## Methodology

As previously discussed the practical aspects of this investigation were carried out referencing the OWASP IoT Top ten testing project and methodology.

### Data Collection Methods

A total of six IoT devices were tested in this investigation, the devices were all tested in a lab and home environments using a PC computer and IPhone 6s for the mobile interface applications. All results have been kept strictly confidential with the investigator and project supervisor.

### Ethical Considerations

Only proof of concept data was using the testing of all applications therefore no personal user information was obtained or distributed. Any testing which could impact negatively on the devices services was not performed, therefore some areas of the OWASP IoT methodology testing was not performed such as web application testing due to ethical and legal reasons.

### IP Security Camera 1

The first device which was tested in this investigation was IP Security Camera 1, this generic IP camera operated over Wi-Fi and through Ethernet connection. It operates as a live CCTV network camera which has the capability of mobile, browser and desktop application viewing. Mobile viewing can also be done remotely via a mobile application. The camera allows for the user so store images and video at set intervals or constantly.

IP Security camera was set up in a lab testing environment connected via Ethernet to begin with for calibration until Wi-Fi capability was set up.

The first stage of testing this camera involved performing a Nmap scan against the ip address of the camera. Nmap is a free to use tool which is used for port scanning and network exploration including what services are running on the open ports of the device [28]. The IP address of the camera in this case was found to be 192.168.1.104.

The Nmap scan results for IP Camera 1 can be evidenced in figure 8 below;

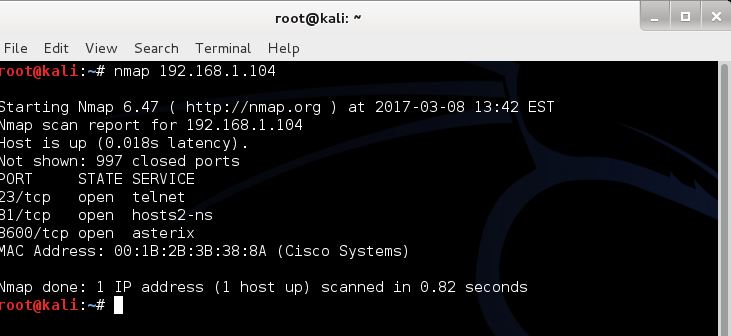


Figure 8 – Nmap Scan IP Camera 1

The Nmap scan clearly shows three open ports on the device;

* TCP Port 23- Telnet
* TCP Port 81 – hosts2-ns
* TCP Port 8600 – asterix

The presence of an open telnet service running on its default port was of much interest as telnet is a service which allows for the remote connection to network devices.

The next stage of the test was to attempt to remotely connect to the camera via its telnet service.



Figure 9 – Telnet IP Camera 1

As evidenced in figure 9 above access to the IP Camera was granted, this was achieved by simply guessing the root password of the device, the following credentials were used to connect via telnet to the camera;

* Login – **root**
* Password – **123456**

Once connected to the camera exploration of the files present on the IP Camera could be carried out. Files present on the device are presented in in figure 10 below;

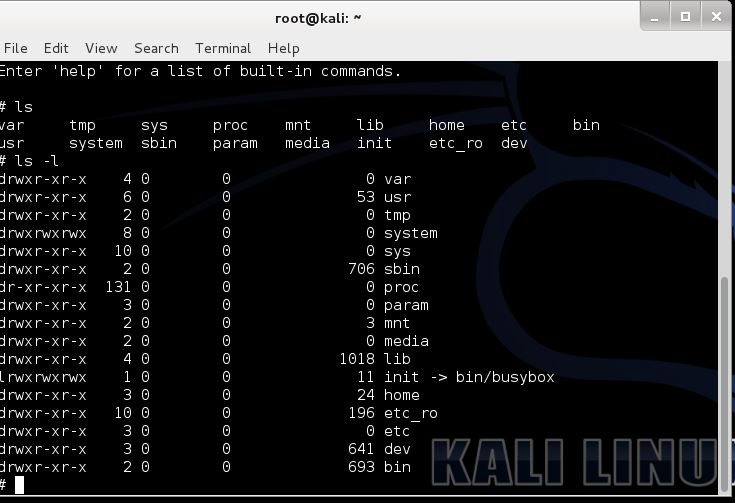


Figure 10 – IP Camera 1 file system

After further exploring the files within the IP Cameras system an interesting file called “ipcam.sh” was discovered, the location of this file within the system was “/system/init/ipcam.sh” which can be seen in figure 11 below;

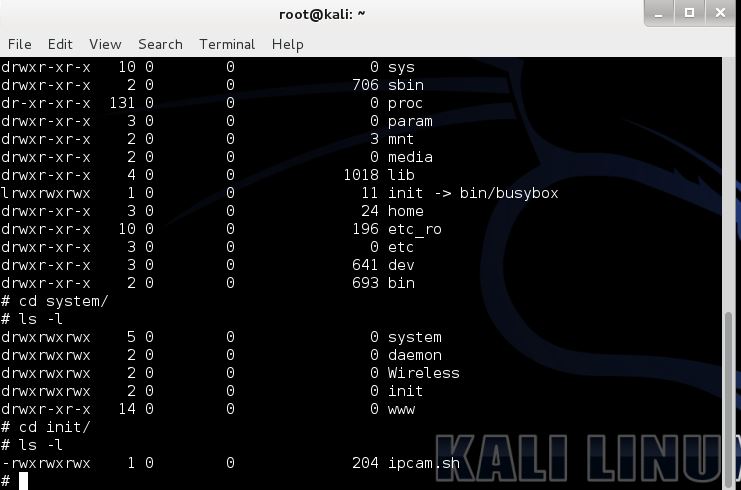


Figure 11- ipcam.sh file location

When the file “ipcam.sh” was run it returned interested results as the complete system setup and configuration of the IP Camera was dumped to the screen.

Upon close inspection of the system configuration dump it was discovered that all username and password information was displayed as shown in figure 12 below;

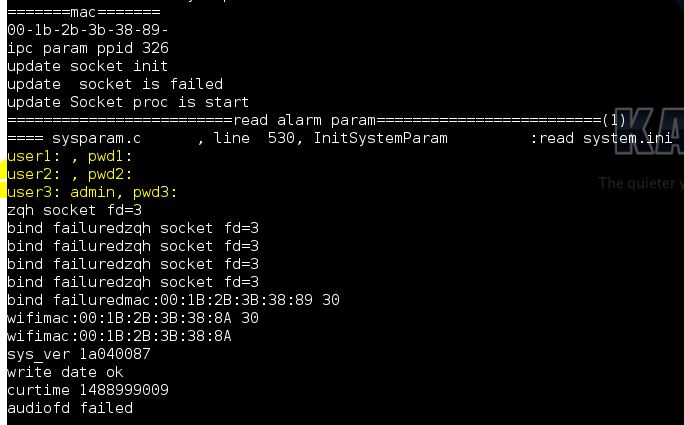


Figure 12 – Username and password dump

In order to confirm that this was in fact the usernames and passwords of the user accounts the password was changed using the mobile application evidenced in figure 13 below;

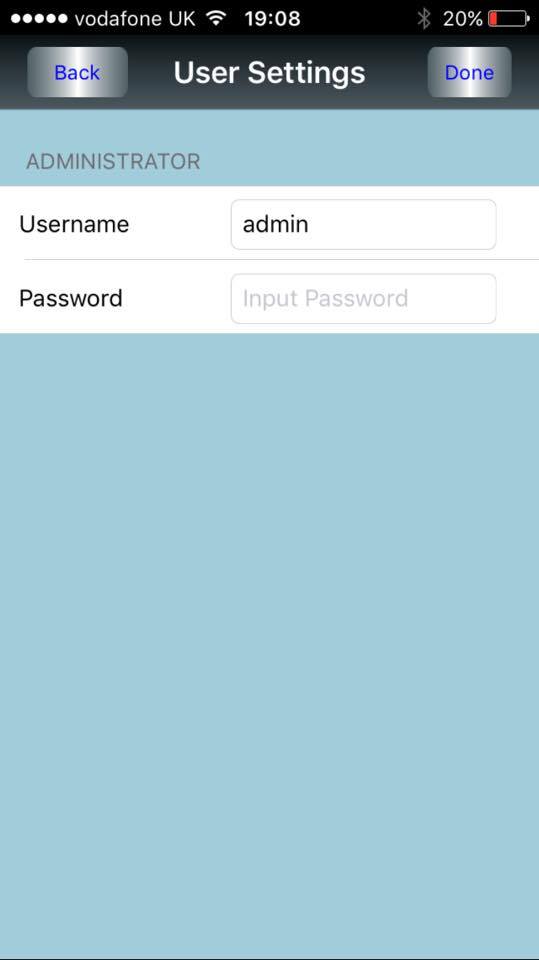
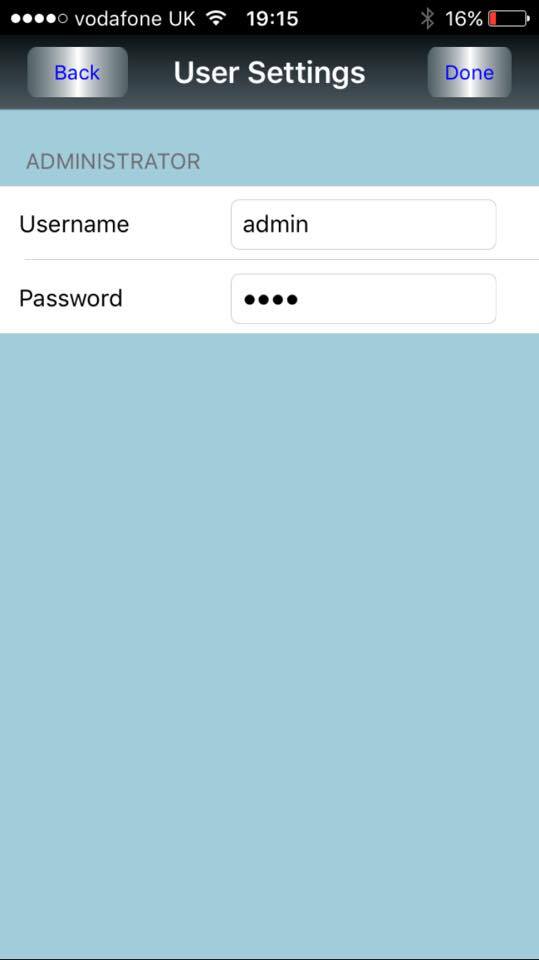


Figure 13 – Password change test

The password was changed to “test” using the mobile application.

Once the password was changed the “ipcam.sh” file was executed once again which confirmed suspicions that the dumped usernames and password in question were in fact accurate. Evidence of the dumped usernames and passwords after the changing of the user’s passwords is demonstrated in figure 14 below;

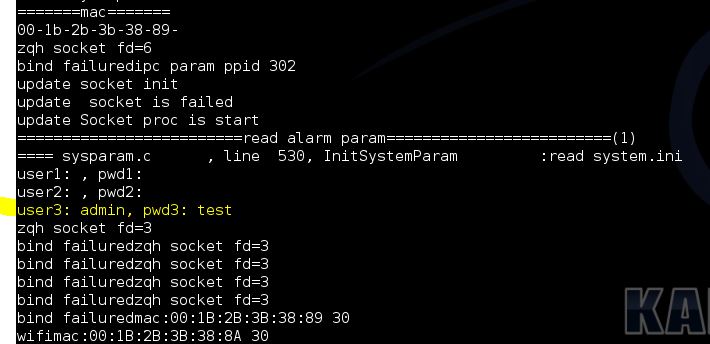


Figure 14 – User Information dump after password change

### IP Security Camera 2

The second device to be tested was a secondary IP Security Camera, it must be noted that this camera was of an entirely separate brand to IP Security camera 1. This IP Camera operated in much the same was as IP Security Camera 1 in that it produced a live video feed which can be sent to a mobile application to allow the user to view it. This camera also allowed for rotation on two axis as well as the ability to both listen to and send live audio to and from the camera. The camera was connected to the network via Ethernet connection until Wi-Fi settings could be configured. The mobile application was also install and set up allowing connection to the camera via Wi-Fi.

The first stage in testing IP Security Camera 2 was to perform a Nmap port and service scan against its IP Address. The IP address of IP Camera 2 was “192.168.1.216”.

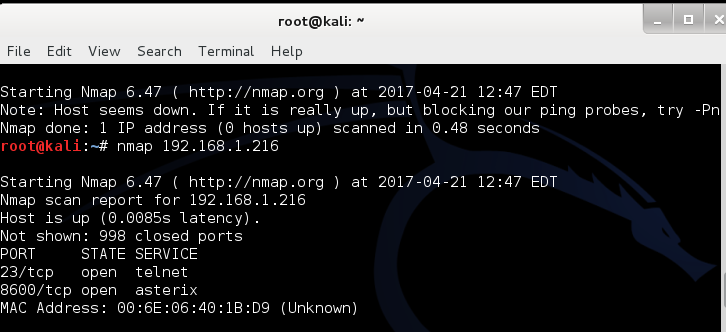


Figure 15 – Nmap Scan IP Camera 2

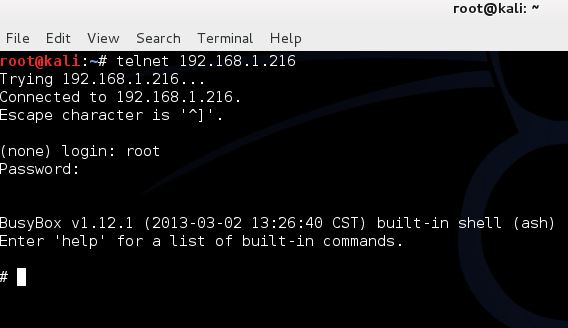
Figure 15 above displays the Nmap port and service scan results for IP Security Camera 2. The two ports open are;

* TCP Port 23 - telnet
* TCP Port 8600 – asterix

An attempt to connect to the IP camera through telnet was then carried out. With the use of password guessing, the telnet login and password were found to be;

* Login – **root**
* Password – **123456**

The process of the connection to IP Security Camera 2 is demonstrated in figure 15 below;

Figure 15 – Telnet login to IP Security Camera 2

Once connection to the IP Camera was established its file system was explored, the files present on IP Security Camera 2 are shown in figure 16 below;

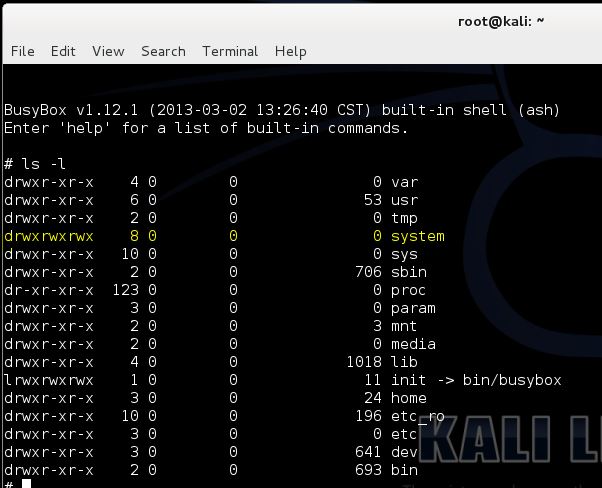


Figure 15 – IP Security Camera 2 file system

As the file system appeared to be very similar to that of IP Security Camera 1 the “/system/init” directory was explored for similar files as “ipcam.sh” which was discovered in IP Camera 1.



Figure 16 – ipcam.sh file position IP Camera 2

The Linux executable file “ipcam.sh” was located in the “system/init” file directory as evidenced by figure 16 above. When executed this file dumped all system configuration to the screen similar to IP Security Camera 1.

Within the output of “ipcam.sh” was a list of usernames and passwords as shown in figure 17 below;

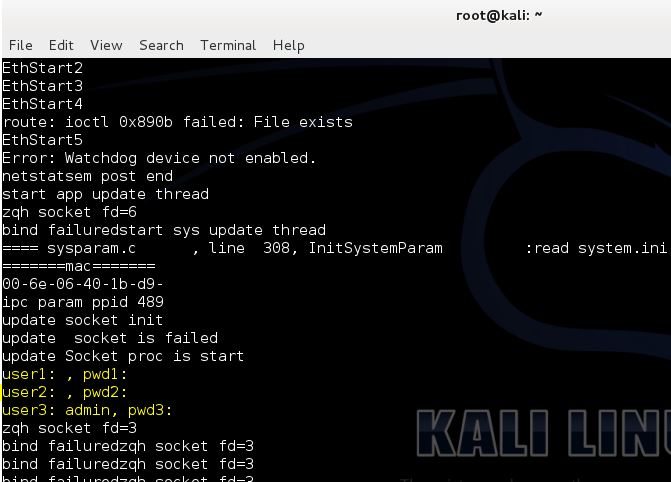


Figure 17 – Logins and password dump IP Camera 2

As with IP camera 1 the password of the user account was changed in order to confirm that the accounts displayed when executing the “ipcam.sh” file were in fact the correct user account details.

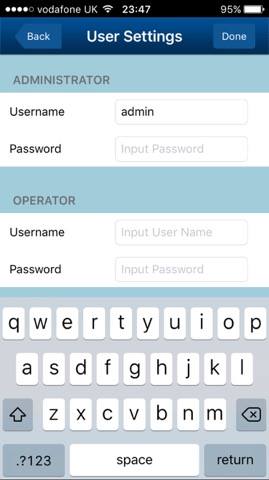
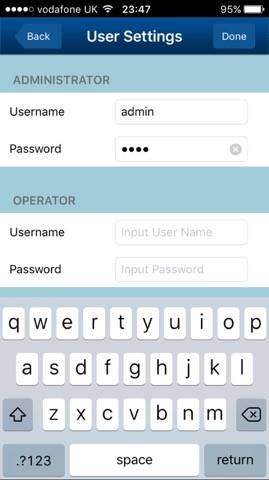


Figure 18 – Password change test IP Camera 2

Figure 18 above evidences the change in password to “test”. Following this change the “system/init/ipcam.sh” file was re executed and the user information had infact changed to “test” as evidenced in figure 19 below;

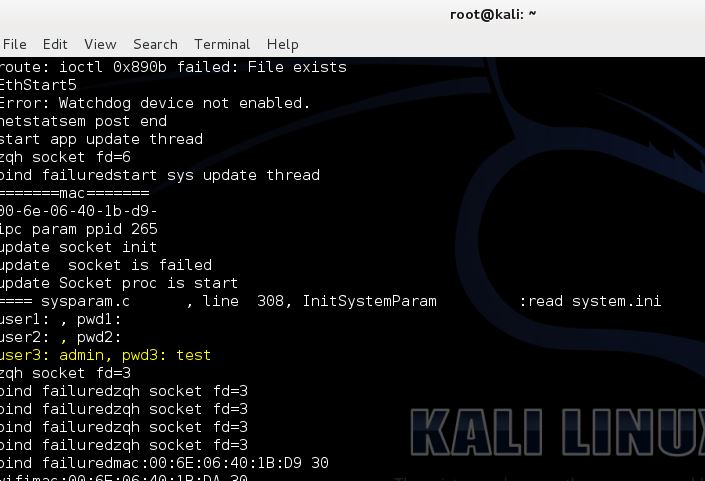


Figure 19 – information dump after password change IP Camera 2

### Baby Monitor

The third device tested was an IP security camera which was marketed as a home baby monitor which has been “secured” for the users protect. The device worked in much of the same way as the other IP Security Camera devices tested in this investigation. The camera allows for live streaming of images to a dedicated mobile and browser applications.

The first stage in testing the baby monitor was to perform a Nmap ports and services scan against its IP address, the IP address of the baby monitor in this case was 192.168.1.214. The baby monitor was found to have only one port open;

* TCP Port 14987 – Unknown Service

After further investigation port 14987 was found to be running a telnet service.

Figure 20 below demonstrates the output of the Nmap scan performed against the baby monitor and the attempt to connect to the telnet service;

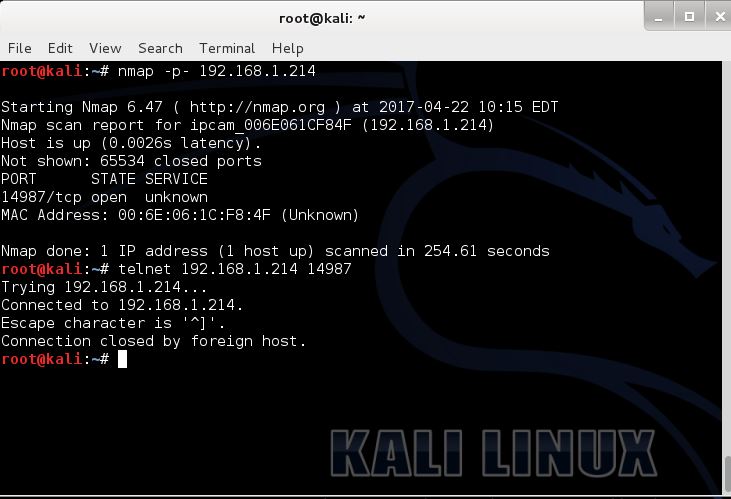


Figure 20 – Baby Monitor Nmap Scan Results and telnet connection

The baby monitor device has disabled remote connection to its systems over telnet however the next stage of the testing, packet sniffing, revealed that the telnet service running on port 14987 was unencrypted.

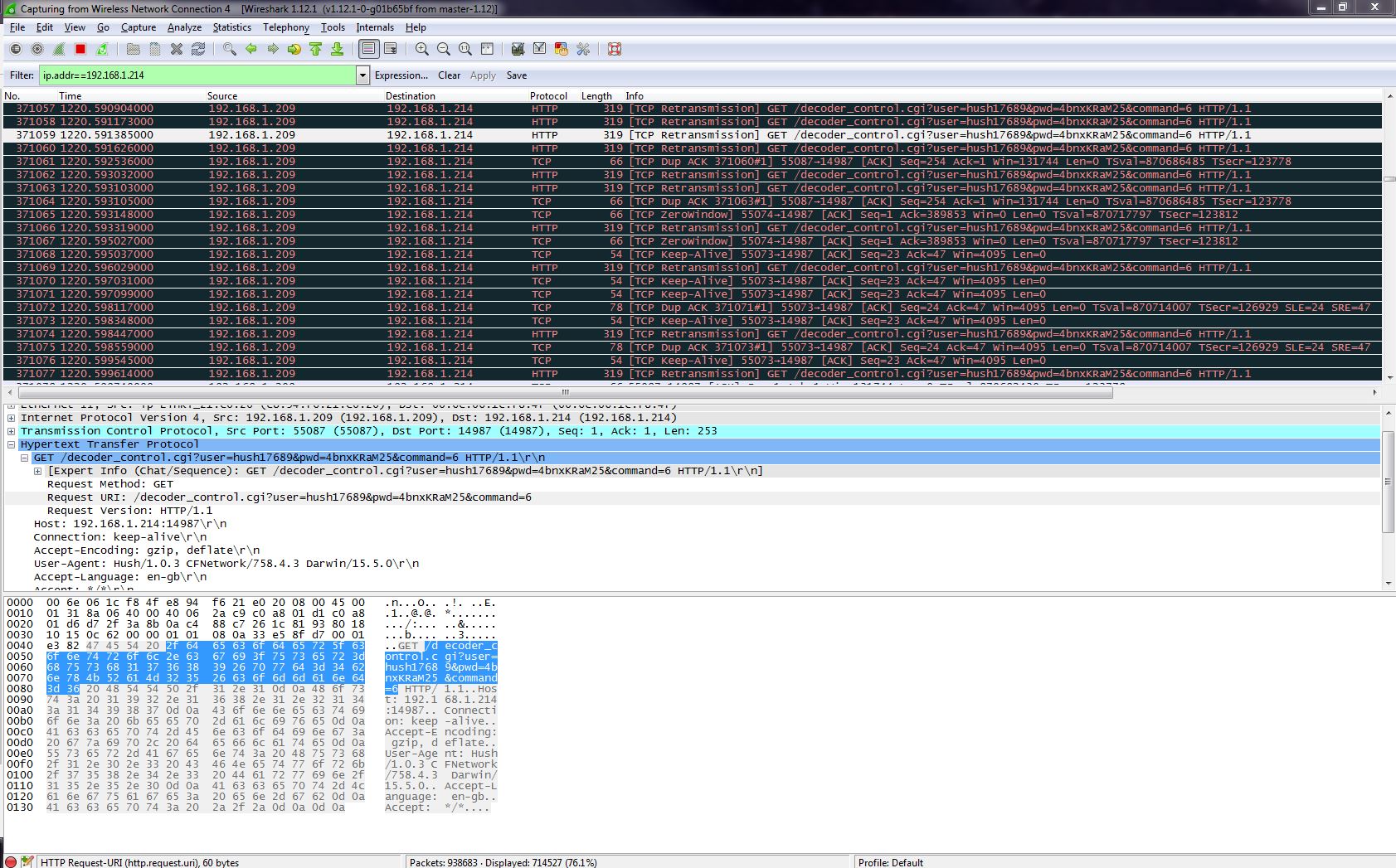


Figure 21 – Packet sniffing of Baby Monitor

Figure 21 above shows the Wireshark analysis of the data packets transferred to and from the baby monitor, within one of the data packets is the user credentials used to connect to “192.168.1.214” port “14987”. The credentials discovered are as follows;

* User – **hush17689**
* Password – **4bnxKRaM25**

### Bluetooth Heart Rate and Blood Pressure Monitor

The fourth device tested was a wireless blood pressure and heart rate monitor, this device used Bluetooth to communicate to and from a dedicated mobile application. The mobile application itself allowed the user to capture and store blood pressure and heart rate information within the mobile device itself. The user is provided with the option to export all data to a healthcare professional or friend/ family member, this functionality of the application makes use of email to export the data.

The first stage of testing the Bluetooth blood pressure monitor was to perform a man in the middle attack between the mobile application and cloud interface. The process was completed using the Cain and able penetration testing tool and Wireshark. Figure 22 below evidences the process of the man in the middle attack on the device;

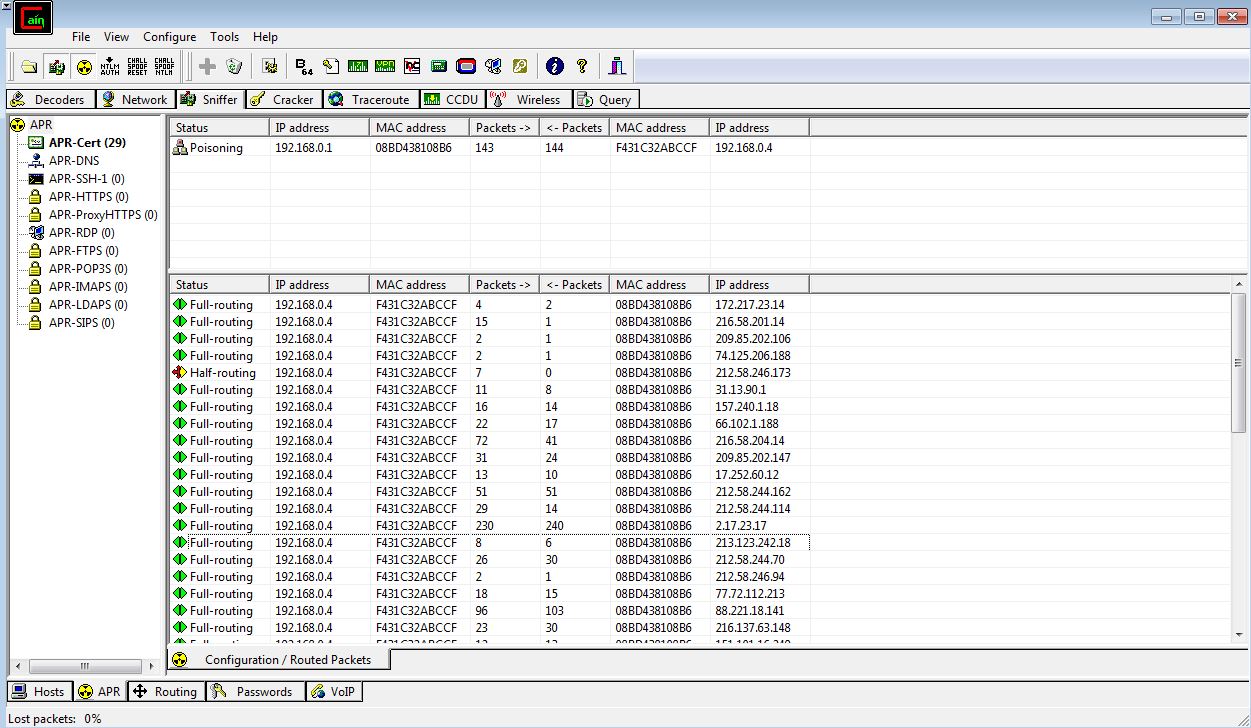


Figure 22 – Man in the middle Cain

Analysis of the certificates generated when performing the man in the middle attack showed a connection to the devices cloud interface as shown in figure 23 below;

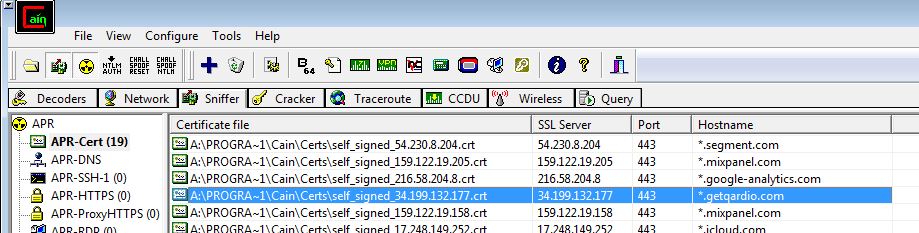


Figure 23 – Cloud interface connection

Analysis of the data packet transfer to this cloud interface via Wireshark showed that all traffic was in fact encrypted using SSL (TLS v1.2). In order to circumnavigate this encryption a too called man in the middle proxy (mitmproxy) was used in order to receive the data unencrypted. Figure 24 below evidences the unencrypted traffic using mitmproxy;

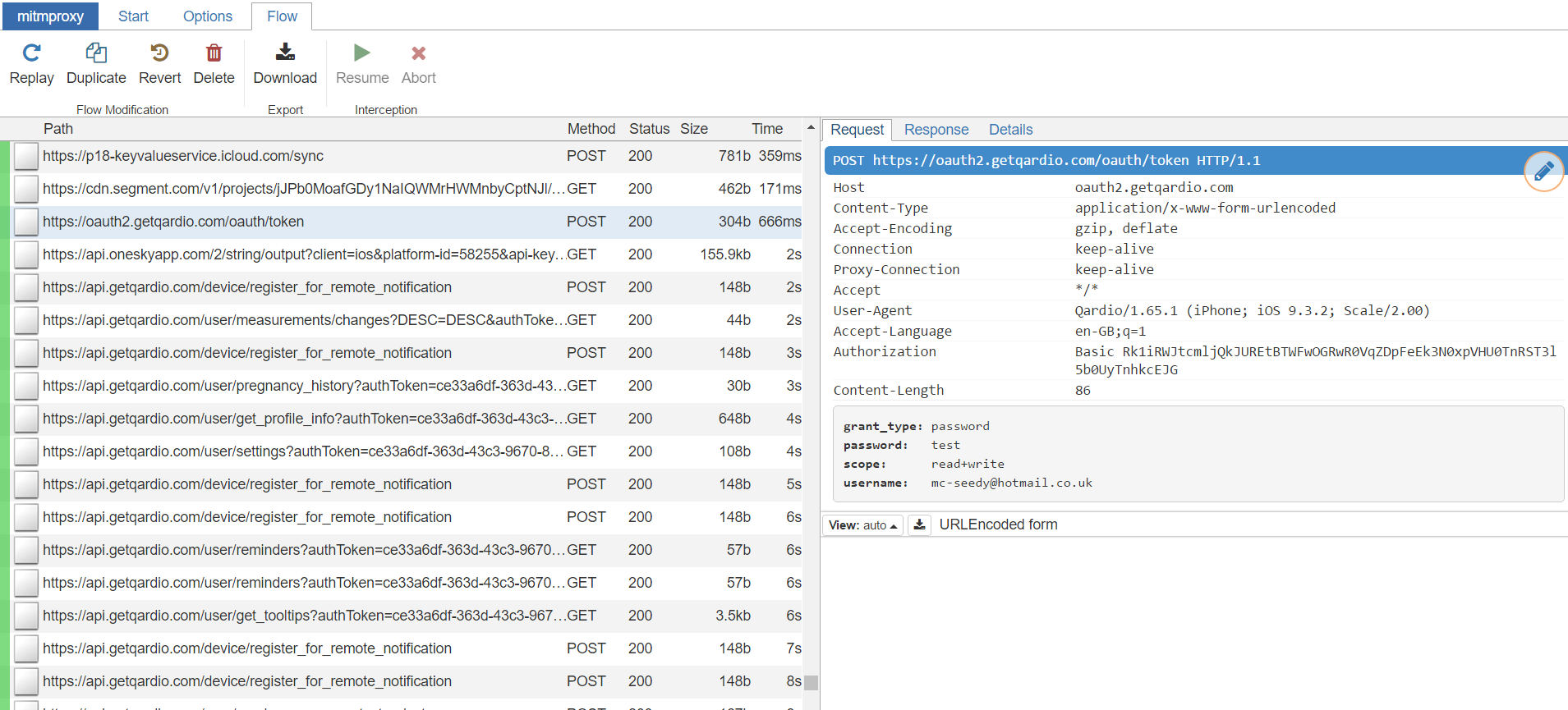


Figure 24 – mitmproxy blood pressure monitor

Within the viewable unencrypted connections and data was the username and password of the user as shown in figure 24 above. Upon further inspection of the data transfers was the complete user profile and information being transferred to the cloud interface of the device as detailed in figure 25 below;

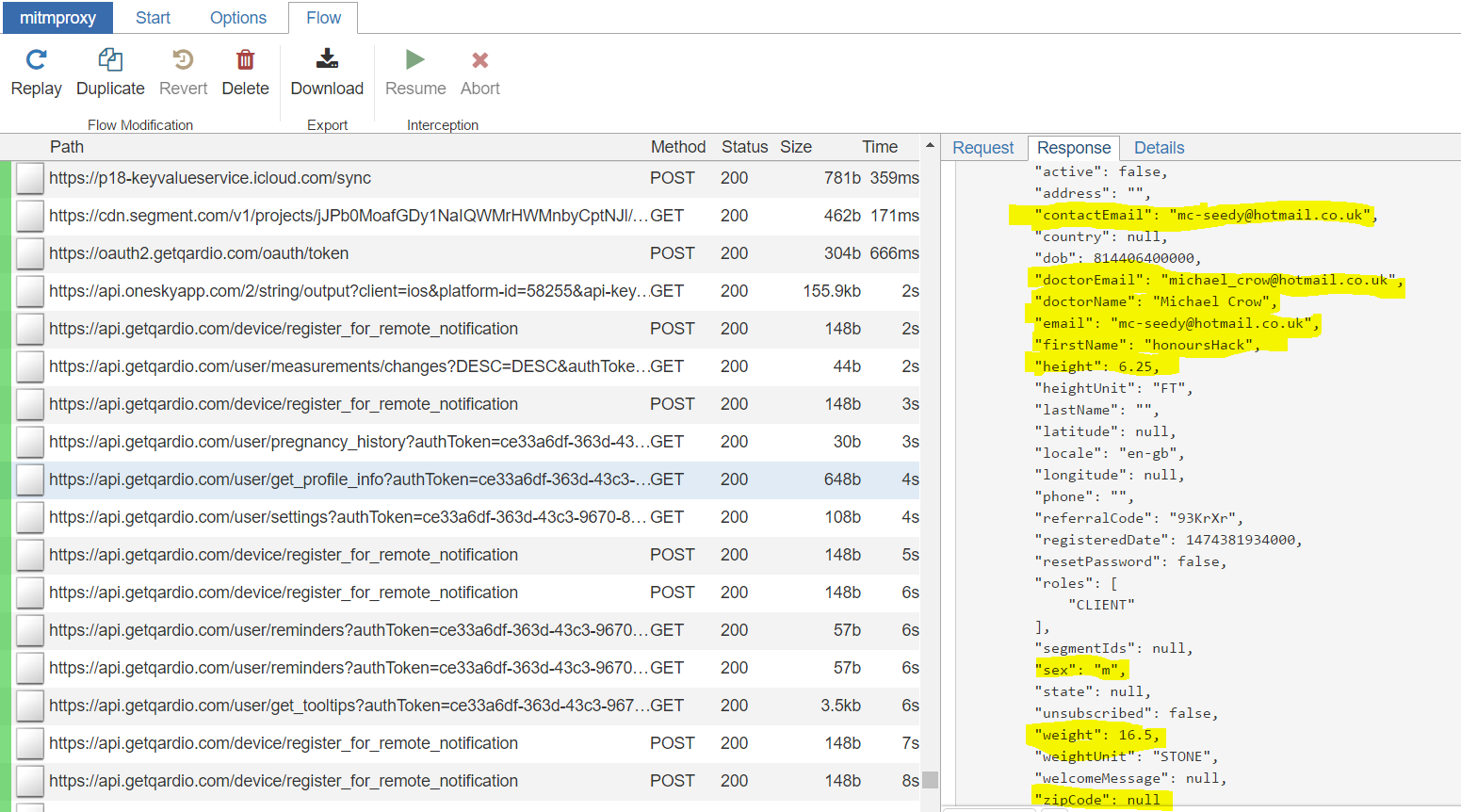


Figure 25 – Personal information Blood pressure monitor

This information included the doctor information and user information such as height, geographical location, phone number, sex, weight and postal code. When the user exported the data within the email functionality of the mobile app the complete contents of the email could be captured including any notes, it should also be noted that using mitmproxy allows for the editing of this data for example email address and then by replaying the connection the data could be sent to another email address chosen by the attacker.

### Smart Power Plug

The next device tested was a generic smart power socket device, this device allows a user to control the turning on and off any electrical device connected via mains plug socket, the device advertises that connection and control of the smart plug is only available through its dedicated mobile application. The device also provides the user with time setting functionality which allows the user to a specific time to allow or disallow power to the electrical item connected via the smart power plug.

The first stage in testing was to attempt to Nmap scan the device in search for open ports and services present on the device however, all communication ports were closed on the device. The next stage of testing was to assess the level of security present on the mobile application, upon inspection it was discovered that there was numerous applications present on the apple app store which would allow any mobile user in the vicinity of the smart plug to add it as a device without a specific password, the user could also access the plug outwith the network in which it is set up, in the case of this investigation the smart plug could be controlled using a second mobile device using 4G as shown in figure 26 below;

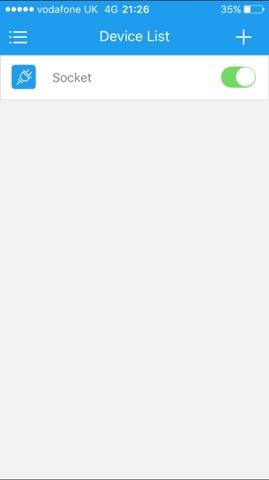


Figure 26 – smart plug remote network

After further research into the smart plug itself, it was discovered that a desktop application had been created by a previous user that when run would allow for the connection to the plug connected to the same network as the desktop itself. The application created by is freely available on GitHub. Figure 27 below is a combination of screenshots evidencing the applications functionality in relation to the turning off and on of the smart plug;

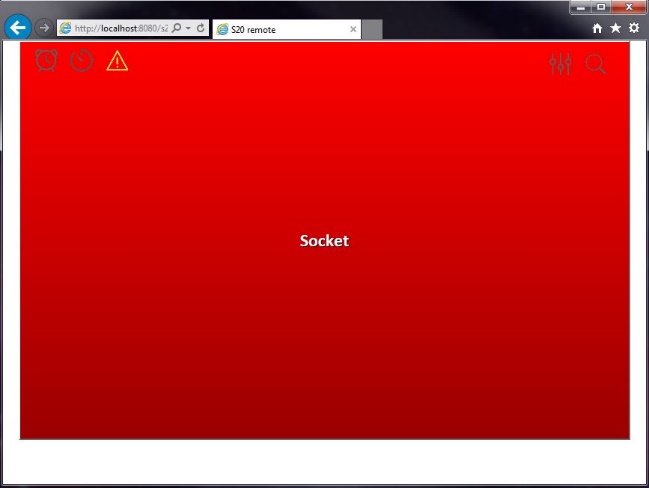
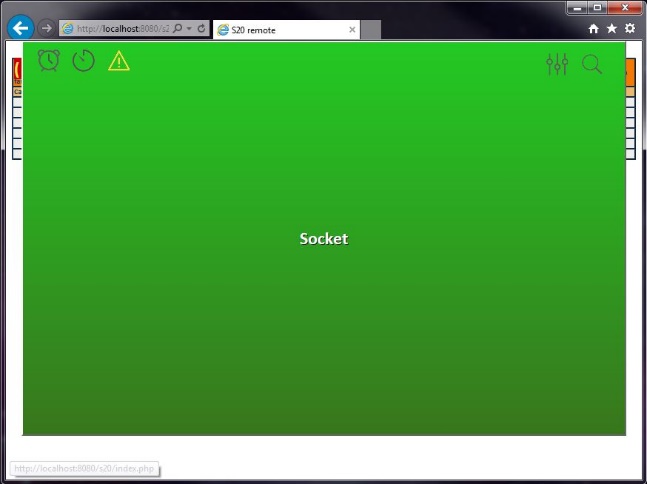


Figure 27 – Smart Plug Desktop Application

### Bluetooth Weight scales

The sixth device tested was a smart weight scale, the scale makes the use of Bluetooth technology to connect to a mobile device which has been pre-installed with a dedicated mobile application. The mobile application acts as a general health tool which contains information on the user’s weight history, height and other personal details. The application also provides an email functionality which can export all health data to a specified email address.

The testing of this device was conducted using mitmproxy in order to capture the communication of the smart scale mobile application. The first interesting communication captured was the immediate connection to the applications cloud interface, this connection displayed the username and password in the request packet, however the user’s password was in fact encrypted. Figure 28 below is details the capture of the cloud connection with username and encrypted password;

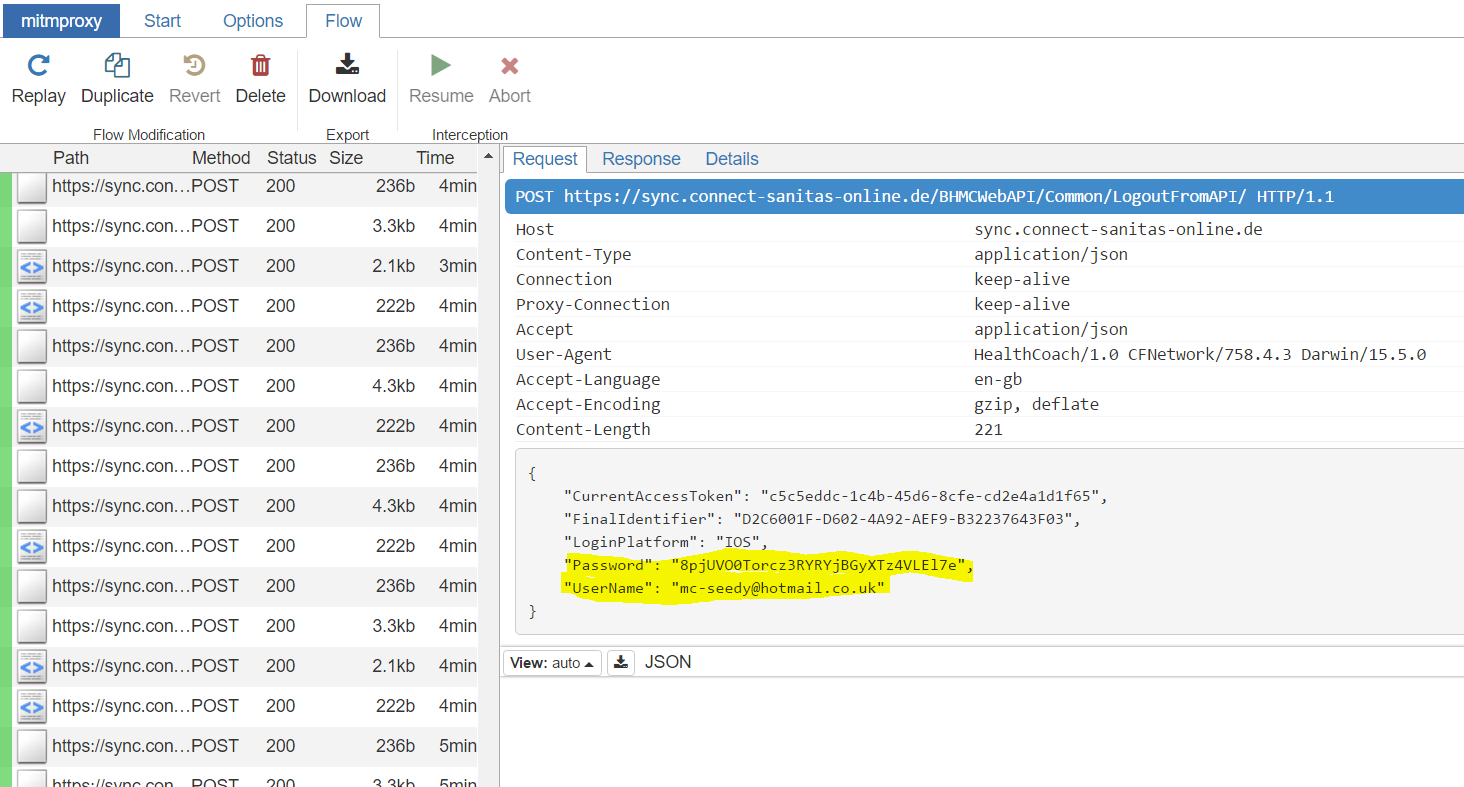


Figure 28 – Smart Scales Cloud Connection

The next stage of the testing involved the analysis of further connection to the cloud interface from the mobile interface, one connection to the cloud interface returned the entire personal profile of the user, information such as medication counters and sleep counters were displayed, furthermore a full database upload was captured which included multiple counts of private information. A summary of the private user information counts can be evidenced in figure 29 below;

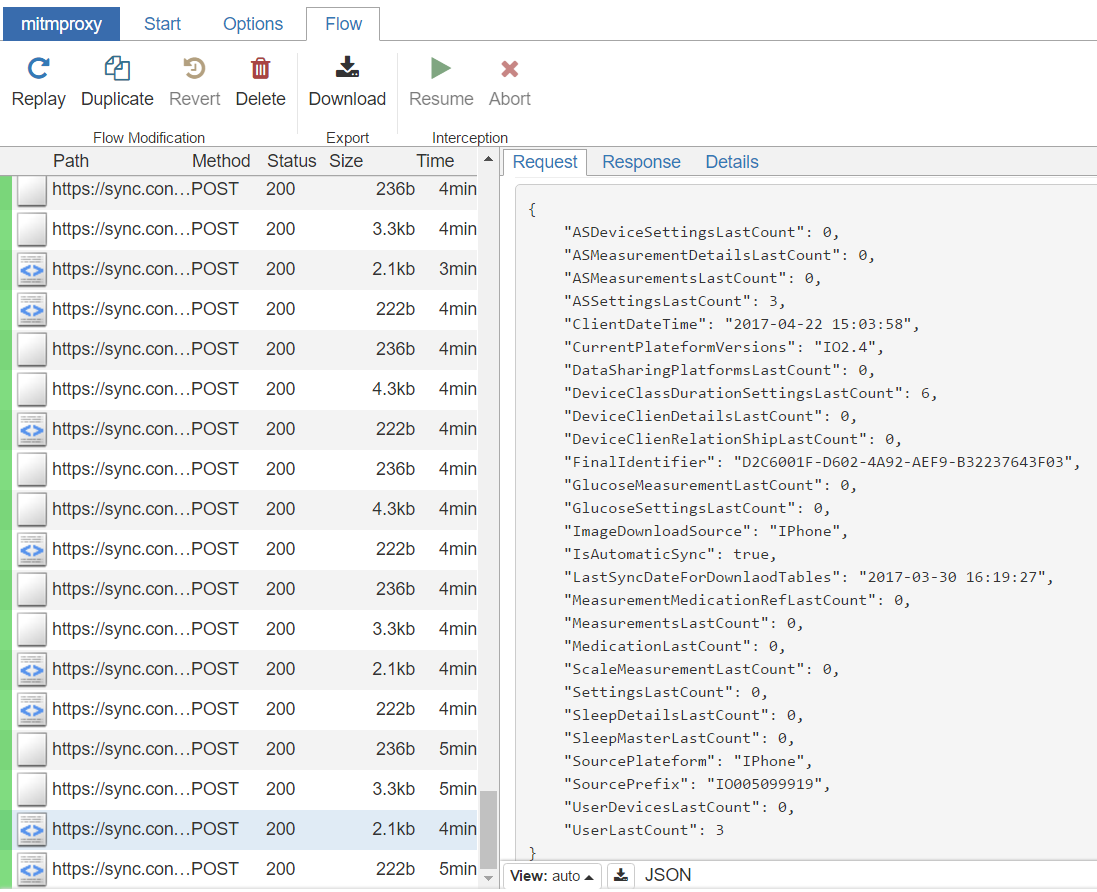


Figure 29 – Smart scales user information count



## Results

Table three bellows depicts the number of vulnerabilities found in each device which was tested in this investigation;

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IOT Device Vulnerability table | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | Vulnerabilities --> | Insufficient Authentication / Authorization | Insecure Network Services | Lack of Transport Encryption | Privacy Concerns | Insecure Cloud Interface | Insecure Mobile Interface | Insufficient security configurability | Insecure software/firmware | Poor Physical Security |
| ID No. | Device |  |  |  |  |  |  |  |  |  |
| 1 | IP Camera 1 |  |  |  |  |  |  |  |  |  |
| 2 | IP Camera 2 (With Sound) |  |  |  |  |  |  |  |  |  |
| 3 | Blood pressure Monitor |  |  |  |  |  |  |  |  |  |
| 4 | Smart Power Plug |  |  |  |  |  |  |  |  |  |
| 5 | IP Camera 3 (Baby Monitor) |  |  |  |  |  |  |  |  |  |
| 6 | Bluetooth Scales |  |  |  |  |  |  |  |  |  |

Table 3 – IoT Vulnerability Results Table

Figure 30 below details the number of vulnerabilities grouped by device;

Figure 30 – No. of Vulnerabilities by Device

### IP Security Camera 1

The first IP Security Camera that was tested in this investigation was found to have a total of six vulnerabilities. The vulnerabilities found were;

* Insufficient Authentication / Authorization
* Insecure Network Services
* Privacy Concerns
* Insufficient security configurability
* Insecure software/firmware
* Poor Physical Security

### IP Security Camera 2

The second IP Security Camera that was tested in this investigation was found to have a total of six vulnerabilities. The vulnerabilities found were;

* Insufficient Authentication / Authorization
* Insecure Network Services
* Privacy Concerns
* Insufficient security configurability
* Insecure software/firmware
* Poor Physical Security

### Baby Monitor

The third IP Security camera tested in this investigation which was marketed as a baby monitor contained a total of six vulnerabilities. The vulnerabilities found consisted of;

* Insufficient Authentication / Authorization
* Insecure Network Services
* Lack of Transport Encryption
* Privacy Concerns
* Insufficient security configurability
* Poor Physical Security

### Bluetooth Blood Pressure and Heart Rate Monitor

The Bluetooth Blood pressure and heart rate monitor tested in this investigation was found to have a total of five security vulnerabilities. The vulnerabilities found consisted of;

* Lack of Transport Encryption
* Privacy Concerns
* Insecure Cloud Interface
* Insecure Mobile Interface
* Insufficient security configurability

### Smart Power Plug

The smart power socket testing in this investigation was found to have a total of four existing vulnerabilities. The vulnerabilities found consisted of;

* Insufficient Authentication / Authorization
* Insecure Mobile Interface
* Insufficient security configurability
* Insecure software/firmware

### Bluetooth Weight Scales

The Bluetooth weight scales tested in this investigation had a total of four security related vulnerabilities. The vulnerabilities found consisted of;

* Lack of Transport Encryption
* Privacy Concerns
* Insecure Cloud Interface
* Insufficient security configurability

### Security Concerns Grouped by Vulnerability

Overall there was a total of thirty-one vulnerabilities found when testing all devices in this investigation. Figure 31 below details the number of vulnerabilities found during testing grouped by the type of vulnerability present on devices;

Figure 31 – No. of Security Concerns Grouped by vulnerability