Vulnerability Research: FOSCAM FI9816P Security IP-Camera

Fontys HOGESCHOOL ICT

Javier Duran

Student Number: 3567885

Cyber Security Specialization

Table of Contents

Research Question

What are the possible security flaws/vulnerabilities that can be exploited by an attacker to perform other malicious attacks, such as espionage or using the product for further attacks, in the FOSCAM FI9816P security IP-Camera as it comes at the time of purchase?

Introduction

In the last years, there has been an increase in the usage of IP cameras worldwide, which has come hand in hand with the arise of smartphones and IoT devices. The key benefits of these cameras are remote video surveillance, intruder alarms, access control, and perimeter surveillance. These characteristics made the IP cameras very popular in buildings and residential areas, because of the ease that it gives by centralizing all cameras into one application and being able to monitor different sections of location from one application and place. Although, IP cameras can be very helpful, time saving and can give the ability to monitor an area remotely, security is a big concern when it comes to choosing the correct IP camera. In this research, the studied device will be FOSCAM FI9816P IP-camera. FOSCAM is a Chinese leading professional high-tech company which provides IP video camera and solutions.

Goal

The goal of this research is to analyze and research about the behavior, functionality and possible security flaws and vulnerabilities, in the FOSCAM FI9816P IP-camera, that might be exploited by an attacker to perform malicious activities such espionage, bridging into the network, gather information for further attacks, etc. This will be done to inform about the possible or proved vulnerabilities in an IoT product. In addition, different tools and techniques will be used to gather information that will be of importance for the goal of this project. Many of the companies release updates for firmware of the cameras, but some are still sold with the older firmware which makes these devices vulnerable to past known exploits.

Objectives

- Improve professional research skills.
- Research about vulnerabilities or security flaws in a product of choice.
- Put into practice different tools and knowledge to gather information about the product.

1. Testing Environment

In this section, a short explanation of the working environment will be explained. Due to the time scope of this research, I will test the camera mostly via ethernet connection, but will also do some testing with the camera connected via wireless. The environment will be made up of a router with the camera connected via ethernet connection. The computer used to research and test for vulnerabilities in the device will be sitting in the same network as the camera. Some tests will be done via ethernet connection and other via WIFI connection. In Figure 1.1 you can observe a visual representation of the network.

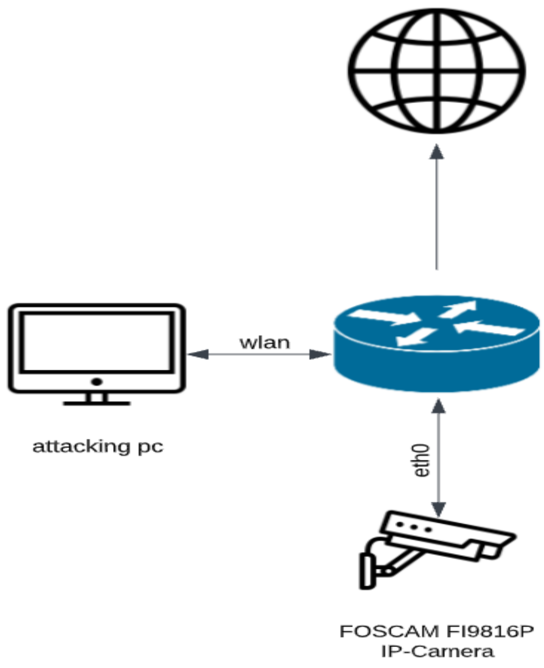


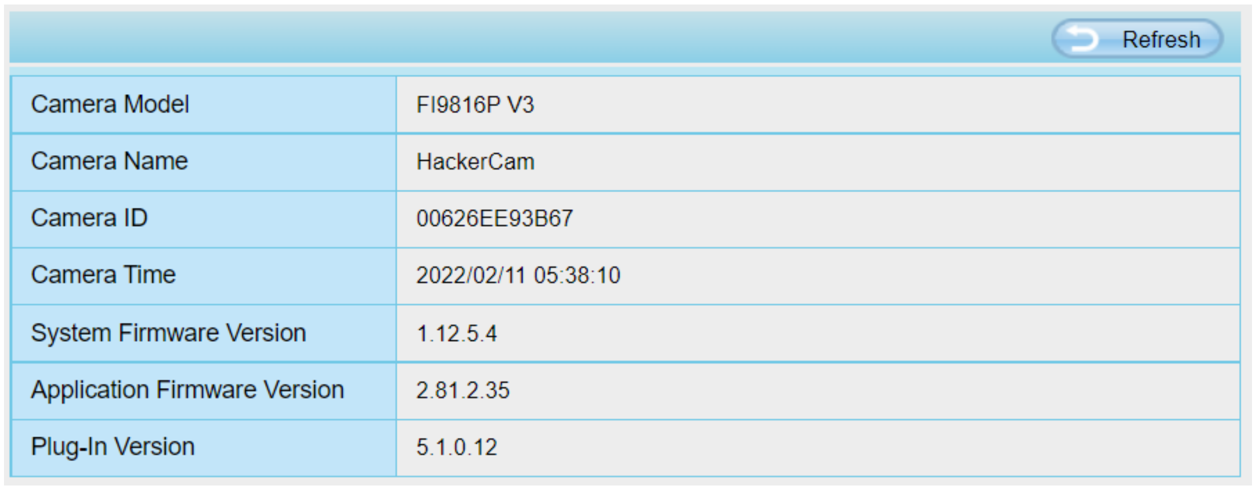
Figure 0.1

2. Product Description

The product to be tested is the Foscam FI9816P. This is an IP camera; this means that is connected to a network and will stream the video into some application or GUI to the client. This will give a camera an IP and will work as an IoT device connected to the network. From the official website of FOSCAM it is also interesting to know that the camera has a built-in webserver and can upload files to FTP servers (FOSCAM).

2.1 Product Specifications

The product will be researched with the firmware and default settings that come with the camera from the moment of purchasing. In Figure 2.1, you can see a screenshot of the specifications which It would be tested with.



Camera Model	FI9816P V3
Camera Name	HackerCam
Camera ID	00626EE93B67
Camera Time	2022/02/11 05:38:10
System Firmware Version	1.12.5.4
Application Firmware Version	2.81.2.35
Plug-In Version	5.1.0.12

Figure 2.1.1

3. Gathering Information

To perform this research, I will be using 3 different research methods related to the DOT framework. The three methods used will be the library method to gain background knowledge on the product. Next, I will use scanning and enumeration tools as lab research to know more about the actual product and find information that is not given by the manufacturer or previously published information. Finally, I will use the workshop method to test different techniques and make use of tools to find security flaws while learning more on how the product works.

Research Method	**Tools used and Information Gathered**
Library	• Known CVE's • Product Manual • Past research on product
Lab	• Nmap • Gobuster • Nikto • Plug-In analysis • Wireshark
Workshop	• Burpsuite • XSS attempts on Web App

3.1 Setup

Very easy setup, only plugged-in power and ethernet cable and worked automatically. I visited the webserver of the camera to check for the default password and username. To begin they have default password and username "admin", and password is blank but once the user logs in with the default credentials, the application requires a username and password change. In addition, to set up the wireless connection, once logged in you just need to click on the quick set up and the camera is connected wireless. Multiple users can be created, and the logs of the events can be checked also in the web server application deployed by the camera.

3.2 Past Known CVE's

- CVE-2018-6832
- CVE-2018-6831
- CVE-2018-6830

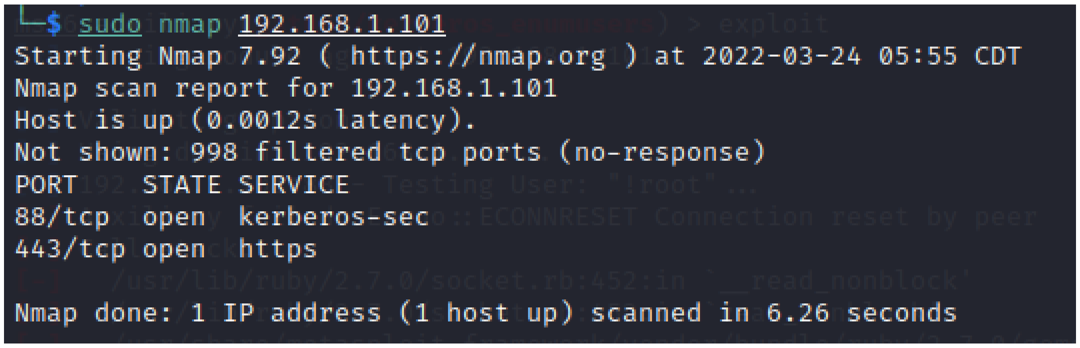
To learn more about these specific vulnerabilities, refer to: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Foscam+FI9816P+IP-camera%5C

3.3 Scanning and Enumerating

In this section, I will explain all the steps taken and the tools used to gather information about the camera for further analysis and vulnerability analysis in the camera, plugin, and web application. As a first step, I ran a simple Nmap scan in the IP of the camera. The command used was:

- sudo nmap 192.168.1.101

as seen in Figure 3.3.1. The results only showed port 88 and port 443 open. The services in the camera where Kerberos and https.



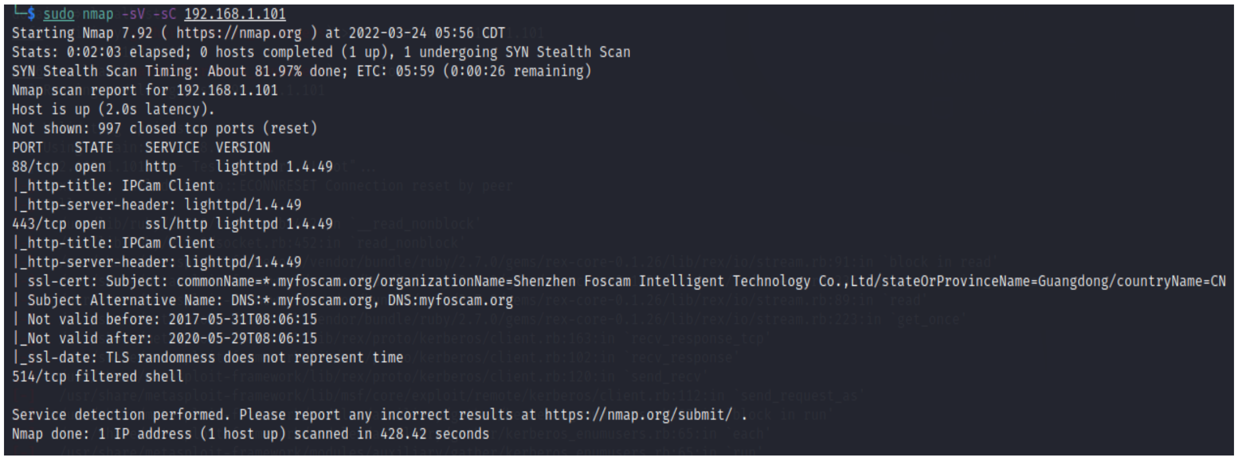
```
└─$ sudo nmap 192.168.1.101
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-24 05:55 CDT
Nmap scan report for 192.168.1.101
Host is up (0.0012s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
88/tcp  open  kerberos-sec
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 6.26 seconds
```

Figure 3.3.1

To get more information on the services running in the camera I used another type of Nmap scan to learn more about the service version and the service certificates that are running in the FOSCAM camera.

- sudo nmap -sV -sC 192.168.1.101

The results can be seen in Figure 3.3.2, both ports are running a lighttpd 1.4.49 server, the only difference is that port 443 has the SSL certificates. In addition, this scan shows a different result because it also showed port 514 as a filtered shell port. Doing some research, I also find out that FOSCAM might use RTSP protocol which is used by some cameras as a streaming protocol when connected via wireless.

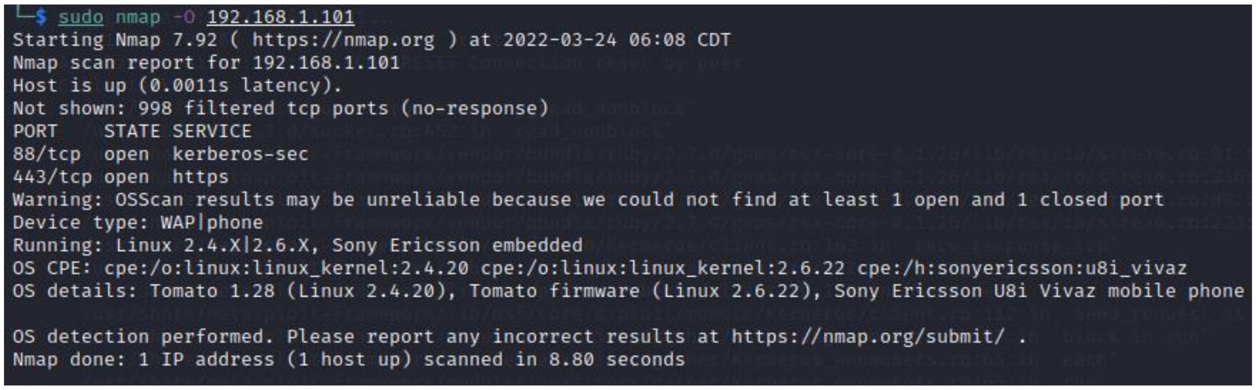


```
└─$ sudo nmap -sV -sC 192.168.1.101
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-24 05:56 CDT
Stats: 0:02:03 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 81.97% done; ETC: 05:59 (0:00:26 remaining)
Nmap scan report for 192.168.1.101
Host is up (2.0s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE    SERVICE  VERSION
88/tcp  open     http     lighttpd 1.4.49
|_http-title: IPCam Client
|_http-server-header: lighttpd/1.4.49
443/tcp open     ssl/http lighttpd 1.4.49
|_http-title: IPCam Client
|_http-server-header: lighttpd/1.4.49
| ssl-cert: Subject: commonName=*.myfoscam.org/organizationName=Shenzhen Foscam Intelligent Technology Co.,Ltd/stateOrProvinceName=Guangdong/countryName=CN
| Subject Alternative Name: DNS:*.myfoscam.org, DNS:myfoscam.org
| Not valid before: 2017-05-31T08:06:15
|_Not valid after:  2020-05-29T08:06:15
|_ssl-date: TLS randomness does not represent time
514/tcp filtered shell

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 428.42 seconds
```

Figure 3.3.2

The last Nmap scan done was the OS scan in the camera to check what operating system it is running in the camera. The camera is running a Linux 2.4, Sony Ericsson embedded operating system as seen in Figure 3.3.3. The scan was performed with the following command:

- sudo nmap -O 192.168.1.101



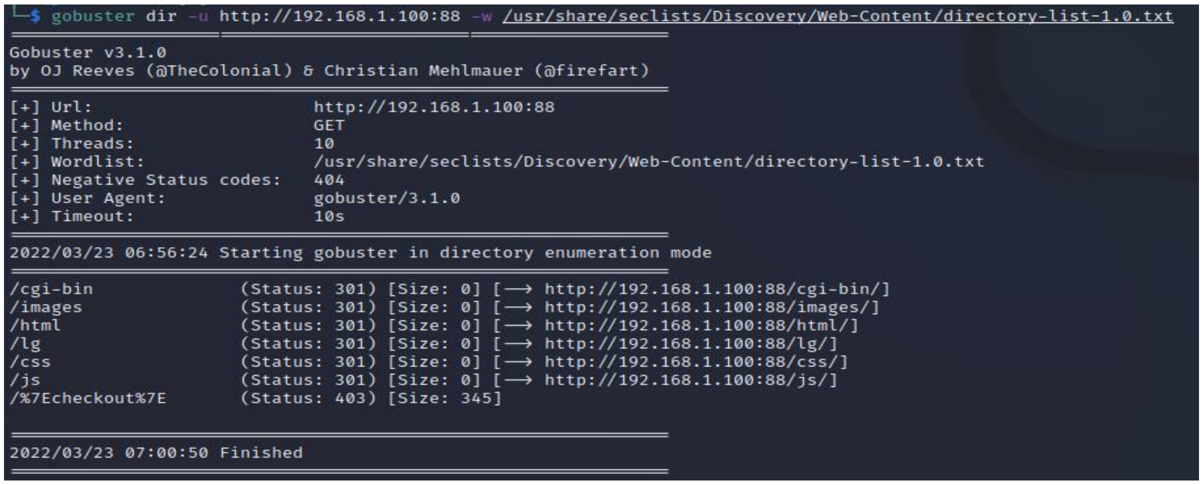
```
└─$ sudo nmap -O 192.168.1.101
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-24 06:08 CDT
Nmap scan report for 192.168.1.101
Host is up (0.0011s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
88/tcp  open  kerberos-sec
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: WAP|phone
Running: Linux 2.4.X|2.6.X, Sony Ericsson embedded
OS CPE: cpe:/o:linux:linux_kernel:2.4.20 cpe:/o:linux:linux_kernel:2.6.22 cpe:/h:sonyericsson:u8i_vivaz
OS details: Tomato 1.28 (Linux 2.4.20), Tomato firmware (Linux 2.6.22), Sony Ericsson U8i Vivaz mobile phone

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.80 seconds
```

Figure 3.3.3

As I know there is a web server in the camera, I used Gobuster to brute force the URIs of a website, which are the directories and files in the webserver. The brute force of the URIs gave us some directories that can be interesting such as cgi-bin, js and lg directories. The Gobuster command used was:

- gobuster dir -u http://192.168.1.100:88 -w /usr/share/seclists/Discovery/Web-Content/directory-list-1.0.txt

The command tells Gobuster to brute force the directories on the http server and port represented by the “-u” flag. The “-w” flag is to give the path for the wordlist that is going to be used to brute force the directories. As seen in Figure 3.3.4 the results also show the status of each directory. The http status 301 means that the file has been moved permanently. When checking in these directories, I just run into a forbidden page.



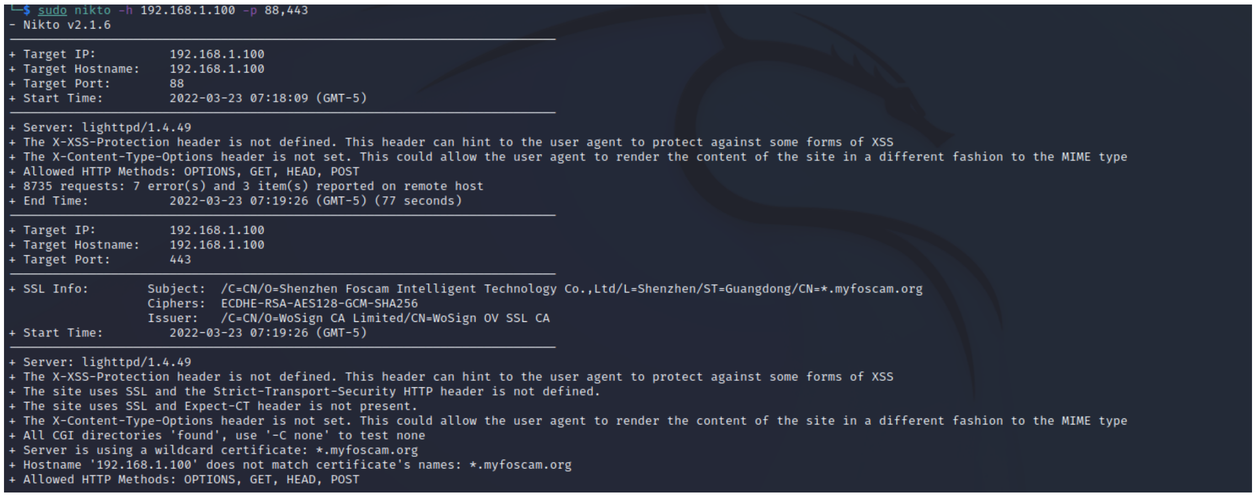
```
└─$ gobuster dir -u http://192.168.1.100:88 -w /usr/share/seclists/Discovery/Web-Content/directory-list-1.0.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.100:88
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/directory-list-1.0.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/03/23 06:56:24 Starting gobuster in directory enumeration mode
===============================================================
/cgi-bin              (Status: 301) [Size: 0] [→ http://192.168.1.100:88/cgi-bin/]
/images               (Status: 301) [Size: 0] [→ http://192.168.1.100:88/images/]
/html                 (Status: 301) [Size: 0] [→ http://192.168.1.100:88/html/]
/lg                   (Status: 301) [Size: 0] [→ http://192.168.1.100:88/lg/]
/css                  (Status: 301) [Size: 0] [→ http://192.168.1.100:88/css/]
/js                   (Status: 301) [Size: 0] [→ http://192.168.1.100:88/js/]
/%7Echeckout%7E       (Status: 403) [Size: 345]

===============================================================
2022/03/23 07:00:50 Finished
===============================================================
```

Figure 3.3.4

To get further information on the Lighttpd servers I used Nikto. Nikto is a tool that is used to scan a web server for a vulnerability that can be exploited and can compromise the server. In this case I will use Nikto to scan on ports 88 and 443 since those are the ports running the web servers for the camera. To get the output I used the command:

- sudo nikto -h 192.168.1.100 -p 88,443

The results for port 88 show some hints that there might be some type of XSS vulnerability in the server since the X-XSS-Protection headers are not defined. Similarly, port 443 shows the same lack of protection against some XSS attacks, but also contains some extra information such as the CGI directories and the use of SSL but lacks some definition of headers.



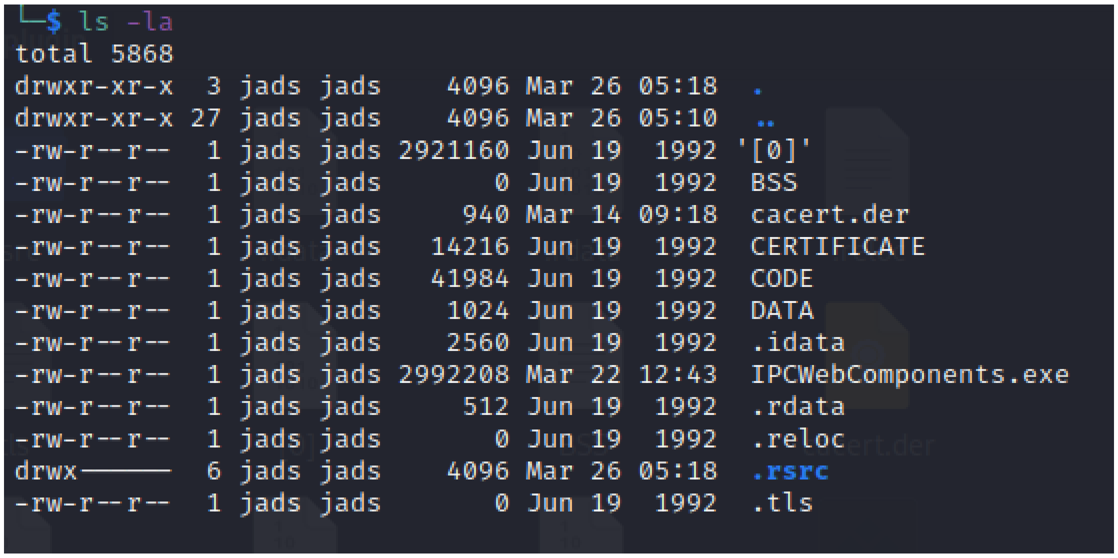
```
└─$ sudo nikto -h 192.168.1.100 -p 88,443
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.100
+ Target Hostname:    192.168.1.100
+ Target Port:        88
+ Start Time:         2022-03-23 07:18:09 (GMT-5)
---------------------------------------------------------------------------
+ Server: lighttpd/1.4.49
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ 8735 requests: 7 error(s) and 3 item(s) reported on remote host
+ End Time:           2022-03-23 07:19:26 (GMT-5) (77 seconds)
---------------------------------------------------------------------------
+ Target IP:          192.168.1.100
+ Target Hostname:    192.168.1.100
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /C=CN/O=Shenzhen Foscam Intelligent Technology Co.,Ltd/L=Shenzhen/ST=Guangdong/CN=*.myfoscam.org
                   Ciphers:  ECDHE-RSA-AES128-GCM-SHA256
                   Issuer:   /C=CN/O=WoSign CA Limited/CN=WoSign OV SSL CA
+ Start Time:         2022-03-23 07:19:26 (GMT-5)
---------------------------------------------------------------------------
+ Server: lighttpd/1.4.49
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ The site uses SSL and Expect-CT header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ All CGI directories 'found', use '-C none' to test none
+ Server is using a wildcard certificate: *.myfoscam.org
+ Hostname '192.168.1.100' does not match certificate's names: *.myfoscam.org
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
```

Figure 3.3.5

Next, I downloaded the web server plugin for Windows, which is the only supported operating system for this plugin, for this reason the plugin is a .exe file. These files are compressed binary files that will contain the code of the plugin and instructions that will operate with the web server application from the camera and the client computer. I extracted the files to give it a further look by using the 7z tool, and running the following command:

- 7z x IPCWebComponents.exe

Some of the files extracted are binary encoded, while others where hidden files and directories as shown in Figure 3.3.6.

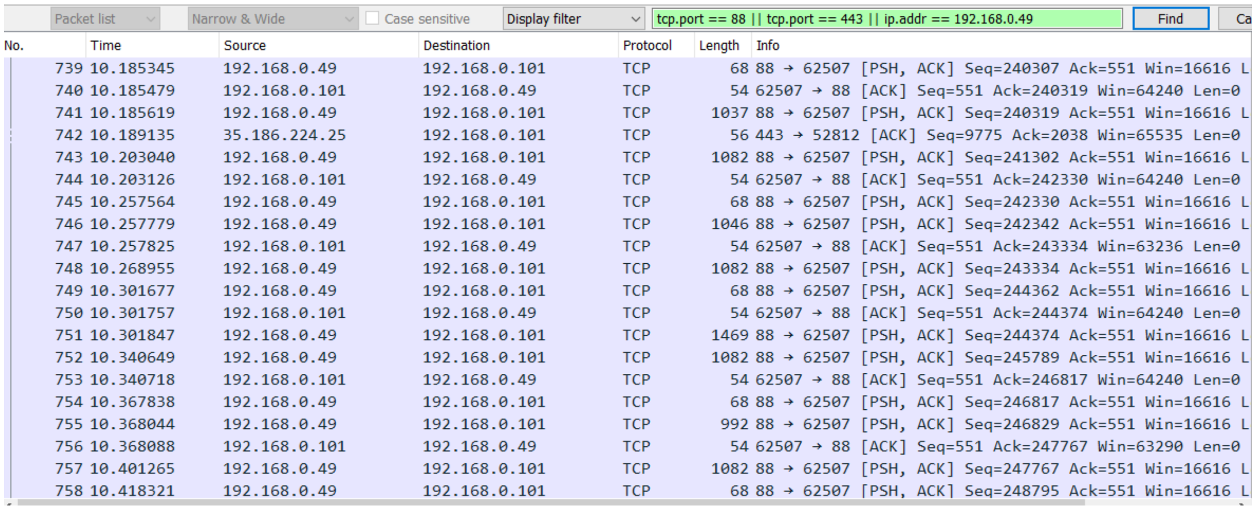


```
└─$ ls -la
total 5868
drwxr-xr-x  3 jads jads    4096 Mar 26 05:18  .
drwxr-xr-x 27 jads jads    4096 Mar 26 05:10  ..
-rw-r--r--  1 jads jads 2921160 Jun 19  1992 '[0]'
-rw-r--r--  1 jads jads       0 Jun 19  1992  BSS
-rw-r--r--  1 jads jads     940 Mar 14 09:18  cacert.der
-rw-r--r--  1 jads jads   14216 Jun 19  1992  CERTIFICATE
-rw-r--r--  1 jads jads   41984 Jun 19  1992  CODE
-rw-r--r--  1 jads jads    1024 Jun 19  1992  DATA
-rw-r--r--  1 jads jads    2560 Jun 19  1992  .idata
-rw-r--r--  1 jads jads 2992208 Mar 22 12:43  IPCWebComponents.exe
-rw-r--r--  1 jads jads     512 Jun 19  1992  .rdata
-rw-r--r--  1 jads jads       0 Jun 19  1992  .reloc
drwx------  6 jads jads    4096 Mar 26 05:18  .rsrc
-rw-r--r--  1 jads jads       0 Jun 19  1992  .tls
```

Figure 3.3.6

In Figure 3.3.6 some of the most interesting files in the plug-in files are the .rsrc directory, the CODE file, the DATA file and the .idata file but all are binary files, after trying some tools such as string and hexdump I just managed to extract certain sections of the file to become readable by a human.

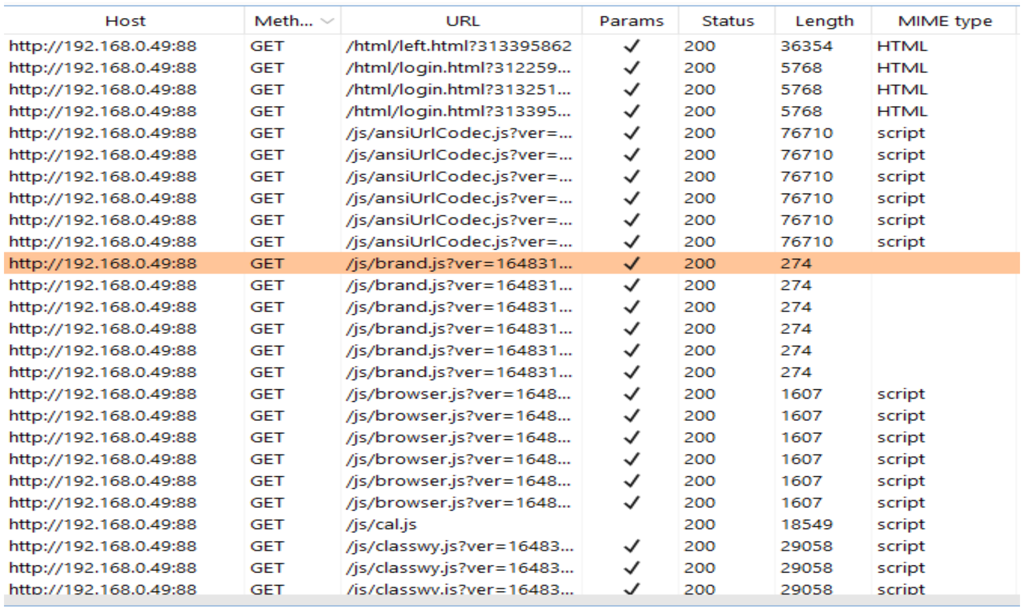
Next, I used Wireshark to monitor the traffic of the camera in the network. The camera streams TCP packets constantly, depending on the resolution of the camera. To find the interesting packets for me. I filtered for the IP and the ports found through the Nmap scans to get the desired packets. In Figure 3.3.7 you can see some of the packets filtered which have flags like PSH and ACK. The PSH (push) flag indicates that the incoming data should be passed on directly to the application instead of getting buffered. In the other hand, ACK (acknowledgment) flag is used to confirm that the data packets have been received, also used to confirm the initiation request (Howtouselinux).



```
tcp.port == 88 || tcp.port == 443 || ip.addr == 192.168.0.49

No.   Time       Source         Destination    Protocol Length Info
  739 10.185345  192.168.0.49   192.168.0.101  TCP        68 88 → 62507 [PSH, ACK] Seq=240307 Ack=551 Win=16616 L
  740 10.185479  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=240319 Win=64240 Len=0
  741 10.185619  192.168.0.49   192.168.0.101  TCP      1037 88 → 62507 [PSH, ACK] Seq=240319 Ack=551 Win=16616 L
  742 10.189135  35.186.224.25  192.168.0.101  TCP        56 443 → 52812 [ACK] Seq=9775 Ack=2038 Win=65535 Len=0
  743 10.203040  192.168.0.49   192.168.0.101  TCP      1082 88 → 62507 [PSH, ACK] Seq=241302 Ack=551 Win=16616 L
  744 10.203126  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=242330 Win=64240 Len=0
  745 10.257564  192.168.0.49   192.168.0.101  TCP        68 88 → 62507 [PSH, ACK] Seq=242330 Ack=551 Win=16616 L
  746 10.257779  192.168.0.49   192.168.0.101  TCP      1046 88 → 62507 [PSH, ACK] Seq=242342 Ack=551 Win=16616 L
  747 10.257825  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=243334 Win=63236 Len=0
  748 10.268955  192.168.0.49   192.168.0.101  TCP      1082 88 → 62507 [PSH, ACK] Seq=243334 Ack=551 Win=16616 L
  749 10.301677  192.168.0.49   192.168.0.101  TCP        68 88 → 62507 [PSH, ACK] Seq=244362 Ack=551 Win=16616 L
  750 10.301757  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=244374 Win=64240 Len=0
  751 10.301847  192.168.0.49   192.168.0.101  TCP      1469 88 → 62507 [PSH, ACK] Seq=244374 Ack=551 Win=16616 L
  752 10.340649  192.168.0.49   192.168.0.101  TCP      1082 88 → 62507 [PSH, ACK] Seq=245789 Ack=551 Win=16616 L
  753 10.340718  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=246817 Win=64240 Len=0
  754 10.367838  192.168.0.49   192.168.0.101  TCP        68 88 → 62507 [PSH, ACK] Seq=246817 Ack=551 Win=16616 L
  755 10.368044  192.168.0.49   192.168.0.101  TCP       992 88 → 62507 [PSH, ACK] Seq=246829 Ack=551 Win=16616 L
  756 10.368088  192.168.0.101  192.168.0.49   TCP        54 62507 → 88 [ACK] Seq=551 Ack=247767 Win=63290 Len=0
  757 10.401265  192.168.0.49   192.168.0.101  TCP      1082 88 → 62507 [PSH, ACK] Seq=247767 Ack=551 Win=16616 L
  758 10.418321  192.168.0.49   192.168.0.101  TCP        68 88 → 62507 [PSH, ACK] Seq=248795 Ack=551 Win=16616 L
```

Figure 3.3.7

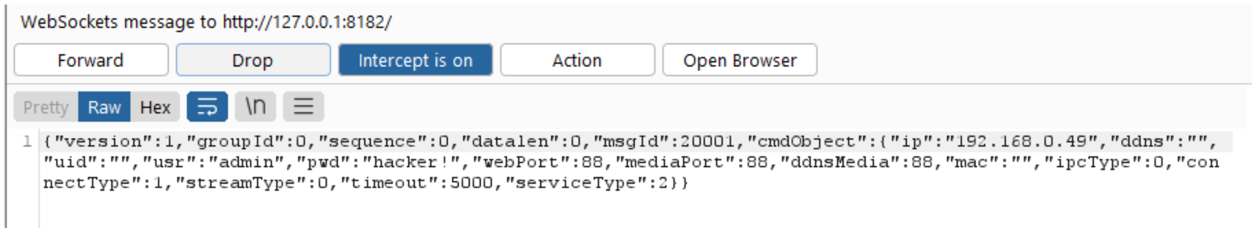
One of the last steps taken in my research was to use Burpsuite to capture requests regarding the functions of the web server application such as logins, logout, moving the camera around and other actions that are possible by the user from the application. As mentioned before, a plug-in must be installed for the web application to be functional. First, I captured the login requests and noticed the path for some of the camera code. At first the directories are forbidden as mentioned before, but using a whole path such as: "*http://<ip>:88/js/brand.js?ver=x* " will redirect me to the JS code. In addittion, in Figure 3.3.8 you can observe that most of the requests captured have a "MIME type" that are scripts.



Host	Meth...	URL	Params	Status	Length	MIME type
http://192.168.0.49:88	GET	/html/left.html?313395862	✓	200	36354	HTML
http://192.168.0.49:88	GET	/html/login.html?312259...	✓	200	5768	HTML
http://192.168.0.49:88	GET	/html/login.html?313251...	✓	200	5768	HTML
http://192.168.0.49:88	GET	/html/login.html?313395...	✓	200	5768	HTML
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/ansiUrlCodec.js?ver=...	✓	200	76710	script
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/brand.js?ver=164831...	✓	200	274	
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/browser.js?ver=1648...	✓	200	1607	script
http://192.168.0.49:88	GET	/js/cal.js		200	18549	script
http://192.168.0.49:88	GET	/js/classwy.js?ver=16483...	✓	200	29058	script
http://192.168.0.49:88	GET	/js/classwy.js?ver=16483...	✓	200	29058	script
http://192.168.0.49:88	GET	/js/classwy.js?ver=16483...	✓	200	29058	script

Figure 3.3.8

Next, with Burpsuite I managed to capture the WebSockets.js script. The script sends JSON data to the localhost proxy of the computer running the plug-in of the webserver. As you can see in Figure 3.3.9, the requests have the credentials as plain text in the second line of the output.

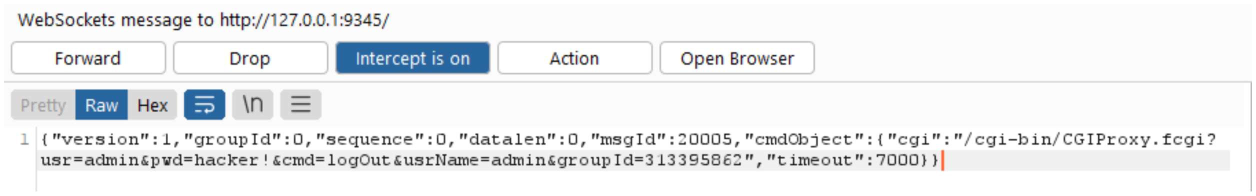


WebSockets message to http://127.0.0.1:8182/

```
{"version":1,"groupId":0,"sequence":0,"datalen":0,"msgId":20001,"cmdObject":{"ip":"192.168.0.49","ddns":"",
"uid":"","usr":"admin","pwd":"hacker!","webPort":88,"mediaPort":88,"ddnsMedia":88,"mac":"","ipcType":0,"con
nectType":1,"streamType":0,"timeout":5000,"serviceType":2}}
```

Figure 3.3.9

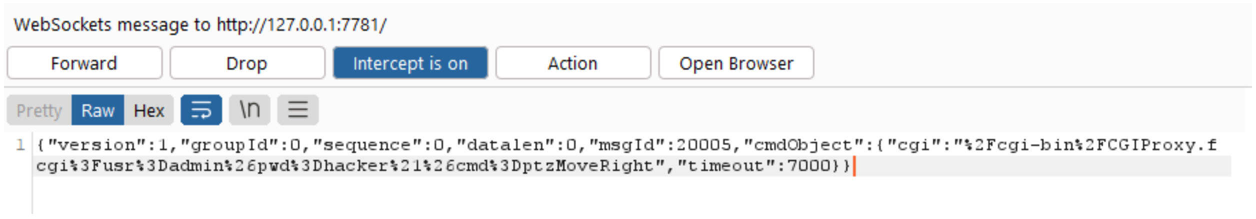
In addition, another request made from WebSockets.js script was captured using Burpsuite. I think this request is very interesting. In Figure 3.3.10 you can see the data being sent in this request; this data was URL encoded but I used the Burpsuite option to convert it into plain text. This request was made from a logout and communicates with the CGIProxy.fcgi with the credentials of the user and the command “logout”.



```
{"version":1,"groupId":0,"sequence":0,"datalen":0,"msgId":20005,"cmdObject":{"cgi":"/cgi-bin/CGIProxy.fcgi?usr=admin&pwd=hacker!&cmd=logOut&usrName=admin&groupId=313395862","timeout":7000}}
```

Figure 3.3.10

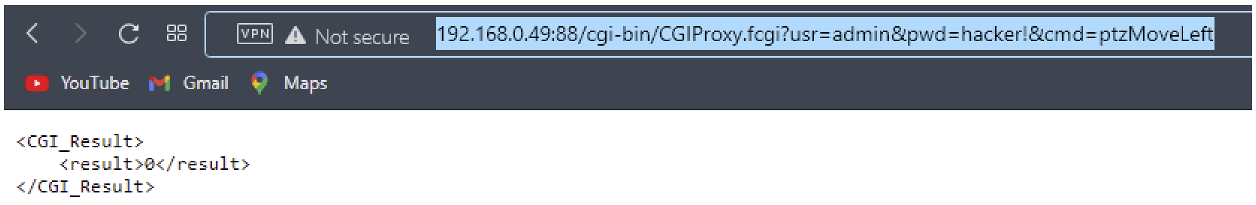
The above finding required me to check the source code of script sending this request. To visit the source code of this request I typed in the URL the following address: *http:<IP>/js/websocket_core.js.* In this code it was possible to see that different message IDs represented some sort of command in the system. In the other hand, "CMD" represents the command being done in the FOSCAM camera. For example, if we compare the "CMD" in Figure 3..3.10 and Figure 3.3.11 we can see that one logs the user out and the other moves the camera to a certain direction until another request is sent with the command "ptzStop".



```
{"version":1,"groupId":0,"sequence":0,"datalen":0,"msgId":20005,"cmdObject":{"cgi":"%2Fcgi-bin%2FCGIProxy.fcgi%3Fusr%3Dadmin%26pwd%3Dhacker%21%26cmd%3DptzMoveRight","timeout":7000}}
```

Figure 3.3.11

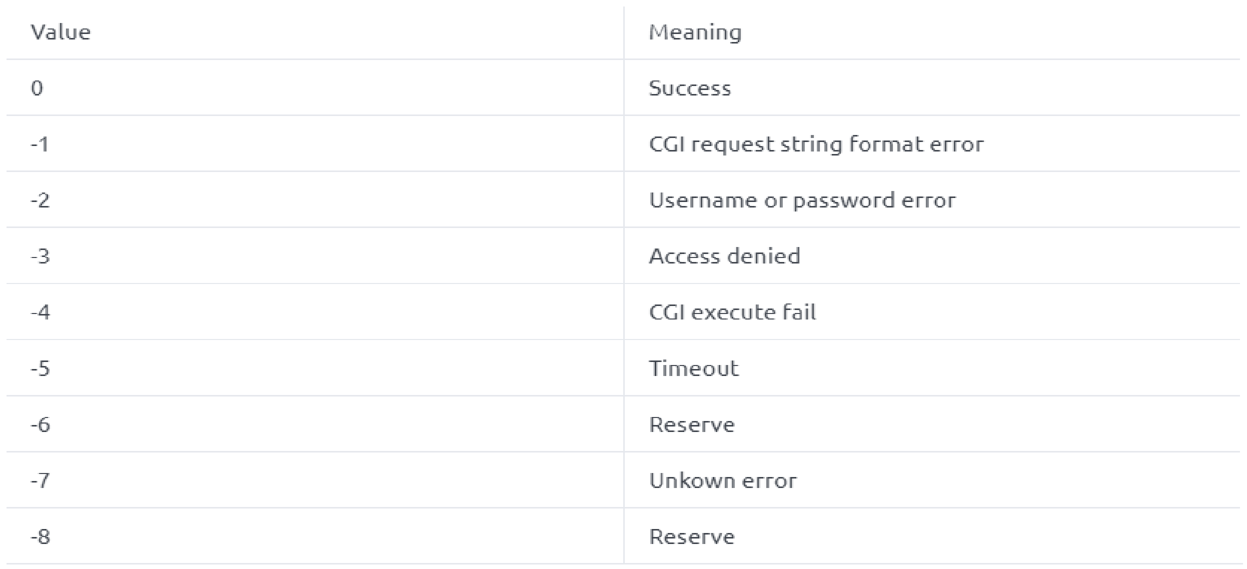
Next, I tried to use XSS techniques to try to run some application functions such as moving left or right and trying to run other Java functions. As seen in Figure 3.3.12, I tried to run application related commands successfully. All the attempts resulted in an integer result. In Figure 3.3.13 it is possible to see the different meaning from each of the integer results given by the application. I successfully managed to "move" the camera and run different application functions without opening the web application. To perform these actions, it is required to have authentication, this means we need a user and a password that match according to the Kerberos authentication from the FOSCAM camera.



```
192.168.0.49:88/cgi-bin/CGIProxy.fcgi?usr=admin&pwd=hacker!&cmd=ptzMoveLeft
```

```
<CGI_Result>
    <result>0</result>
</CGI_Result>
```

Figure 3.3.12



Value	Meaning
0	Success
-1	CGI request string format error
-2	Username or password error
-3	Access denied
-4	CGI execute fail
-5	Timeout
-6	Reserve
-7	Unkown error
-8	Reserve

Figure 3.3.13

4. Conclusions

In conclusion, the FOSCAM FI9816P security as it comes out of the box has different vulnerabilities and possible security flaws that can be of critical importance. In addition, there are some intriguing actions done by the web application, such as not supporting HTTPS for login. Another example isrunning application commands without being logged in but with authentication. If an attacker somehow gains access to some user in the camera, these commands can be running freely. In the other hand, FOSCAM is aware of the past known vulnerabilities and gave importance to the matter, as they are updating the firmware continuously. It is important to keep the firmware and the plug-in up to date to avoid already known attacks and reduce the chances of the product being exploited for further attacks or misuse the product.

References

FOSCAM. (n.d.). *FOSCAM FI9816P IP Camera.* Retrieved from FOSCAM: https://www.foscam.nl/pan-tilt-ip-camera/fi9816p.html

Howtouselinux. (n.d.). *Understanding PSH ACK TCP Flags.* Retrieved from Howtouselinux: https://www.howtouselinux.com/post/psh-ack-tcp-flags