  }
  pFVar3 = fopen("/tmp/var/readydropd.conf","r");
  if (pFVar3 == (FILE *)0x0) {
    system("cp -f /www/cgi-bin/readydropd.conf /tmp/var/");
    iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
    if (iVar2 != 0) {
      puts("\r\n####################copy readydropd.conf\r");
    }
  }
  else {
    fclose(pFVar3);
  }
  pcVar4 = strstr(param_3,"cgi-bin");
  if (pcVar4 != (char *)0x0) {
    iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
    if (iVar2 != 0) {
      printf("\r\n###########%s(%d)\r\n","netgear_commonCgi",0x4c);
    }
    pcVar5 = strchr(pcVar4,0x3f);
    if (pcVar5 == (char *)0x0) {
      iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
      if (iVar2 != 0) {
        printf("\r\n###########%s(%d)\r\n","netgear_commonCgi",99);
      }
      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,
      param3,
89                   acStack_ac);
90          }
91          pcVar4 = local_32c;
92          strcpy(pcVar4,param3);
93          iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
94          if (iVar2 != 0) {
95            pcVar5 = "\r\n################%s(%d)path_info=%s\r\n";
96            uVar8 = 0x6e;
97            goto LAB_000340e8;
98          }
99        }
100     }
101     else {
102       iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
103       if (iVar2 != 0) {
104         printf("\r\n##########%s(%d)\r\n","netgear_commonCgi",0x50);
105       }
106       pcVar5 = strchr(pcVar4,0x3f);
107       iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","1");
108       if (iVar2 != 0) {
109         printf("\r\n##############%s(%d)tmp1=%s,tmp2=%s\r\n","netgear_commonCgi",0x53,
      pcVar4,
110              pcVar5 + 1);
111       }
112       strcpy(local_22c,pcVar5 + 1);
113       iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
114       if (iVar2 != 0) {
115         printf("\r\n################%s(%d)query_string=%s\r\n","netgear_commonCgi",0x56,
      local_22c);
116       }
117       pcVar6 = strchr(param_3,0x2f);
118       if (pcVar6 != (char *)0x0) {
119         pcVar4 = acStack_ac;
120         memset(pcVar4,0,0x40);
121         strncpy(pcVar4,pcVar6 + 1,(size_t)(pcVar5 + (-1 - (int)pcVar6)));
122         iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
123         if (iVar2 != 0) {
124           pcVar5 = "\r\n################%s(%d)cgi_name=%s\r\n";
125           uVar8 = 0x5d;
126           LAB_000340e8:
127           printf(pcVar5,"netgear_commonCgi",uVar8,pcVar4);
128         }
129       }
130     }
131   }
```

```
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,
      acStack_3c);
150   }
151   if (local_32c[0] != '\0') {
152     setenv("PATH_INFO",local_32c,1);
153   }
154   iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
155   if (iVar2 != 0) {
156     pcVar4 = getenv("PATH_INFO");
157     printf("\r\n################%s(%d)PATH_INFO=%s\r\n","netgear_commonCgi",0x88,pcVar4);
158   }
159   setenv("LD_LIBRARY_PATH","/usr/lib",1);
160   iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
161   if (iVar2 != 0) {
162     pcVar4 = getenv("LD_LIBRARY_PATH");
163     printf("\r\n################%s(%d)LD_LIBRARY_PATH=%s\r\n","netgear_commonCgi",0x8c,
      pcVar4);
164   }
165   setenv("REQUEST_METHOD",acStack_3c,1);
166   iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
167   if (iVar2 != 0) {
168     pcVar4 = getenv("REQUEST_METHOD");
169     printf("\r\n################%s(%d)REQUEST_METHOD=%s\r\n","netgear_commonCgi",0x90,
      pcVar4);
170   }
171   if (local_22c[0] != '\0') {
172     setenv("QUERY_STRING",local_22c,1);
173   }
174   iVar2 = strcmp(acStack_3c,"POST");
175   if (iVar2 == 0) {
176     pFVar3 = fopen("/tmp/post_result","r");
177     if (pFVar3 != (FILE *)0x0) {
178       fclose(pFVar3);
179       system("rm -f /tmp/post_result");
180       iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
181       if (iVar2 != 0) {
182         puts("\r\n###########del post ############\r");
183       }
184     }
185     system("rm -f /tmp/post_data.txt");
186     sleep(1);
187     pFVar3 = fopen("/tmp/post_data.txt","w");
188     if (pFVar3 != (FILE *)0x0) {
189       fputs(param_1,pFVar3);
190       fclose(pFVar3);
191     }
192     pcVar4 = "/www/cgi-bin/%s > /tmp/post_result < /tmp/post_data.txt";
193   }
194   else {
195     iVar2 = strcmp(acStack_3c,"OPTIONS");
196     if (iVar2 == 0) {
197       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");
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");
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,
      acStack_ec);
268           }
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,
      acStack_2052c);
278     }
279     sVar7 = strlen(acStack_2052c);
280     FUN_0000f27c(param_2,acStack_2052c,sVar7);
281     iVar2 = acosNvramConfig_match("ntgr_cgi_debug_msg","2");
282     if (iVar2 != 0) {
283       printf("\r\n######==========######%s(%d)\r\n","netgear_commonCgi",0x118);
284     }
285     LAB_00034638:
286     /* WARNING: Subroutine does not return */
287     exit(0);
288 }
```