Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

MONITORIZAÇÃO E ANÁLISE SEGURA DE CONSUMOS ENERGÉTICOS

GONÇALO GRAÇA RIBEIRO

Leiria, setembro de 2020

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

MONITORIZAÇÃO E ANÁLISE SEGURA DE CONSUMOS ENERGÉTICOS

GONÇALO GRAÇA RIBEIRO
Número: 2180076

Projeto realizado sob orientação do Professor Doutor Miguel Monteiro de Sousa Frade (miguel.frade@ipleiria.pt).

Leiria, setembro de 2020

AGRADECIMENTOS

Gostava de começar por agradecer à minha família, por me apoiar nos meus melhores momentos e por me «aturar» nos meus piores. É graças a eles que hoje estou onde estou, devo em grande parte à minha família os valores e educação que hoje tenho e que me acompanharão durante o resto da minha vida e por isso estar-lhes-ei para sempre grato e espero um dia poder retribuir todo o apoio que me foi dado.

Agradeço ao meu orientador, o professor Miguel Frade, por me guiar durante o decorrer deste trabalho e por toda a paciência e tempo que tal processo acarreta, sei que sem o professor este trabalho não seria possível. Estou grato pelo entusiasmo, incentivo e orientação prestados pelo professor sem nunca por em causa a minha autonomia, deixando-me aprender e crescer ao longo deste ano com os seus conselhos e com os meus erros.

Agradeço também, não só aos meus professores do Mestrado em Cibersegurança e Informática Forense pelo gosto e interesse incutidos em mim nas várias áreas da Cibersegurança ao longo do Mestrado mas também aos meus professores da Licenciatura em Engenharia Informática pelos conhecimentos que me foram transmitidos e que estão hoje na base deste trabalho.

Aproveito ainda para agradecer aos meus dois cães, responsáveis por tornar cada dia neste percurso de cinco anos um pouco mais colorido.

Por último dedico este trabalho ao meu pai. Espero tornar o meu pai orgulhoso não só do meu trabalho, como da pessoa que sou e que vou ser.

Obrigado.

RESUMO

Hoje em dia é possível utilizar dispositivos intitulados de *smart meters* para registar e monitorizar os consumos energéticos de uma rede elétrica. O uso deste tipo de dispositivo permite aos seus utilizadores comprovar a veracidade das suas faturas energéticas, compreender melhor a natureza dos seus consumos energéticos e desta forma proceder a alterações para reduzir o consumo eletrico.

No entanto, a utilidade proporcionada pelo uso de *smart meters* deve ser sempre acompanhada pelo cuidado no processamento e transmissão dos dados recolhidos por estes. Com efeito, as leituras energéticas de um utilizador podem, por exemplo, ser utilizadas para derivar destas os hábitos quotidianos deste, devendo tanto quanto possível zelar-se pela confidencialidade e integridade deste tipo de dados.

Neste trabalho foi feita uma análise de segurança ao *smart meter* Mirubee, bem como às duas plataforma web associadas a este, a Smilics e a Circutor. Foi feita também uma análise dos dados armazenados nestas plataformas para fundamentar o cuidado que se deve ter ao tratar este tipo de dados.

Com base nesta análise de segurança foi então projetada e implementada uma solução alternativa que não só colmatou os problemas de segurança detetados inicialmente mas que foi também desenvolvida com o objetivo de ser facilmente instanciada pelo utilizador. Para comprovar a eficácia dos mecanismos de segurança empregues nesta solução, foram ainda realizados testes de penetração à plataforma criada e foram tratadas as vulnerabilidades identificadas por estes testes.

ABSTRACT

Nowadays, it is possible to register and monitor energy consumptions of an electrical grid through devices known as smart meters. By using this type of device, its users are capable of verifying the accuracy of their energy bills, are able to better understand the nature of their energy consumption and thusly make the necessary adjustments in order to decrease their electrical consumption.

However, the utility provided by the use of smart meters should be accompanied by a careful processing and transmission of the data collected by these devices. For example, the energy readings of a user could be used to derive from these his daily habits and as such, the confidentiality of this type of data should be, as much as possible, emphasized.

In this work, a security analysis was made not only to the Mirubee smart meter but also to two web platforms associated with this device: Smilics and Circutor. An in depth analysis was also made to the data stored on these platforms to justify the care that should be employed when handling this kind of data.

Based on this security analysis, an alternative solution was then designed and implemented that not only addressed the security problems initially detected but was also developed with the aim of being easily deployed by the user. In order to prove the effectiveness of the security mechanisms employed in this solution, penetration tests were also carried out on this newly implemented platform and the vulnerabilities identified by these tests were then fixed.

ÍNDICE

Agradecimentos i
Resumo iii
Abstract v
Índice vii
Lista de Figuras xi
Lista de Tabelas xiii
Lista de Abreviaturas xv

1 INTRODUÇÃO 1
1.1 Enquadramento . . . 1
1.1.1 Internet of Things . . . 1
1.1.2 *Smart Meters* . . . 10
1.2 Objetivos . . . 18

2 TRABALHO RELACIONADO 21
2.1 Efergy . . . 21
2.2 OWL . . . 23
2.3 EDP Re:dy . . . 26
2.4 OpenEnergyMonitor . . . 28
2.5 Mirubee . . . 31
2.6 Tabela Comparativa . . . 32

3 ANÁLISE DO MIRUBEE 35
3.1 Engenharia Reversa da Aplicação Móvel Wibeee . . . 35
3.2 Análise de Segurança das Plataformas Associadas . . . 36
3.3 Métricas Calculadas pelo Mirubee . . . 38

4 ANÁLISE DE PADRÕES DE CONSUMO 41

5 SEGURANÇA NO DESENVOLVIMENTO DE APLICAÇÕES WEB 53
5.1 Top 10 Vulnerabilidades Segundo a OWASP . . . 53
5.1.1 Injeção . . . 55

5.1.2 Quebra de Autenticação . . . 56
5.1.3 Exposição de Dados Sensíveis . . . 57
5.1.4 Entidades Externas de XML . . . 57
5.1.5 Quebra de Controlo de Acessos . . . 58
5.1.6 Configurações de Segurança Incorretas . . . 59
5.1.7 Cross-Site Scripting . . . 59
5.1.8 Desserialização Insegura . . . 62
5.1.9 Utilização de Componentes Vulneráveis . . . 62
5.1.10 Registo e Monitorização Insuficiente . . . 63
5.2 Soluções . . . 63
5.2.1 Injeção . . . 63
5.2.2 Quebra de Autenticação . . . 64
5.2.3 Exposição de Dados Sensíveis . . . 65
5.2.4 Entidades Externas de XML . . . 65
5.2.5 Quebra de Controlo de Acessos . . . 65
5.2.6 Configurações de Segurança Incorrectas . . . 66
5.2.7 Cross-Site Scripting . . . 66
5.2.8 Desserialização Insegura . . . 67
5.2.9 Utilização de Componentes Vulneráveis . . . 67
5.2.10 Registo e Monitorização Insuficiente . . . 67

6 SOLUÇÃO PROPOSTA 69
6.1 Enquadramento . . . 69
6.2 Arquitetura da Solução Proposta . . . 71
6.2.1 Arquitetura do Servidor . . . 72
6.2.2 Arquitetura dos *Sites* do Utilizador . . . 74
6.3 Funcionalidades da Plataforma Web Desenvolvida . . . 75
6.3.1 Funcionalidades de Segurança . . . 76
6.3.2 Funcionalidades de Monitorização de Consumos Energéticos . 80
6.4 Síntese . . . 86

7 DESENVOLVIMENTO DA PLATAFORMA WEB 87
7.1 Implementação do *Proxy* Mirubee . . . 87
7.1.1 Instanciação do *Endpoint* e Envio de Leituras . . . 87
7.2 Implementação do Servidor (*Back-end*) da Plataforma Web . . . 89
7.2.1 Rotas do Servidor . . . 93
7.2.2 Alertas Definidos pelo Utilizador . . . 94
7.2.3 Armazenamento de Leituras . . . 95

7.3 Implementação do Cliente (*Front-end*) da Plataforma Web 97
7.3.1 Rotas do Cliente 98
7.3.2 Desenho de Gráficos 99
7.4 Síntese 101

8 TESTES DE SEGURANÇA 103
8.1 *Test Bed* 103
8.2 Injeção SQL 104
8.2.1 *Endpoint* «https://mirubee.test/api/login» 104
8.2.2 *Endpoint* «https://mirubee.test/api/users/email/{userEmail}» 107
8.2.3 *Endpoint* «https://mirubee.test/api/sites/{siteId}/devices/» 108
8.3 Quebra de Autenticação 109
8.4 Ataques de Man-In-The-Middle 112
8.5 Cross-Site Scripting 116

9 CONCLUSÕES 123
9.1 Trabalho Futuro 125
9.2 Notificação Responsável 125

BIBLIOGRAFIA 127

Apêndices

A OUTPUT DOS TESTES DE SEGURANÇA REALIZADOS 133
A.1 Testes de injeção 133
A.2 Quebra de Autenticação 145
A.3 Ataques de Man-In-The-Middle 146

B NOTIFICAÇÃO RESPONSÁVEL À SMILICS 149

DECLARAÇÃO 151

LISTA DE FIGURAS

Figura 1 Arquiteturas IoT . . . 3
Figura 2 Hierarquia dos problemas de segurança em WSN . . . 4
Figura 3 Taxonomia dos riscos de segurança IoT . . . 8
Figura 4 Gráfico da corrente elétrica aquando de uma avaria . . . 11
Figura 5 Horário simples praticado pela EDP . . . 12
Figura 6 Horários bi-horários praticados pela EDP . . . 13
Figura 7 Horários tri-horários praticados pela EDP . . . 14
Figura 8 Exemplo de um *smart meter* distribuído pela EDP . . . 16
Figura 9 Exemplos de *smart meters* sem conectividade à Internet . . . 16
Figura 10 Exemplos de *smart meters* com conectividade à Internet . . . 16
Figura 11 *Smart meter* Mirubee utilizado neste projeto . . . 19
Figura 12 Gama de equipamentos Efergy . . . 22
Figura 13 Aplicação móvel Efergy Pro . . . 23
Figura 14 Plataforma web Engage . . . 24
Figura 15 Gama «Eletricity Monitors» da OWL . . . 24
Figura 16 Plataforma web Intuition . . . 25
Figura 17 Intuituion-lc (versão trifásica) . . . 25
Figura 18 EDP Re:dy Box . . . 26
Figura 19 Periféricos Re:dy . . . 27
Figura 20 Aplicação móvel EDP Re:dy . . . 28
Figura 21 EmonPi . . . 28
Figura 22 Sensor CT . . . 29
Figura 23 Dashboard emonCMS . . . 30
Figura 24 Mirubee . . . 31
Figura 25 Dashboard Smilics . . . 32
Figura 26 Gráfico da média de consumo energético ao longo do dia . . . 44
Figura 27 Evolução semanal dos consumos médios . . . 45
Figura 28 Consumo diário durante a semana e o fim de semana . . . 47
Figura 29 Consumo energético ao longo do dia entre estações do ano . . . 49
Figura 30 Evolução média dos consumos diários durante o confinamento 50
Figura 31 Diagrama de uma ataque de XSS refletido . . . 61
Figura 32 Processo de configuração do Mirubee . . . 70

Figura 33 Arquitetura geral da solução proposta 71
Figura 34 Arquitetura do servidor da aplicação 73
Figura 35 Arquitetura de um *site* 74
Figura 36 Configuração do endereço e porto do servidor do Mirubee . 75
Figura 37 Certificado SSL gerado 76
Figura 38 Processo de *handshake* e troca de chaves 77
Figura 39 Transmissão de dados cifrados através do protocolo TLS v1.2 77
Figura 40 Autenticação do *forwarder* do Raspberry Pi 79
Figura 41 Consulta da voltagem da rede elétrica 81
Figura 42 Janela de gestão da tarifa energética 82
Figura 43 Gráfico do custo energético mensal de uma tarifa tri-horária 83
Figura 44 Simulação dos custos mensais de uma tarifa tri-horária . . . 84
Figura 45 Janela de gestão de alertas 84
Figura 46 Janela de gestão de dispositivos 85
Figura 47 Janela de detalhes de dispositivos 86
Figura 48 Diagrama de sequência do ciclo de vida do *proxy* do Mirubee 91
Figura 49 Exemplo dos tipos de gráficos utilizados neste projeto 100
Figura 50 Intercepção do pedido HTTPS de *login* 106
Figura 51 Ataque de dicionário efetuado com a ferramenta Patator . . 111
Figura 52 Pedido efetuado através da ferramenta Postman 111
Figura 53 Esquema de rede do cenário de teste MITM 112
Figura 54 *Output* da *framework* Bettercap 113
Figura 55 Pacotes transmitidos entre o cliente e o servidor 114
Figura 56 Pacotes capturados pelo Burp Suite 115
Figura 57 Erro de autenticação da assinatura reenviada 116
Figura 58 Teste manual de Cross-Site Scripting 117
Figura 59 Teste de Cross-Site Scripting no *endpoint* «api/sites» 117
Figura 60 Pedidos gerados pelo XSStrike capturados pelo Burp Suite . 118
Figura 61 Teste de Cross-Site Scripting no *endpoint* «api/alerts» . . . 119
Figura 62 Página de alertas na aplicação web 120
Figura 63 Segundo teste de Cross-Site Scripting no *endpoint* «api/alerts» 121
Figura 64 Certificado empregue pelo Burp Suite 146

LISTA DE TABELAS

Tabela 1 Preços das tarifas energéticas praticadas pela EDP 12
Tabela 2 Tabela comparativa entre *smart meters* 33
Tabela 3 Especificações da habitação monitorizada 42
Tabela 4 Consumo energético do utilizador consoante a estação do ano 46
Tabela 5 Modelo de classificação de risco da OWASP 54
Tabela 6 Classificação de vulnerabilidades web 54

LISTA DE ABREVIATURAS

AES	Advanced Encryption Standard.
API	Application Programming Interface.
ARP	Address Resolution Protocol.
AWS	Amazon Web Services.
BLE	Bluetooth Low Energy.
CA	Certificate Authority.
CBC	Cypher Block Chaining.
CoAP	Constrained Application Protocol.
CSV	Comma-separated values.
CT	Current Transformer.
CVE	Common Vulnerabilities and Exposures.
DAC	Discretionary Access Control.
DMBS	Database Management System.
DOM	Document Object Model.
DoS	Denial of Service.
DTLS	Datagram Transport Layer Security.
ECDSA	Elliptic Curve Digital Signature Algorithm.
EDP	Energias de Portugal.
FP7	Seventh Framework Programme.
FTPS	File Transfer Protocol Secure.
GUID	Global Unique Identifier.
HOTP	HMAC-based One-time Password algorithm.

HTML	HyperText Markup Language.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hyper Text Transfer Protocol Secure.
IANA	Internet Assigned Numbers Authority.
IoT	Internet of Things.
IP	Internet Protocol.
IPS	Intrusion Prevention System.
IPv4	Internet Protocol version 4.
IPv6	Internet Protocol version 6.
ITU	International Telecommunication Union.
JSON	JavaScript Object Notation.
kVA	Quilovoltampere.
kWh	Quilowatt-hora.
LAN	Local Area Network.
LCD	Liquid Crystal Display.
LDAP	Lightweight Directory Access Protocol.
MAC	Mandatory Access Control. Media Access Control.
MD5	Message Digest Algorithm 5.
MITM	Man-In-The-Middle.
MPA	Multi Page Application.
NAT	Network Address Translation.
NFC	Near Field Communication.
NILM	Nonintrusive Load Monitoring.
NPM	Node Package Manager.
NVD	National Vulnerability Database.
ORAM	Oblivious Random Access Memory.

ORM	Object-Relational Mapping.
OS	Operating System.
OTP	One-time password.
OWASP	Open Web Application Security Project.
PAN	Personal Area Network.
PHP	PHP: Hypertext Preprocessor.
PIR	Private Information Retrieval.
PKI	Public Key Infrastructure.
QR	Quick Response.
RAM	Random Access Memory.
RBAC	Role-Based AccessControl.
REST	Representational State Transfer.
RFID	Radio-Frequency Identification.
RGPD	Regulamento Geral de Proteção de Dados.
RPL	Routing Protocol for Low-Power and Lossy Networks.
SHA1	Secure Hash Algorithm 1.
SIEM	Security Information Event Management.
SMTP	Simple Mail Transfer Protocol.
SO	Sistema Operativo.
SOAP	Simple Object Access Protocol.
SPA	Single Page Application.
SPOF	Single Point of Failure.
SQL	Structured Query Language.
SQRL	Secure, Quick, Reliable Login.
SSH	Secure Shell.
SSL	Secure Sockets Layer.
TLS	Transport Layer Security.
TPM	Trusted Platform Module.

U2F	Universal 2nd Factor.
UE	União Europeia.
UI	User Interface.
URI	Uniform Resource Identifier.
URL	Uniform Resource Locator.
USB	Universal Serial Bus.
VLAN	Virtual Local Area Network.
WAF	Web Application Firewall.
WSN	Wireless Sensor Networks.
XACML	eXtensible Access Control Markup Language.
XML	Extensible Markup Language.
XSRF	Cross-Site Request Forgery.
XSS	Cross-Site Scripting.

1 INTRODUÇÃO

Com o advento e proliferação da IoT, em Inglês Internet of Things (IoT), na última década, tornou-se acessível ao público geral desenvolver projetos informáticos na web acessíveis em qualquer altura e em qualquer lugar.

Este ramo da Informática permite criar soluções simples de forma útil e cómoda, utilizando para tal dispositivos, denominados neste contexto por *things*, capazes de comunicar entre si mas normalmente limitados em termos de recursos (processamento, memória, armazenamento, *etc*). No entanto, a comodidade proporcionada por estes dispositivos e soluções «prontos a usar» muitas vezes contrasta com a segurança (ou falta dela) dos dados transmitidos em cenários IoT. Quer seja por falta de recursos, ou mesmo desleixo por parte dos *developers* destes projetos, muitas vezes a segurança dos dados nestes projetos é posta em segundo plano.

1.1 ENQUADRAMENTO

Nesta secção será feito um enquadramento aos conceitos fundamentais que serão abordados ao longo deste documento. Será inicialmente feito um enquadramento geral à IoT, bem como a conceitos relacionados com esta, seguido de um enquadramento aos *smart meters*, um tipo específico de *thing* que permite monitorizar consumos energéticos.

1.1.1 *Internet of Things*

Apesar do termo ter sido utilizado pela primeira vez por Kevin Ashton em 1999 (Ashton, 2009), foram precisos cerca de dez anos para se assistir ao desenvolvimento e expansão da IoT. Com efeito, na última década têm-se observado os mais variados desenvolvimentos neste ramo da Informática bem como o progressivo aumento, tanto do número de *things* ligadas à Internet, como do investimento realizado nesta área, estimando-se que anualmente se invista mais de um bilião de dólares em projetos IoT (IoTAnalytics, 2018; Jain e Gupta, 2018).

O trabaho de Madakam e Siddharth Tripathi (2015), realizado com o objetivo de criar uma visão geral da IoT, faz uma análise das principais arquiteturas e tecnologias empregues nesta área da Informática. Este estudo começa por estabelecer que não existe uma única definição de IoT visto que, diferentes entidades ao longo do tempo moldaram o significado deste termo e que hoje este significa algo diferente do que quando inicialmente foi cunhado por Ashton (2009).

De seguida o artigo realça que, uma vez que a IoT se trata de conceito tão lato, não existe uma única arquitetura globalmente aceite de soluções IoT. Assim sendo é feito um levantamento das principais arquiteturas deste ramo da Informática tais como: a arquitetura adotada pelo projeto de pesquisa europeu Seventh Framework Programme (FP7), a arquitetura estabelecida pela International Telecommunication Union (ITU) e a arquitetura de três camadas (Aplicação, Transporte e Percepção) definidas por Xiaocong e Jidong (2010) (ver Figura 1) entre outras. Por último são definidas as tecnologias mais relevantes em contextos IoT tais como:

- **Radio-Frequency Identification (RFID)**: através de ondas rádio, o RFID permite identificar pessoas ou objetos através de *tags* e leitores de *tags* ou antenas. Em cenários IoT esta tecnologia revela-se bastante útil para identificar objetos e dispositivos de forma automatizada;
- **Wi-Fi**: este protocolo permite aceder a uma Local Area Network (LAN) e à Internet sem utilizar fios;
- **Near Field Communication (NFC)**: apesar de ser uma tecnologia de curto alcance (permitindo um alcance máximo prático de cerca de 4 centímetros), o NFC permite a comunicação entre dispositivos e tornou-se numa tecnologia indispensável hoje em dia nas mais variadas soluções IoT. Através do NFC é possível estabelecer uma ligação e comunicação entre *things* de forma simples e eficiente;
- **Bluetooth**: permite criar uma Personal Area Network (PAN) entre dispositivos, geralmente com um alcance entre 10 a 100 metros, através de ondas rádio;
- **ZigBee**: esta tecnologia permite criar soluções escaláveis de baixo débito e consumo com um alcance semelhante ao Bluetooth (10 a 100 metros). Este protocolo permite ainda criar redes com topologias distintas (*e. g.* em estrela, malha ou árvore).

Também com o objetivo de caraterizar a IoT, Muntjir et al. (2017), realizou uma análise a novas arquiteturas IoT, bem como aos aspetos de segurança a ter em

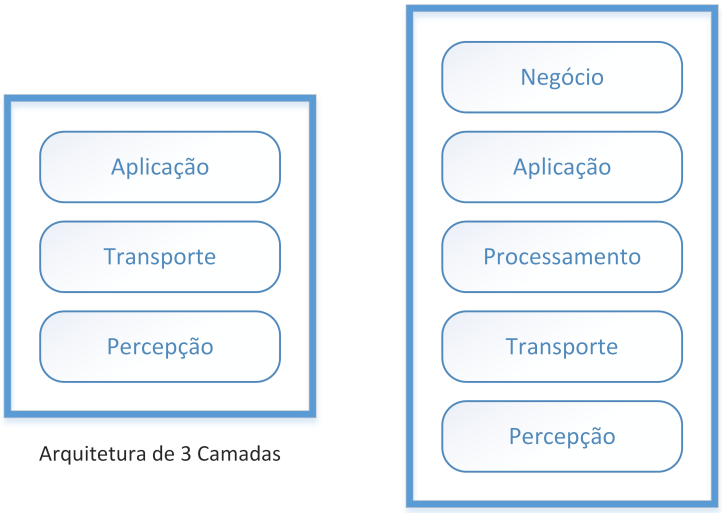


Figura 1: Arquiteturas IoT (adaptado do trabalho realizado por Muntjir et al. (2017))

conta em aplicações deste ramo. Segundo os autores, a arquitetura de três camadas definida por Xiaocong e Jidong (2010) evoluiu gradualmente para passar a conter cinco camadas Negócio, Aplicação, Processamento, Transporte e Percepção (ver Figura 1). Para além disto, este artigo introduz ainda os conceitos de arquiteturas *Cloud* e *Fog Based*[1] em soluções IoT.

No que toca às tecnologias utilizadas em cenários IoT, para além das enumeradas por Madakam e Siddharth Tripathi (2015), o artigo menciona ainda os seguintes protocolos:

- **Bluetooth Low Energy (BLE)**: este protocolo foi desenvolvido com o objetivo de permitir criar PANs com um alcance semelhante ao Bluetooth original com um consumo reduzido de energia. Em soluções IoT o BLE pode tornar-se bastante útil devido ao contrangimento de recursos (neste caso energia) das *things* normalmente empregues;
- **Quick Response (QR)**: um código QR é um tipo específico de código de barras que ao ser lido por um dispositivo é convertido para texto (tipicamente um Uniform Resource Identifier (URI)).

São ainda enumeradas as principais aplicações da IoT, tais como:

- **No setor da saúde**: onde é proposto o uso de sensores para monitorizar remotamente os pacientes de modo a facilitar a compreensão do seu estado de saúde por parte dos médicos;

1 *Fog Computing*: arquitetura semelhante ao *Cloud Computing* mas que atribui mais recursos (*e. g.* armazenamento ou processamento) a dispositivos no limite da rede (*Edge devices*)

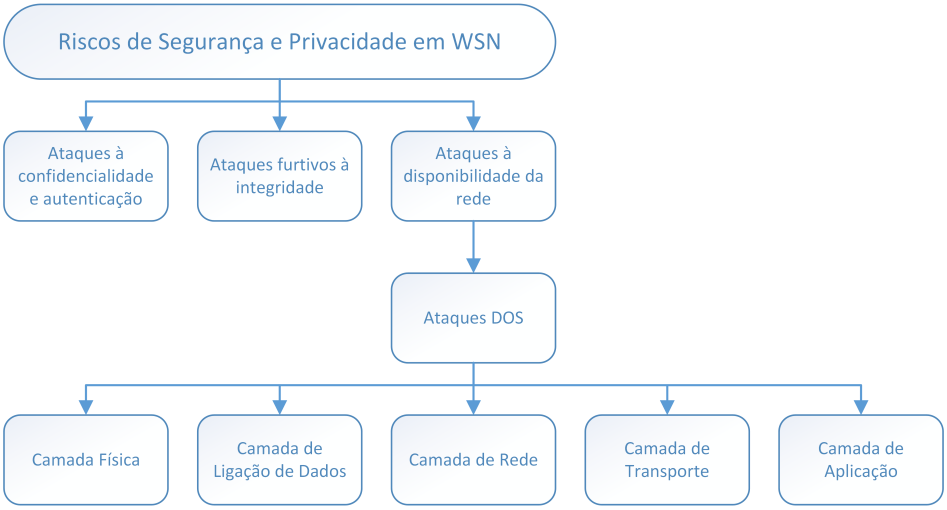


Figura 2: Hierarquia dos problemas de segurança em WSN (adaptado do trabalho realizado por Borgohain et al. (2015))

- **Em *smart homes* e *offices***: em que o utilizador pode controlar remotamente os vários dispositivos instalados nestes, tais como a iluminação, sistema de aquecimento, ar condicionado, *etc* com base em sensores que recolhem e transmitem dados em tempo real;
- **Em cenários de *smart parking***: sensores instalados nos locais de estacionamento de um parque podem monitorizar quais os lugares disponíveis e conduzir os utilizadores para estes;
- **Na indústria do retalho**: a IoT pode ser utilizada para melhorar a logística da cadeia de abastecimento empregue por estes negócios através de sensores RFID e NFC que rastreiam os produtos vendidos, desde a sua fonte até à sua eventual venda.

Finalmente, Muntjir et al. (2017), baseados no trabalho realizado por Borgohain et al. (2015), destacam os principais riscos de segurança observados em cenários IoT. Para tal, começam por estabelecer a hierarquia dos riscos de segurança em Wireless Sensor Networks (WSN). Tal como ilustrado na Figura 2 os riscos de seguranças em WSN podem ser divididos em três grupos distintos:

- Ataques à confidencialidade e autenticação dos dados transmitidos;
- Ataques furtivos à integridade do serviço prestado;
- Ataques à disponibilidade da rede do serviço, também conhecidos por Denial of Service (DoS). Estes por sua vez podem ser subdivididos em cincos grupos distintos dependendo da camada da arquitetura protocolar onde ocorre o ataque (Física, Ligação de Dados, Rede, Transporte e Aplicação).

Khan e Salah (2017) fizeram uma revisão literária aos principais riscos de segurança em projetos IoT. Para tal, categorizaram cada um dos dezanove riscos enumerados numa de três classes consoante a camada arquitetural onde estes ocorrem (ver Figura 3):

- Riscos de segurança de baixo nível;
- Riscos de segurança de médio nível;
- Riscos de segurança de alto nível.

A fim de facilitar a compreensão da Figura 3, serão de seguida explicados os ataques, vulnerabilidades e conceitos presentes nesta:

- **Interferência**: este tipo de ataque ocorre em redes *wireless* quando são emitidos sinais de frequência rádio sem protocolo associado a estes;
- **Inicialização não segura**: uma inicialização não segura de um dispositivo pode comprometer a segurança deste e consequentemente da rede em que este se encontra;
- **Ataques de *spoofing* e de *sybil* de baixo nível**: um ataque de *spoofing* ocorre quando um atacante faz-se passar por outro utilizador. Ao assumir a identidade de um utilizador legítimo o atacante passa a poder aceder aos dados e recursos privilegiados do utilizador cuja identidade foi «roubada».

  Por outro lado, os ataques de *sybil* ocorrem em redes *peer-to-peer* quando um utilizador se faz passar por vários utilizadores em simultâneo, podendo levar à disseminação de dados maliciosos ou à negação de serviços prestados pela rede;
- **Interface física não segura**: uma interface física não segura pode ser explorada por um atacante para comprometer a integridade de um dispositivo;
- **Ataques de *sleep deprivation***: os ataques de *sleep deprivation* (ou de privação de sono) ocorrem principalmente em dispositivos que se encontram constrangidos por energia. Este tipo de ataque força os sensores destes dispositivos a estarem constantemente «acordados» o que leva ao rápido descarregamento da bateria ou fonte de energia do dispositivo alvo;
- **Ataques de *replay* ou duplicação devido a fragmentação**: este tipo de ataque consiste na interceção e retransmissão de dados entre entidades (*e. g.* na interseção das credencias de autenticação de um utilizador e posterior envio destas para o servidor). Ao usar o protocolo IEEE 802.15.4 (que especifica o funcionamento de PANs de baixo consumo, utilizado como base pelo protocolo

ZigBee) os dispositivos são forçados a realizar a fragmentação dos pacotes Internet Protocol version 6 (IPv6) enviados e respetiva reconstrução quando recebidos. A repetição ou duplicação do envio dos fragmentos destes pacotes pode levar ao esgotamento de recursos dos dispositivos alvo, afetando o processamento de pacotes legítimos;

- **Descoberta de vizinhos não segura**: numa rede IoT torna-se necessário que os vários dispositivos nesta consigam identificar-se e autenticar-se entre si. Este processo de descoberta implica o processamento de pacotes e resolução de endereços. Um atacante pode enviar pacotes de descoberta de vizinhos para saturar a rede em questão resultando na negação de serviços desta;
- **Ataques de reserva de *buffer***: um dispositivo IoT ao receber pacotes de rede necessita de reservar um *buffer* para realizar a montagem dos pacotes recebidos e fazer o seu devido processamento. Ao enviar pacotes incompletos um atacante pode causar a reserva indefinida deste *buffer* levando ao esgotamento de recurso do dispositivo e eventual descartar de pacotes legítimos;
- **Ataques de *routing* Routing Protocol for Low-Power and Lossy Networks (RPL)**: o protocolo de *routing* IPv6 RPL é particularmente vulnerável quando um dos nós de uma rede é comprometido, podendo levar ao esgotamento de recursos e à escuta indevida dos dispositivos dessa rede;
- **Ataques *sinkhole* e *wormhole***: um ataque de *sinkhole* ocorre quando um nó de uma rede é maliciosamente configurado para se promover aos outros nós como o nó mais próximo da *base station* (estação base da WSN). Desta forma, este nó passará a receber indevidamente todos os pacotes direcionados para a *base station*. Por outro lado, os ataques de *wormhole* ocorrem quando dois nós maliciosos criam um túnel entre si a partir dos quais reencaminham os pacotes recebidos. O encaminhamento indevido de pacotes, tanto nos ataques de *sinkhole* como de *wormhole*, pode levar à saturação da rede (e consequente negação do serviço prestado) e à alteração de pacotes através dos nós maliciosos;
- **Ataques de *sybil* em camadas intermédias**: semelhante aos ataques de sybil em camadas de baixo nível;
- **Autenticação e comunicação segura**: Para garantir a autenticação dos dispositivos numa rede é normalmente empregue um sistema de gestão de chaves. Tipicamente, em cenários IoT, devido ao constragimento de recursos dos dispositivos, é empregue o protocolo Datagram Transport Layer Security (DTLS);

- **Segurança ponto a ponto na camada de transporte**: para garantir a segurança ponto a ponto nesta camada devem ser utilizados protocolos e mecanismos que garantam a confidencialidade e autenticidade das mensagens enviadas que tenham em conta as limitações de processamentos dos dispositivos IoT;
- **Estabelecimento e retomada de sessão**: ao «roubar» a sessão de um dispositivo, um atacante passa a poder fazer passar-se por este. Assim, o atacante passará a receber os pacotes que seriam originalmente destinados ao dispositivo cuja sessão foi apropriada, o que pode levar ao comprometimento da confidencialidade dos dados transmitidos e à negação dos serviços prestados pela rede;
- **Violação de privacidade em sistema IoT na *cloud***: o *deployment* de soluções IoT na *cloud* implica a exposição destas soluções a uma série de desafios de segurança associados a esta arquitetura que podem comprometer a privacidade dos dados processados por estas;
- **Segurança do Constrained Application Protocol (CoAP)**: o CoAP é um protocolo aplicacional desenvolvido especificamente para o uso em dispositivos constragidos. Por omissão este protocolo não oferece «primitivas protocolares para autenticação ou autorização» (Shelby et al., 2014). Para garantir a confidencialidade e a autenticidade dos dados transmitidos por este protocolo, os programadores deverão utilizar o protocolo DTLS;
- **Interfaces não seguras**: a interfaces utilizadas para aceder a serviços IoT devem ser devidamente protegidos de forma a combater eventuais falhas de privacidade;
- ***Firmware*/*Software* não seguro**: o *firmware* e *software* utilizado deve ser sempre que possível devidamente testado e autenticado para garantir a segurança deste. Mais, tanto o *firmware* como *software* deve ser atualizado de maneira segura o mais atempadamente possível;
- **Segurança do *middleware***: deve-se zelar pela segurança do *middleware* empregue para garantir a correta comunicação entre os dispositivos da rede, independentemente da sua heterogeneidade.

Para além de fazer a enumeração dos principais riscos de segurança em cenários IoT, Minhaj Khan, com base nas referências bibliográficas utilizadas propõe uma solução para mitigar ou mesmo prevenir completamente cada um dos riscos mencionados. Este artigo científico sugere ainda casos de uso em que a implementação da

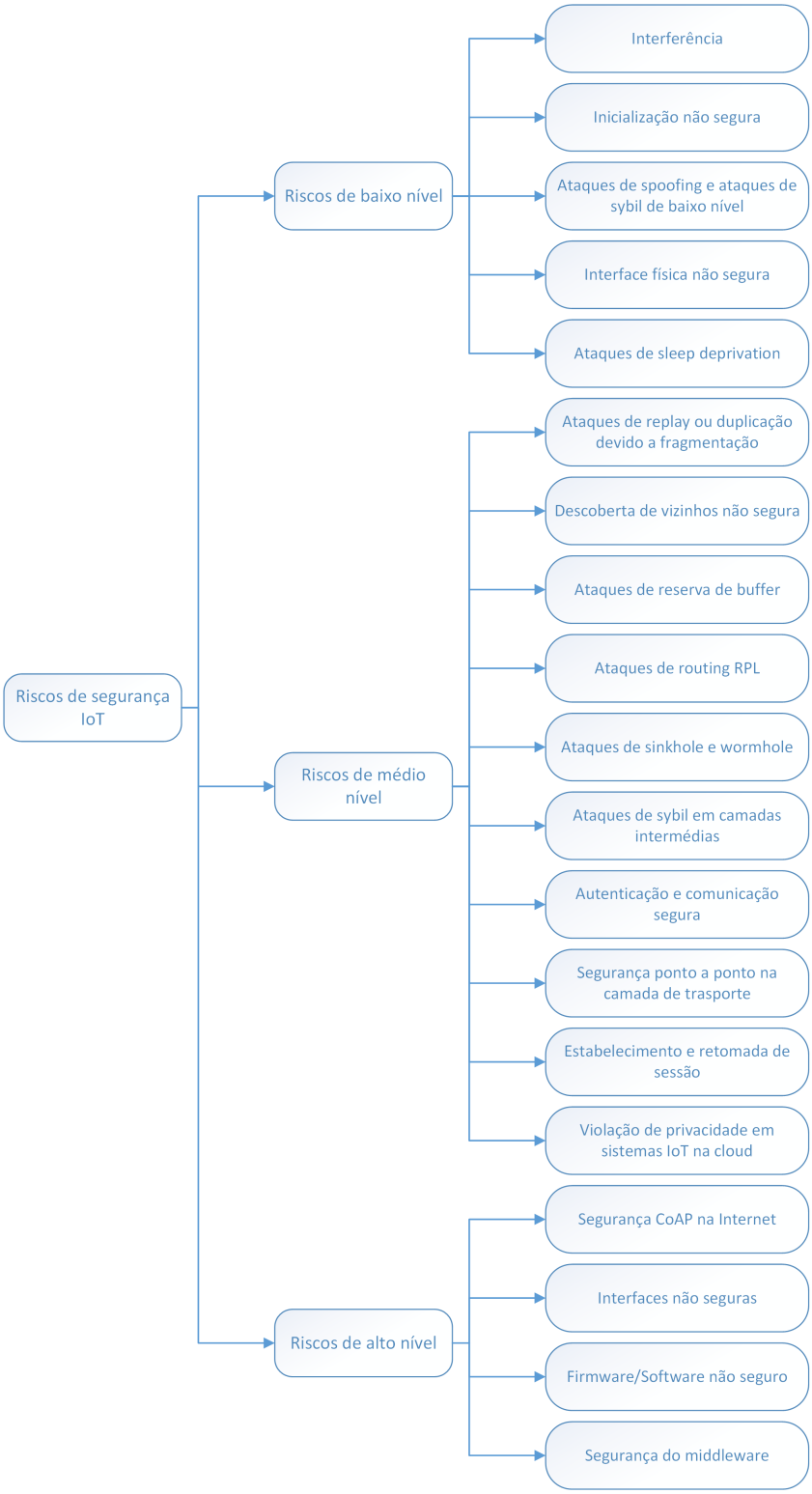


Figura 3: Taxonomia dos riscos de segurança IoT (adaptado do trabalho realizado por Khan e Salah (2017))

tecnologia *blockchain* pode ser utilizada para mitigar vulnerabilidades de segurança IoT, principalmente no que toca ao espaço de endereçamento para dispositivos IoT.

Ao utilizar como espaço de endereçamento de um dispositivo a *hash* Secure Hash Algorithm 1 (SHA1) de 160 bits da chave pública gerada pelo algoritmo de curva elíptica Elliptic Curve Digital Signature Algorithm (ECDSA) empregue pelo *blockchain*, torna-se possível atribuir endereços a $1{,}46 \times 10^{48}$ dispositivos distintos; um valor bastante superior aos $3{,}4 \times 10^{38}$ endereços distintos teóricos permitidos pelo protocolo IPv6 (128 bits).

O uso desta tecnologia seria suficiente para reduzir drasticamente o risco de colisões de endereços e permitiria a atribuição de um Global Unique Identifier (GUID) para cada dispositivo IoT na Internet. Mais, uma vez que se trata de uma tecnologia descentralizada, o *blockchain* removeria a necessidade de uma entidade global que supervisiona e controla a atribuição de endereços Internet Protocol version 4 (IPv4) e IPv6, como acontece atualmente com a Internet Assigned Numbers Authority (IANA), tornando o *blockchain*, em ultima instância, uma solução mais escalável e *future proof* do que a utilizada atualmente.

Inclusive, uma vez que numa rede *blockchain* qualquer transação realizada pode ser autenticada criptograficamente através da sua *hash*, os dados enviados numa rede deste tipo teriam ainda a sua autenticidade e integridade inerentemente garantidas.

Por último, Khan e Salah (2017) fazem um levantamento dos desafios a enfrentar no futuro ao implementar soluções de IoT seguras, entres os principais desafios referidos destacam-se os seguintes:

- **Constrangimento de recursos dos dispositivos IoT**: o facto de um dispositivo ser constrangido, quer seja em termos de processamento, memória, armazenamento ou qualquer outro recurso, pode ser um impedimento no que toca ao uso de protocolos seguros (*e. g.* protocolos criptográficos);
- **Heterogeneidade dos dispositivos**: em cenários IoT nem todos os dispositivos são dotados das mesmas capacidades. Na mesma solução pode ser empregue uma *thing* extremamente limitada (*e. g.* um ESP32) que transmita dados para um servidor de alta disponibilidade. Esta discrepância da capacidade de processamento entre dispositivos, bem como os protocolos e interfaces utilizados por estes para comunicarem entre si, é algo que deve ser tido em conta na fase de planeamento e implementação do projeto;
- **Single Point of Failure (SPOF)**: a heterogeneidade de dispositivos em topologias IoT pode levar a que um único dispositivo comprometa todo o

sistema. Como tal, devem ser tomadas as medidas apropriadas para manter todos os dispositivos o mais seguros possível, especialmente os «elos mais fracos» da topologia.

Existem vários tipos de dispositivos IoT, de seguida será abordada em maior detalhe uma categoria específica de dispositivos IoT denominada de *smart meter* (ou contador inteligente em português) que se encontra na base deste projeto.

1.1.2 *Smart Meters*

Segundo Koponen et al. (2017), um *smart meter* é um dispositivo que monitoriza um ou vários parâmetros (*e. g.* consumo energético, valores de temperatura ou consumo de água) dotado de funcionalidades que o tornam *smart*, tais como:

- Transmissão em tempo real dos dados monitorizados;
- Gestão e processamento de tarifas associadas aos parâmetros monitorizados;
- Leitura automática de consumos para efeitos de faturação ou análise do consumo energético do utilizador;
- Monitorização da qualidade de serviços prestados (*e. g.* monitorização da variação da tensão da rede elétrica). Através de um *smart meter* os utilizadores podem facilmente verificar se a tensão de alimentação se encontra dentro do limite definido no seu local de residência. Em Portugal Continental, em redes de baixa tensão, a tensão pode variar entre os 195.5 e os 253 Volts (respetivamente -15% e +10% da tensão nominal de 230 Volts) (ERSE, 2017) (CENELEC, 2001).

Os *smart meters* podem ainda ser utilizados para identificar e diagnosticar eventuais avarias de equipamentos. A Figura 4 é composta pela representação gráfica de dados capturados por um *smart meter* num caso real. Neste caso em específico, a análise da evolução anómala dos valores da corrente elétrica permitiu identificar uma avaria numa bomba de água que foi prontamente resolvida.

Posto isto, será de seguida feito um enquadramento aos *smart meters* energéticos que monitorizam os consumos energéticos de um utilizador de forma a este poder verificar e aferir a precisão dos valores dos consumos registados na sua fatura energética e, em última instância, estudar e compreender melhor a natureza do seu consumo energético.

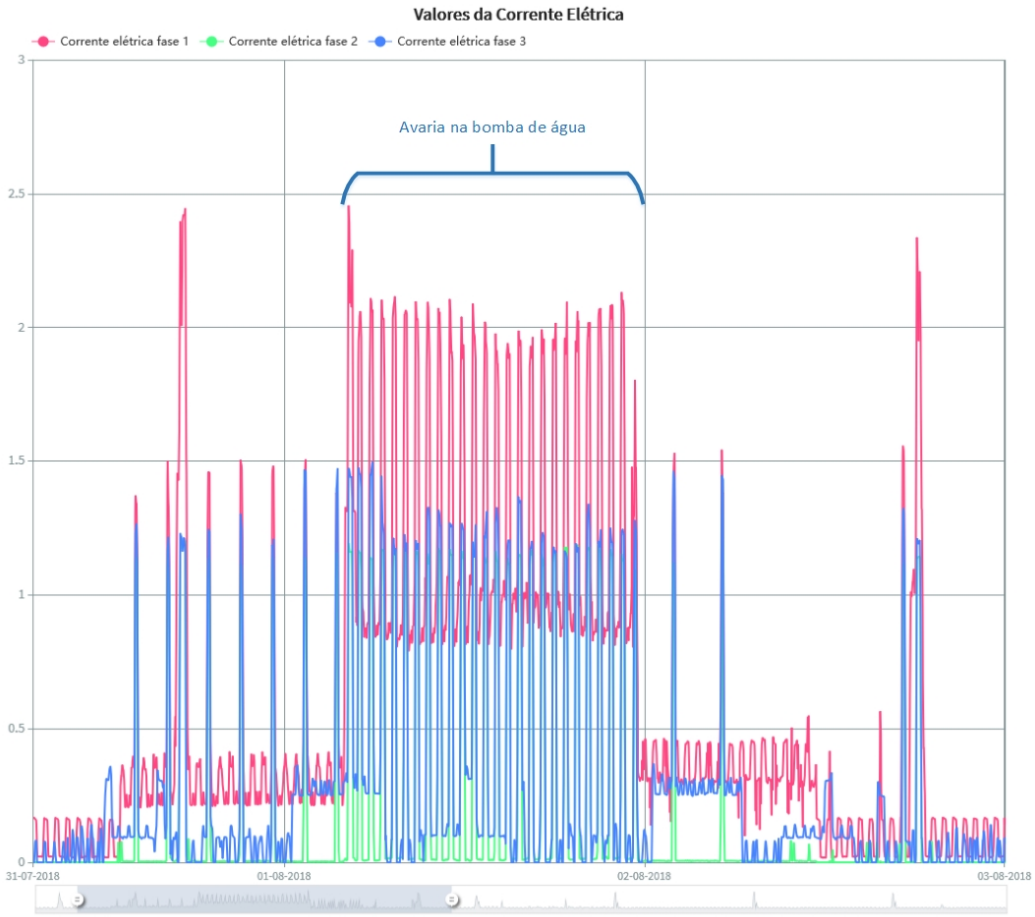


Figura 4: Gráfico da corrente elétrica aquando de uma avaria numa bomba de água (dados retirados de um caso real com base nas leituras de um *smart meter*)

Tabela 1: Preços das tarifas energéticas praticadas pela EDP

Tipo de Tarifa	**Horas do dia**	**Preço (€/kWh)**
Simples	Normal	0.1456
Bi-horária	Normal	0.1875
	Económicas	0.1008
Tri-horária	Normal	0.2720
	Económicas	0.1565
	Super económicas	0.0967

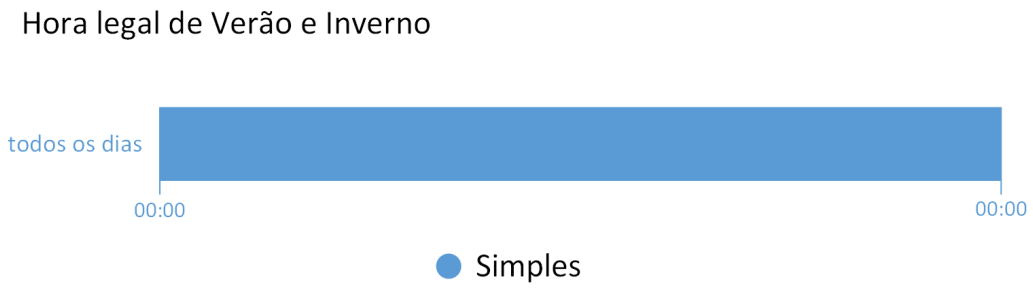


Figura 5: Horário simples praticado pela EDP

Hoje em dia, a maioria dos *smart meters* energéticos permite ao utilizador definir diferentes tipos de tarifas energéticas (simples, bi-horárias e tri-horárias) com diferentes custos para cada uma das alturas do dia (horas de ponta, horas de cheias e horas de vazio), consoante o tipo tarifas a simular.

De forma a contextualizar as tarifas energéticas praticadas em Portugal, foram compilados os preços dos consumos praticados pela EDP (EDP, 2020b) na Tabela 1. Foram também criadas as Figuras 5, 6 e 7 que ilustram os diferentes horários das tarifas energéticas praticados pela EDP (EDP, 2020a).

Existem ainda diferentes tipos de *smart meters* energéticos para diferentes topologias de rede elétrica, mais concretamente, redes monofásica e trifásicas; sendo que, em redes trifásicas, certos *smart meters* conseguem monitorizar cada uma das três fases em separado.

Dentro do universo dos *smart meters* energéticos é possível ainda fazer a distinção entre:

- os *smart meters* distribuidos por fornecedores de energia, tal como acontece em Portugal com a Energias de Portugal (EDP), que enviam remotamente os

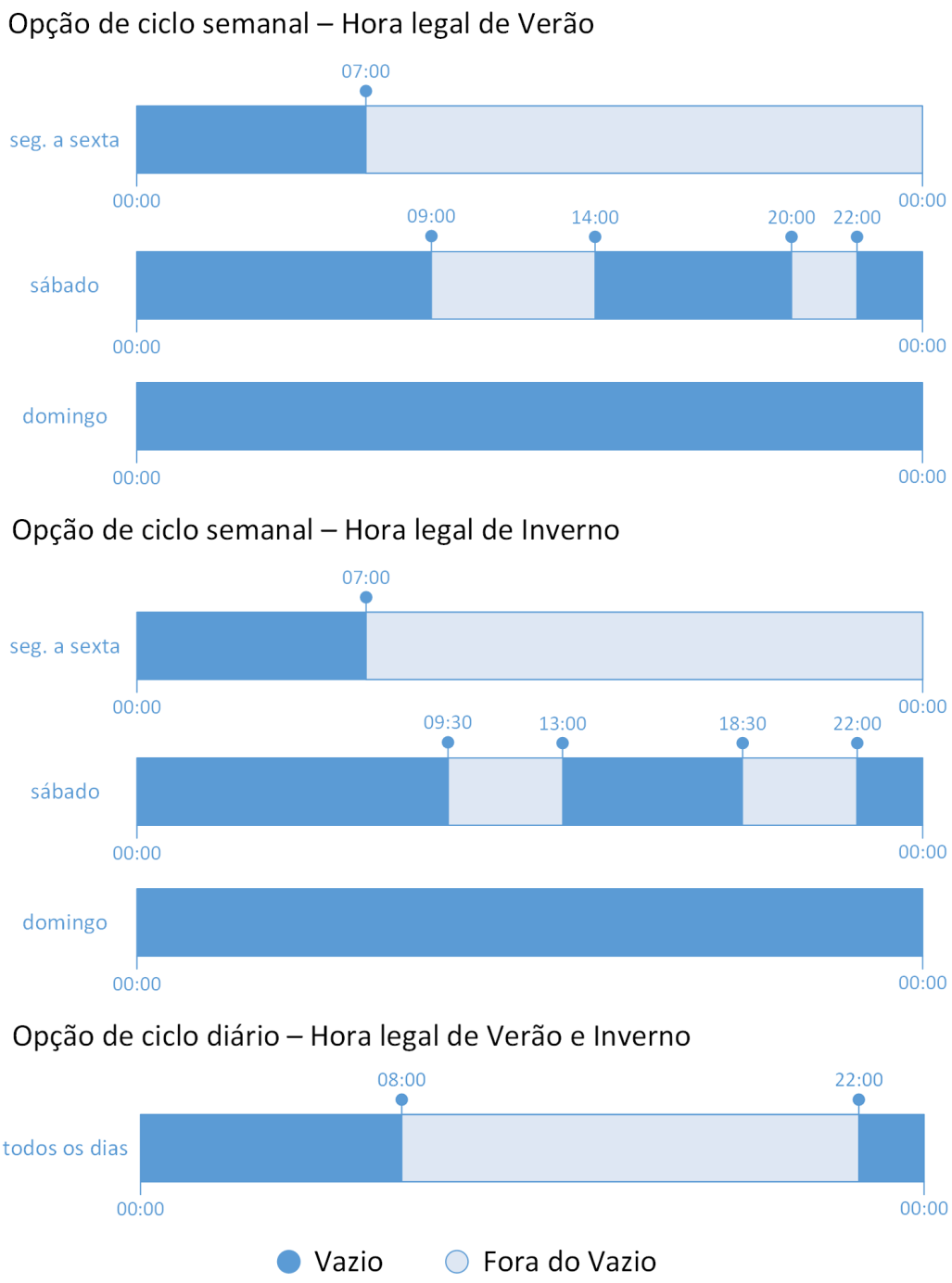


Figura 6: Horários bi-horários praticados pela EDP

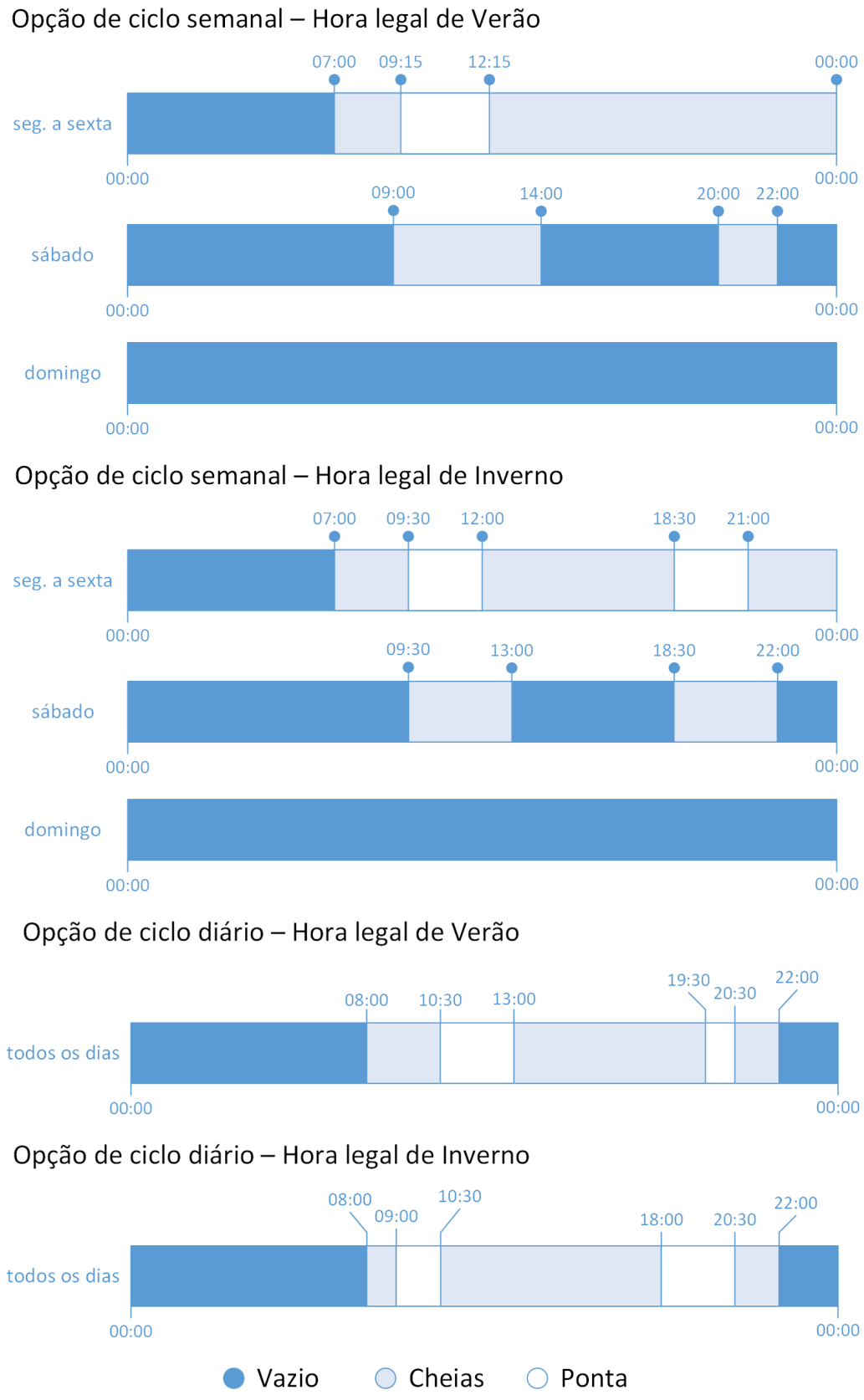


Figura 7: Horários tri-horários praticados pela EDP

dados do consumo energético para a plataforma destes (EDP, 2018) através da tecnologia Powerline (Oborkhale e Shoewu, 2018). Deste modo as leituras do consumo energético do utilizador são feitas remota e automaticamente (quando existe uma infraestrutura de comunicação para o efeito). Para além disto, com estes dispositivos, a fatura energética será baseada nas leituras deste e não em estimativas feitas pelos fornecedores energéticos. A Figura 8 ilustra um dos tipos de *smart meters* disponibilizados pela EDP.

- os *smart meters* utilizados em soluções independentes do fornecedor de energia por entusiastas ou por utilizadores interessados em monitorizar os seu consumos. Assim, através destes dispositivos o utilizador consegue simular, antecipar as suas despesas energéticas mensais, identificar hábitos ou dispositivos menos eficientes do ponto de vista energético ou mesmo diagnosticar eventuais avarias num dispositivo. Este tipo de soluções são caracterizadas por terem um grau precisão geralmente inferior às soluções disponibilizadas pelos fornecedores de energia, apresentando tipicamente erros de medição na casa dos 2% e sendo-lhes como tal atribuído a classe 2 do *standard* IEC/AS 62053 (SATEC, 2014).

  Estes *smart meters* «independentes» podem ser divididos em dois grupos distintos consoante as suas capacidades de comunicação com dispositivos externos:

  1. O primeiro grupo de *smart meters*, ilustrados na Figura 9, é incapaz de comunicar com dispositivos externos. Normalmente estes *smart meters* são dotados de um ecrã e botões com os quais o utilizador pode interagir para consultar os dados armazenados nestes;

  2. O segundo grupo de *smart meters*, ilustrados na Figura 10, envia os dados registados para outros dispositivos, tipicamente servidores de uma plataforma web que armazenam estes dados numa base de dados e permitem ao utilizador, através de um navegador web ou de uma aplicação móvel, analisar em detalhes estes dados, frequentemente com recurso a ferramentas de visualização gráficas tais como gráficos de barras, tabelas ou gráficos circulares.

No trabalho realizado por Asghar et al. (2017) é feita uma análise à privacidade dos dados transmitidos por *smart meters* e é feito um levantamento sumário dos principais desafios de privacidade enfrentados nestas soluções bem como as suas respetivas soluções. Os autores deste trabalho identificaram seis principais desafios de privacidade que serão de seguida enumerados e descritos:

Figura 8: Exemplo de um *smart meter* distribuído pela EDP (podem ser consultados outros tipos de contadores no *website* da CEVE (2017))

Figura 9: Exemplos de *smart meters* sem conectividade à Internet (da esquerda para a direita: Efergy E2 Classic e OWL Micro+)

Figura 10: Exemplos de *smart meters* com conectividade à Internet (da esquerda para a direita: Wibeee (versão trifásica) e emonPi)

- **Resistência a *tampering* dos *smart meters***: caso um atacante tenha acesso físico a um *smart meter*, o dispositivo pode tornar-se vulnerável a uma série de ataques que podem comprometer a privacidade dos dados transmitidos por este. Este tipo de vulnerabilidade é normalmente colmatada quando o *smart meter* é dotado de um Trusted Platform Module (TPM), um micro-controlador que utiliza chaves criptográficas para garantir a segurança do *hardware* do dispositivo;
- **Confidencialidade dos dados**: a confidencialidade dos dados é normalmente garantida através da cifragem destes. Os autores deste trabalho indicam ainda as melhores soluções criptográficas dependendo do nível de confiança do ambiente em que o *smart meter* se encontra instalado: confiável, semi-confiável e não confiável. Para ambientes confiáveis é recomendado o uso de soluções que tenham por base um modelo de Public Key Infrastructure (PKI) para criar um canal de comunicação a partir do qual os dados seriam transmitidos de maneira segura. No entanto em ambientes semi-confiáveis, para além da transmissão segura de dados é também importante assegurar o armazenamento seguro destes. Para tal, Asghar et al. (2017) recomendam a cifragem do volume de armazenamento utilizado pelo *smart meter* em conjunto com protocolos como o Oblivious Random Access Memory (ORAM) e o Private Information Retrieval (PIR) o que pode causar *overhead* computacional. Em ambientes não confiáveis torna-se ainda necessário validar os resultados das operações computacionais realizadas pelos *smart meters*;
- **Consentimento do utilizador**: face aos mais recentes regulamentos publicados pela União Europeia relativos à privacidade dos dados de utilizadores, que impõem que o utilizador aprove o uso dos seus dados pessoais para cada um dos usos distintos que lhes é dado por entidades externas, torna-se necessário informar aos utilizadores de forma clara as várias finalidades para as quais os dados são recolhidos por partes das entidades que parametrizem os dados medidos pelos *smart meters*, como acontece por exemplo no caso da EDP;
- **Controlo de acesso**: para controlar o acesso aos dados neste tipo de soluções os autores recomendam o uso de mecanismos tais como o Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based AccessControl (RBAC) e o eXtensible Access Control Markup Language (XACML);
- **Integridade dos dados**: garantir a integridade dos dados transmitidos através de um modelo PKI pode comprometer a identidade e privacidade dos

utilizadores cujos dados estão a ser transmitidos a entidades externas. Assim, os autores recomendam o uso de soluções de autenticação anónima;

- **Auditoria dos dados**: auditorar os dados enviados pelo utilizador sem comprometer a sua privacidade nem sempre é uma tarefa fácil. As soluções oferecidas pelos autores para esta questão passam, em ambientes confiáveis, pela verificação por parte dos utilizadores dos seus custos de consumos à entidade auditora e, em ambientes semi-confiáveis e não confiáveis, pelo uso de autenticadores homomórficos lineares que tenham como base protocolos de cifragem assimétrica.

No segundo Capítulo deste documento serão analisados e comparados alguns dos *smart meters* energéticos mais populares no mercado atual bem como as plataformas web utilizadas por estes para armazenar e apresentar os seus dados.

1.2 OBJETIVOS

Neste projeto vai ser utilizado um *smart meter* Mirubee (ver Figura 11). Os principais objetivos deste projeto passam por: analisar o funcionamento do Mirubee, identificando os problemas de segurança deste tendo como base o trabalho realizado por Grácio e Faria (2018). Com base nas vulnerabilidades identificadas, é proposta e implementada uma solução alternativa (Ribeiro e Frade, 2020) que permite a monitorizão remota de forma segura utilizando este *smart meter*. Esta solução irá ainda funcionar de maneira a colmatar as vulnerabilidade e falhas de segurança presentes no Mirubee. Mais concretamente, a topologia da solução criada irá isolar o processo de envio de dados do Mirubee (não seguro) e irá utilizar protocolos e *standards* do ramo da cibersegurança para zelar pela confidencialidade e integridade dos dados processados por esta plataforma.

A solução proposta poderá ser hospedada na rede local em que o utilizador se encontra (*e. g.* em sua casa) ou numa plataforma online própria para o efeito (*e. g.* Amazon Web Services (AWS) ou Digital Ocean) e permitirá ao utilizador visualizar e consultar os dados transmitidos pelo seu Mirubee.

Figura 11: *Smart meter* Mirubee utilizado neste projeto

2

TRABALHO RELACIONADO

Esta secção irá debruçar-se sobre trabalhos previamente existentes que foram considerados relevantes para o desenvolvimento deste projeto.

Mais concretamente serão escrutinados os *smart meters* das marcas: Efergy, OWL, EDP, OpenEnergyMonitor e Mirubee; serão enumeradas as funcionalidades bem como as diferenças entre cada um destes dispositivos. Finalmente as funcionalidades de cada um dos dispositivos referidos ao longo deste Capítulo serão sumariamente ilustradas numa tabela.

2.1 EFERGY

A Efergy especializa-se no desenvolvimento e comercialização de monitores de redes elétricas de uso pessoal em tempo real, disponibilizando soluções adequadas para cenários domésticos onde o utilizador pretenda monitorizar e reduzir o seu consumo energético.

Esta marca disponibiliza quatro gamas diferentes de equipamentos, ilustrados na Figura 12, com funcionalidades e objetivos distintos:

- **Efergy Pro**: permite ao utilizador monitorizar tanto a geração de energia solar como o consumo energético de um sistema monofásico. Para tal o utilizador deverá instalar os sensores fornecidos na rede a parametrizar e, através do Efergy Pro Transmitter, transmitir os dados lidos pelos sensores para o Efergy Pro Gateway que irá retransmiti-los por Wi-Fi para a plataforma da Efergy (Efergy, 2018d);
- **Engage**: o Engage funciona de maneira semelhante ao Efergy Pro no entanto, em vez de utilizar o Efergy Pro Gateway para retransmitir os dados recebidos por Wi-Fi, utiliza um *hub* que se liga diretamente ao *router* do utilizador por *ethernet*. Para além disto, o Engage permite ao utilizador monitorizar sistemas trifásicos (Efergy, 2018c);

Figura 12: Gama de equipamentos Efergy (da esquerda para a direita: Efergy Pro, Engage, E2 Classic e Elite Classic)

- **E2 Classic**: os dados do consumo da rede elétrica são transmitidos para um E2 Display, um dispositivo dotado de um ecrã Liquid Crystal Display (LCD), que irá armazenar e exibir em tempo real (através do ecrã) os dados recebidos por este. O utilizador pode ainda, através do E2 Display, definir a voltagem do sistema monitorizado, criar diferentes tarifas, e definir o câmbio a utilizar. Os dados armazenados por este podem ser transferidos para o computador do utilizador, através de um cabo Universal Serial Bus (USB), para posterior consulta (Efergy, 2018a);
- **Elite Classic**: o Elite Classic funciona de modo análogo ao E2 Classic distinguindo-se deste apenas pelo facto de não permitir transferir os dados armazenados por este para o computador do utilizador (Efergy, 2018b).

Estes equipamentos podem ser adquiridos no *website* da Efergy e o seu valor pode variar entre os 55,81 Euros (Elite Classic) e os 162,63 Euros (Efergy Pro) (Efimarket, 2019).

Cada uma das soluções analisadas em cima utiliza uma User Interface (UI) diferente, o Efergy Pro utiliza a aplicação móvel com o mesmo nome, Figura 13, para permitir ao utilizador consultar em tempo real e em qualquer lugar o seu consumo energético e ainda gerir o modo de funcionamento do Efergy Pro Gateway. Já o Engage utiliza um *website* em conjunto com uma aplicação móvel (baseada no website), Figura 14 para permitir ao utilizador consultar os dados recebidos pelo *hub*.

A plataforma web da Efergy utiliza o protocolo Hyper Text Transfer Protocol Secure (HTTPS) proporcionando ao utilizador um nível superficial de segurança adequado ao garantir a confidencialidade dos dados transmitidos e a autenticidade da plataforma.

2.2 OWL

A OWL começou por ser uma marca sediada no Reino Unido de soluções de controlo e parametrização de energia (OWL, 2015a). Com a expansão da Internet of Things, a OWL rapidamente se adaptou e começou a desenvolver *smart meters* capazes não só de parametrizar os dados da rede elétrica mas também de transmiti-los pela Internet para os seus servidores onde estes podem ser consultados pelos utilizadores, em qualquer altura e em qualquer lugar.

Assim sendo, a Owl divide os seus produtos em duas gamas distintas: «Eletricity Monitors» e «Smart Energy Monitors». A primeira gama de produtos é composta pelo OWL Micro+ e OWL+USB, Figura 15. Estes dispositivos são bastante semelhantes aos monitores E2 Classic e Elite Classic da Efergy na medida em que são dotados de um ecrã LCD, capazes de monitorizar os consumos energéticos de uma rede elétrica mas que não transmitem estes dados pela Internet. Tal como acontece com o E2 Classic da Efergy, o OWL+USB permite ao utilizador ligar este dispositivo a um computador e consultar o histórico dos dados armazenados neste em maior detalhe (OWL, 2015b).

Já a gama «Smart Energy Monitors» é composta por três dispositivos que, para além de serem capazes de monitorizar a rede elétrica são também capazes de transmitir estes dados para a plataforma online da OWL (OWL, 2015c), intitulada de Intuition, Figura 16. Estes três dispositivos são:

- **Intuition-e**: este dispositivo permite ao utilizador parametrizar os dados de uma rede monofásica. Este dispositivo, de doze em doze segundos irá, através dos seus sensores, registar os dados do consumo da rede elétrica e enviá-los

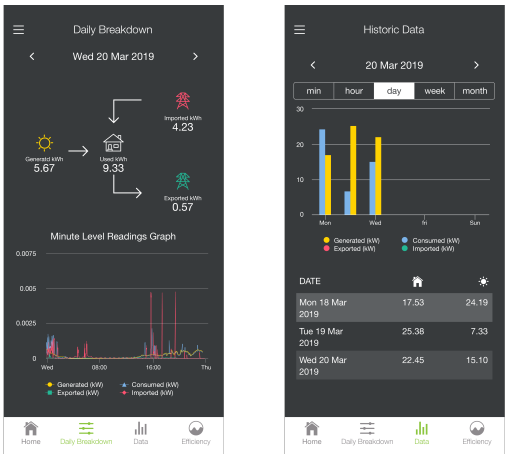


Figura 13: Aplicação móvel Efergy Pro

Figura 14: Plataforma web Engage

Figura 15: Gama «Eletricity Monitors» da OWL (da esquerda para a direita: OWL Micro+ e OWL+USB)

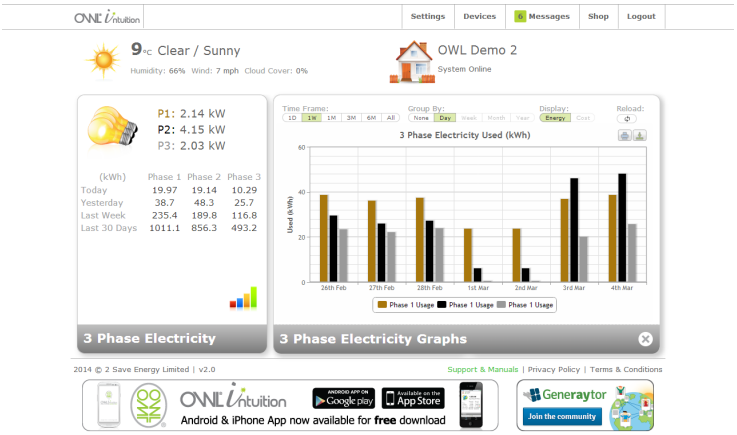


Figura 16: Plataforma web Intuition

Figura 17: Intuituion-lc (versão trifásica)

para a plataforma Intuition. O utilizador pode ainda definir tarifas energéticas personalizáveis de forma a melhor simular os seus custos energéticos;

- **Intuition-lc**: versão trifásica do Intuition-e, Figura 17;
- **Intuition-pv**: para além de todas as funcionalidades presentes nos dois *smart meters* em cima, o Intuition-pv permite ainda monitorizar a geração de energia através de painéis fotovoltaicos.

Em Portugal os monitores da marca OWL podem ser adquiridos através de retalhistas online e o seu valor pode variar entre os 45,99 Euros e os 127,95 Euros dependendo do modelo e dos sensores pretendidos (Efimarket, 2019).

2.3 EDP RE:DY

O EDP Re:dy é a solução desenvolvida pela EDP para permitir aos seus clientes monitorizarem e analisarem os seus consumos energéticos (EDP, 2016).

Para além da monitorização dos consumos energéticos, esta solução permite ainda monitorizar a produção de energia solar e os consumos associados ao carregamento de carros elétricos. Mais, o EDP Re:dy permite aos utilizadores controlar remotamente certos aspetos da sua casa tais como o ar condicionado, caldeiras, sistemas de aquecimento central, bem como sistemas de rega entre outros. O Re:dy permite ainda ao utilizador configurar e automatizar os seus equipamentos, permitindo a este configurar a ativação e desativação automática de equipamentos consoante a a altura do dia e a tarifa energética em vigor, a criação de perfis de consumos, e ainda a configuração de alertas de consumos personalizados.

Para implementar esta solução, o utilizador deverá adquirir e instalar na sua casa o equipamento comercializado pela EDP. O dispositivo principal desta solução trata-se da Re:dy Box, Figura 18, responsável por controlar e atuar como *gateway* dos periféricos Re:dy para a plataforma para onde os seus dados serão transmitidos. A aquisição da Re:dy Box, não tem custos associados, sendo a sua aquisição gratuita aquando da subscrição do utilizador a este serviço o que implica a aquisição de pelo menos três periféricos Re:dy.

Figura 18: EDP Re:dy Box

Existem cincos tipos distintos de periféricos disponibilizados pela EDP ilustrados na Figura 19:

- Re:dy Plug: o Re:dy Plug trata-se de uma tomada inteligente que permite controlar e monitorizar os consumos dos equipamentos ligados a esta. Este dispositivo comunica com a Re:dy Box através do protocolo Zigbee e tem um custo associado de 34 Euros;

- Re:dy Plug solar: semelhante em todos os aspetos ao Re:dy Plug mas desenhado especificamente para sistemas de energia solar. Cada periférico deste tipo custará ao utilizador 59 Euros;
- Re:dy Plug A/C: semelhante em todos os aspetos ao Re:dy Plug mas desenhado especificamente para sistemas de ar condicionado. Cada periférico deste tipo custará ao utilizador 49 Euros;
- Re:dy Meter: através do Re:dy Meter é possível monitorizar até três equipamentos elétricos que não se liguem por tomada à corrente elétrica (*e. g.* bombas de piscina ou sistema de rega). O Meter permite ainda controlar dois equipamentos permitindo ao utilizador ligar ou desligá-los remotamente. este dispositivo deve ser instalado diretamente no quadro elétrico do utilizador, comunica com a Re:dy Box a partir do protocolo PLC Green phy e custará ao utilizador 89 Euros;
- Re:dy Switch: o Re:dy Switch permite controlar um equipamento que não se ligue por tomada à corrente elétrica e tem um custo associado de 39 Euros.

Figura 19: Periféricos Re:dy (da esquerda para a direita: Re:dy Plug e Plug solar, Re:dy Plug A/C, Re:dy Meter e Re:dy Switch)

Após instalar os dispositivos desejados na sua casa, o utilizador poderá utilizar a aplicação móvel associada, ilustrada na Figura 20, para monitorizar e controlar remotamente os seus dispositivos. Para além do custo associado à aquisição dos periféricos desta solução, para usufruir deste serviço, o utilizador terá de pagar também uma subscrição mensal de 3,90 Euros.

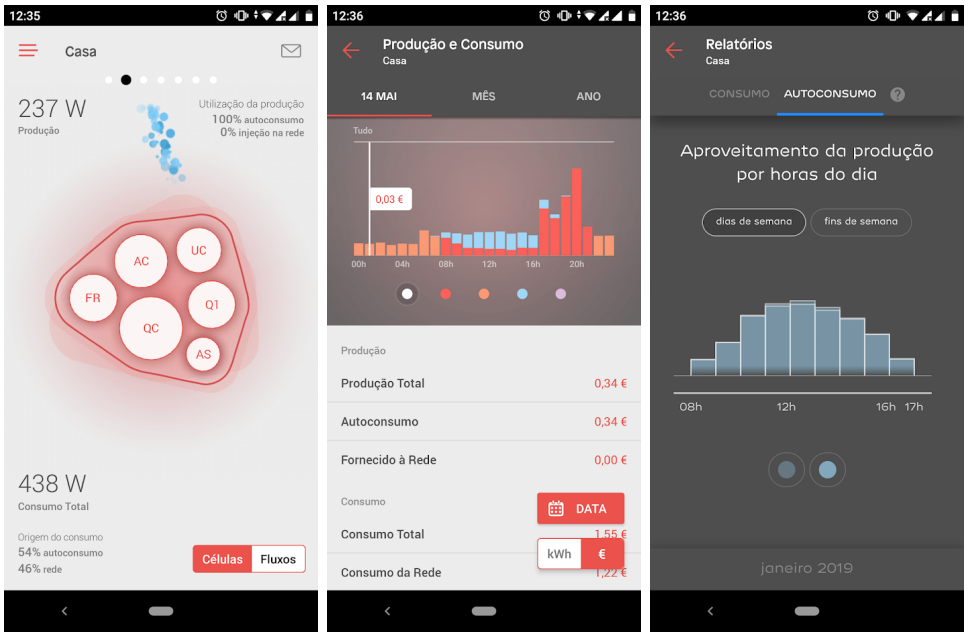


Figura 20: Aplicação móvel EDP Re:dy

2.4 OPENENERGYMONITOR

O OpenEnergyMonitor é um projeto *open source* criado com o objetivo de facilitar a correta monitorização dos mais variados sistemas energéticos que os seus utilizadores possam ter (OpenEnergyMonitor, 2017b).

Para monitorizar em tempo real o consumo energético de uma rede o OpenEnergyMonitor desenvolveu o emonPi, Figura 21. O emonPi utiliza como base o Raspberry Pi que, em conjunto com sensores de corrente alterna, denominados de Current Transformer (CT) *sensors*, Figura 22 regista o consumo energético da rede elétrica em que este se encontra instalado e envia estes dados para a aplicação

Figura 21: EmonPi

Figura 22: Sensor CT

web emonCMS onde estes são armazenados para futura consulta por parte dos utilizadores.

O utilizador pode visualizar o histórico do consumo energético registado pelo emonPi através de um histograma ou de um gráfico de linhas no emonCMS, Figura 23. Uma vez que se trata de um sistema *open source* o emonCMS pode ser *deployed* na rede local do utilizador sem quaisquer encargos associados, o que garante um maior nível de privacidade dos dados armazenados por esta plataforma. Para utilizadores que prefiram aceder remotamente ao emonCMS, o OpenEnergyMonitor também hospeda este serviço através do modelo *pay-as-you-go*.

Ao contrário do que acontece com outros monitores, o emonPi monitoriza principalmente o consumo energético (em Quilowatt-hora (kWh)) da rede elétrica onde este se encontra, não sendo assim possível monitorizar parâmetros tais como: a voltagem, potência ou fator de potência.

A comunicação entre os vários dispositivos de leitura do utilizador e o emonCMS deve ser feito através da Application Programming Interface (API) Representational State Transfer (REST) deste que está apropriadamente documentada no respetivo *website* (OpenEnergyMonitor, 2017a). Esta API disponibiliza aos utilizadores uma série de de *endpoints* a partir dos quais estes podem enviar os dados lidos pelos seus dispositivos e ainda aceder aos dados armazenados por este serviço. A autenticação do utilizador através desta API deve ser feita a partir de duas chaves geradas previamente pelo servidor: a primeira chave gerada permite a leitura de dados armazenados na plataforma enquanto que a outra chave permite a leitura e escrita

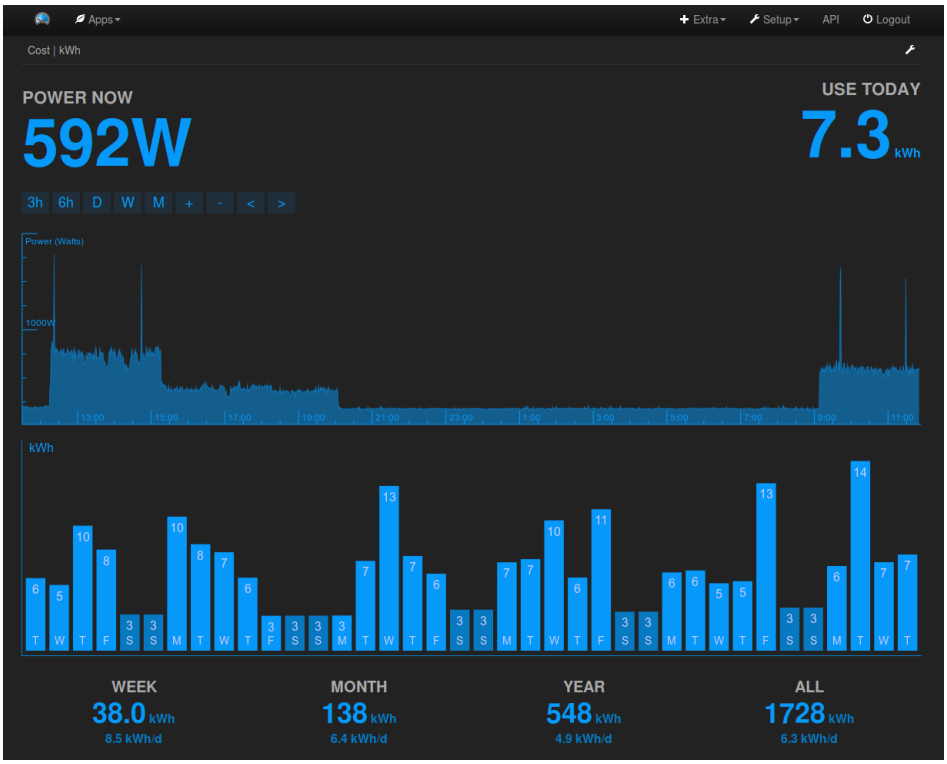


Figura 23: Dashboard emonCMS

de dados na plataforma. Dependendo do tipo de pedido realizado as chaves API podem ser utilizadas de três maneiras distintas:

- em pedidos GET a chave deve ser adicionada ao Uniform Resource Locator (URL) («...&apikey=APIKEY»);
- em pedidos POST deve ser adicionado ao corpo deste o parametro «apikey» juntamente com o valor da chave indicada;
- a inclusão da chave API pode ainda ser feita através do cabeçalho Hypertext Transfer Protocol (HTTP) do pedido realizado («Authorization: Bearer APIKEY»).

O emonCMS está preparado para utilizar um certificado Secure Sockets Layer (SSL) o que, em conjunto com o protocolo HTTPS, se traduz num sistema relativamente robusto no que toca à segurança dos dados enviados e recebidos pelos utilizadores nesta plataforma. Mais, se por algum motivo o utilizador decida não utilizar o protocolo HTTPS, o emonCMS disponibiliza ainda uma solução de cifragem da camada de transporte através da chave API que é utilizada (como chave pré-partilhada) pelo algoritmo Advanced Encryption Standard (AES)-128-Cypher Block Chaining (CBC) para cifrar os dados enviados.

Figura 24: Mirubee

O EmonPi encontra-se à venda no *website* oficial do OpenEnergyMonitor por 201,87 Euros podendo ainda ser adquirido em conjunto com os respetivos sensores da corrente alterna necessários por 211,47 Euros (OpenEnergyMonitor, 2017c).

2.5 MIRUBEE

O Mirubee, Figura 24, é um *smart meter* desenvolvido pela Mirubee, uma *startup* espanhola que se dedicava à monitorização de consumos energéticos e à criação de perfis de utilização de cada aparelho ligado à rede elétrica, através de *machine learning*.

Desde de então, a Mirubee foi adquirida pela empresa Smilics que se dedica maioritariamente ao desenvolvimento de equipamento de monitorização e controlo de energia elétrica com o objetivo de melhorar a eficiência energética dos utilizadores dos seus clientes (Smilics, 2016). O Mirubee foi ainda licenciado à empresa Circutor (Circutor, 2015a) por parte da Smilics.

Hoje em dia, tanto a Smilics como a Circutor disponibilizam a sua versão do Mirubee, atualmente denominado de Wibeee, bem como as suas plataformas web e aplicações móveis, que são bastante semelhantes.

Através do Mirubee, o utilizador pode registar, e posteriormente consultar e analisar, os dados do consumo da rede elétrica onde este se encontra instalado (Circutor, 2015b). Para tal o utilizador deverá instalar o Mirubee na rede elétrica a monitorizar e configurar o seu acesso à Internet, através da aplicação móvel, de forma a que este consiga transmitir os dados do consumo energético para os servidores da Circutor na *cloud*.

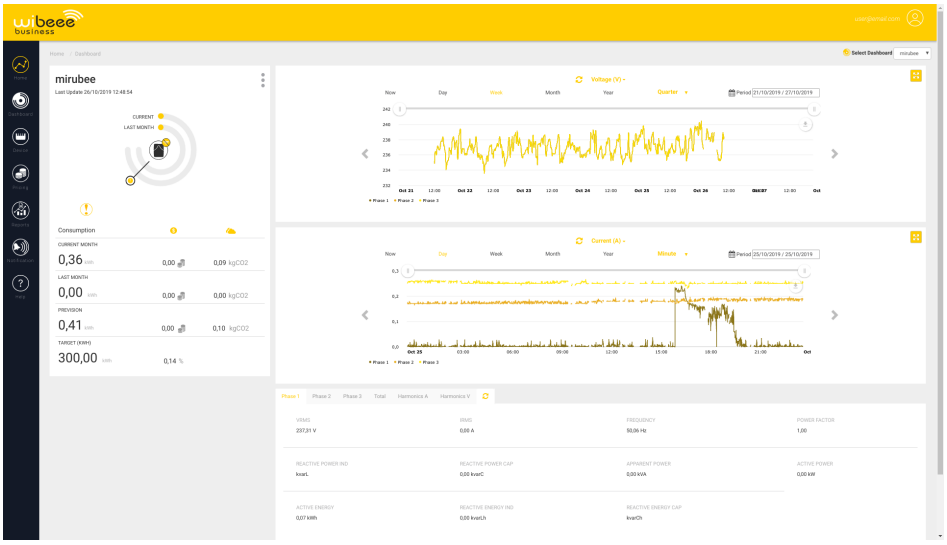


Figura 25: Dashboard Smilics

Ao aceder à plataforma online o utilizador pode consultar o histórico dos dados registados pelo Mirubee através de ferramentas de visualização gráficas tais como gráficos de linhas e tabelas, conforme ilustrado na Figura 25. A partir desta plataforma o utilizador pode consultar os parâmetros capturados pelo Mirubee, que faz o cálculo em tempo real de certos parâmetros mais complexos da rede elétrica tais como: a potência ativa, o fator de potência ou a energia ativa que serão abordados em maior detalhe no Capítulo 3, «Análise do Mirubee».

Na plataforma online, quer da Smilics quer da Circutor, é possível ainda definir tarifas energéticas personalizadas de maneira a simular os custos mensais face aos consumos medidos, gerar relatórios, consultar alertas gerados pela plataforma e ainda gerir os vários dispositivos ligados a esta.

2.6 TABELA COMPARATIVA

A Tabela 2, em baixo, ilustra resumidamente as diferenças entre os vários *smart meters* abordados ao longo deste Capítulo.

É possível observar que a todas as marcas analisadas disponibilizam pelo menos uma solução de monitorização trifásica, enquanto que apenas a Efergy e a OWL oferecem soluções exclusivamente monofásicas. É conveninente referir que apenas uma minoria dos dispositivos abordados não suporta as funcionalidades de «Monitorização Remota» e de uso de «Ferramentas Gráficas», no entanto existe pelo menos uma solução de cada marca que suportam estas funcionalidades.

No entanto, dos onze *smart meters* escrutinados, apenas os da marca Mirubee registam métricas elétricas tais como a Potência Reativa ou a Energia Ativa.

Tabela 2: Tabela comparativa entre *smart meters*. **Monitorização remota**: permite ao utilizador consultar os dados registados remotamente; **Suporte Trifásico**: permite monitorizar as três fases individuais de sistemas trifásicos; **Ferramentas Gráficas**: permite ao utilizador visualizar os dados registados através de gráficos; **Métricas Avançadas**: regista métricas da rede elétricas tais como a Potência Ativa, Potência Reativa, Fator Potência, *etc*

Smart Meters	Monitorização Remota	Suporte Trifásico	Ferramentas Gráficas	Métricas Avançada
Efergy Pro	✓	×	✓	×
Efergy Engage	✓	✓	✓	×
E2 Classic	×	✓	✓	×
Elite Classic	×	✓	×	×
Owl Micro+	×	✓	×	×
OWL+USB	×	✓	✓	×
Intuition-e	✓	×	✓	×
Intuition-lc	✓	✓	✓	×
Intuition-pv	✓	✓	✓	×
EDP Re:dy	✓	✓	✓	×
EmonPi	✓	✓	✓	×
Mirubee	✓	✓	✓	✓

Posto isto foi decidido que, neste projeto seria utilizado o Mirubee para registar os consumos da rede elétrica visto que este tem praticamente todas as funcionalidades presentes nas soluções das outras marcas e ainda devido a este ser o único *smart meter* disponível para o efeito.

3

ANÁLISE DO MIRUBEE

Neste Capítulo serão apresentados os resultados e conclusões derivados da análise realizada ao Mirubee e à plataforma associada a este. Será com base nas vulnerabilidades enumeradas neste Capítulo que será feito o levantamento de funcionalidades e mecanismos de segurança da plataforma desenvolvida ao longo deste projeto.

3.1 ENGENHARIA REVERSA DA APLICAÇÃO MÓVEL WIBEEE

Com base no trabalho de engenharia reversa da aplicação móvel Wibeee realizado por Grácio e Faria (2018), será feito um levantamento das principais vulnerabilidades de segurança desta:

- **Cálculo da *hash* da *password* do utilizador**: a aplicação móvel Wibeee, de forma a proteger a *password* introduzida pelo utilizador, calcula a sua *hash* recorrendo aos algoritmos de *hashing* Message Digest Algorithm 5 (MD5) e SHA1. Ambos os algoritmos utilizados foram, ao longo do tempo, sendo substituidos por soluções mais seguras, existindo hoje em dia alternativas mais robustas para a cifragem de *passwords* tais como o argon2 que, para além de ser um algoritmo computacionalmente mais lento permite definir um *salt* que será utilizado durante o *hashing* das *passwords*. Um atacante, ao saber os algoritmos utilizados pode tentar realizar o *lookup* reverso das *hashes* podendo eventualmente chegar à *password* introduzida pelo utilizador originalmente. Ao realizar o *salting* das *password*, o argon2 invalida a criação e utilização de «*rainbow tables*» utilizadas para realizar o *lookup* reverso de *passwords* e em ataques de dicionário;
- **Armazenamento da *hash* da *password* após o *logout***: quando o utilizador termina a sua sessão na aplicação móvel, esta regista uma nova «`SharedPreference`» (com a chave valor «`fromLogout`»: «`1`») que, dependendo do seu valor servirá como *flag* para alertar a aplicação que esta deve preencher os campos de autenticação com as credenciais armazenadas previamente. No entanto, esta abordagem não limpa os dados de autenticação do

utilizador, mesmo após este terminar a sua sessão, permitindo que possíveis atacantes tentem aceder a estes dados e quebrem a segurança da autenticação da aplicação;

- **Semelhanças entre as aplicações Circutor e Smilics**: tal como foi referido anteriormente, a Smilics licenciou à Circutor os direitos de comercialização do Mirubee, licenciando também o código fonte da aplicação web e da aplicação móvel. A partilha do código fonte resulta na existência de duas aplicações móveis bastante semelhantes entre si, uma associada à plataforma da Smilics e a outra associada a plataforma da Circutor. Através dos *scripts Androguard* e *Androsim* foi aferido um grau de semelhança entre as duas aplicações de 99.936003% permitindo que o mesmo dispositivo esteja associado a contas diferentes o que pode resultar em falhas de privacidade caso o proprietário original se queira desfazer do dispositivo;

Podemos concluir que, com base nestes três pontos, que a aplicação móvel Wibeee deixa muito a desejar no que toca à segurança dos dados processados por esta. Com efeito, no caso da encriptação da *password* do utilizador, a aplicação móvel utiliza mecanismos de segurança desatualizados enquanto que os restantes pontos demonstram desleixo por parte dos desenvolvedores da aplicação pela segurança dos dados tratados por esta.

3.2 ANÁLISE DE SEGURANÇA DAS PLATAFORMAS ASSOCIADAS

Apesar de se tratar de uma solução prática e eficaz no que toca à medição e análise dos dados da rede elétrica, o Wibeee, bem como as plataformas web associadas, apresentam uma série de lacunas básicas do ponto de vista da cibersegurança, tais como:

- **A não utilização de um certficado SSL**: Apesar de recentemente a plataforma Smilics ter implementado o uso de um certificado SSL no seu *website* este certificado não é utilizado para o resto da plataforma (*e. g.* no envio das leituras energéticas do Mirubee); por outro lado no caso da Circutor, nem sequer o *website* utiliza um certificado SSL.

  O facto do servidor destas plataformas nem sempre utilizar um certificado SSL implica a utilização do protocolo HTTP para aceder a certos conteúdos hospedados por este (em prole da versão segura do mesmo, o protocolo

HTTPS). Com efeito, esta é uma das principais vulnerabilidades da plataforma e é responsável pela maioria dos restantes pontos enumerados de seguida;

- **Envio de dados em *cleartext***: uma vez que estas plataformas não suportam o protocolo HTTPS no acesso a certos recursos nem cifram os dados transmitidos por estas, os utilizadores deste serviço não têm alternativa senão enviar e receber estes dados em *cleartext*; rescindindo assim da sua confidencialidade ou privacidade uma vez que os pacotes trocados entre cliente e servidor podem facilmente ser capturados e analisados por entidades externas.

  Esta vulnerabilidade torna-se ainda mais flagrante no envio das credenciais de autenticação do cliente na plataforma Circutor. Neste caso, para tentar proteger a *password* introduzida pelo utilizador, é calculada a sua *hash* de maneira semelhante à aplicação móvel e é enviado, em *cleartext*, o resultado final deste processamento ao servidor;

- **Não autenticação do servidor**: Ao aceder aos *endpoints* não protegidos por HTTPS, é impossível ao utilizador aferir a autenticidade deste face a uma Certificate Authority (CA). Facto este que leva a que os utilizadores desta plataforma sejam potenciais alvos de um ataque de *phishing* ou de Man-In-The-Middle (MITM);

- **Não autenticação dos dispositivos de medição**: o Wibeee, quando transmite os dados da rede elétrica para o servidor, limita-se fazer pedidos GET, conforme ilustrado na Listagem 1, com o seu conteúdo enviado em *cleartext* sem qualquer tipo de autenticação prévia.

Listagem 1: Exemplo de pedido GET enviado por um Mirubee trifásico

```
1 http://wibeee.smilics.com/Wibeee/receiverAvg?mac=001ec03635e2&ip=192.168.001.100
  ↪ &soft=3.4.614&model=WB3&time=1585817562&v1=234.23&v2=234.23&v3=234.23&vt=234
  ↪ .23&i1=0.34&i2=0.17&i3=0.24&it=0.75&p1=81&p2=39&p3=56&pt=177&a1=43&a2=0&a3=0
  ↪ &at=42&r1=0&r2=0&r3=0&rt=0&q1=50.20&q2=50.20&q3=50.20&qt=50.20&f1=0.528&f2=0
  ↪ .000&f3=-0.003&ft=0.240&e1=111&e2=2392&e3=1006&et=3509&o1=0&o2=0&o3=0&ot=0&p
  ↪ s=123&t1t=0.0&t11=0.0&t13=0.0&t15=0.0&t17=0.0&t19=0.0&t2t=0.0&t21=0.0&t23=0.
  ↪ 0&t25=0.0&t27=0.0&t29=0.0&t3t=0.0&t31=0.0&t33=0.0&t35=0.0&t37=0.0&t39=0.0
```

O pedido GET em cima foi capturado através de um *proxy* HTTP que captura o pedido original enviado pelo Mirubee. Este *proxy* é um dos componentes fundamentais da plataforma desenvolvida no âmbito deste projeto, pelo que a sua implementação e funcionalidades serão abordadas em maior detalhe no Capítulo 6 deste relatório. Podemos constatar que, devido ao facto de se tratar de um pedido GET toda a informação transmitida por este se encontra presente no URL (*e. g.* os valores da voltagem registada, representados pelo

campos «`v1`», «`v2`» e «`v3`» um para cada fase monitorizada) facilitando ainda mais a captura e análise indevida deste dados. A não autenticação por parte dos dispositivos de medição significa ainda que qualquer entidade pode enviar para o servidor desta plataforma pedidos GET desde que sigam a mesma estrutura de dados, podendo assim facilmente enviar dados falsos para a conta de outros utilizadores;

- **Vinculação dos *smart meters* aos utilizadores**: esta plataforma vincula o endereço Media Access Control (MAC) do *smart meter* Mirubee à conta de qualquer utilizador que tenha enviado leituras a partir deste. Deste modo, caso o utilizador original empreste ou venda o seu Mirubee a um segundo utilizador, o utilizador original irá continuar a receber notificações por email relativas às leituras do segundo utilizador.

  Esta vulnerabilidade pode ser facilmente explorada por um utilizador malicioso que, ao emprestar o seu Mirubee a um utilizador alvo, passa a receber notificações da utilização da rede elétrica deste, podendo analisar os seus hábitos quotidianos e em última instância passar a tentar perceber quando é que este se encontra ou não em casa, esta potencial vulnerabilidade será explorada em maior detalhe no Capítulo 4 deste documento.

Esta lista de potenciais riscos e vulnerabilidades de segurança permite chegar à conclusão que a falta de cuidado no que toca à segurança dos dados processados é sistemática, podendo ser observada tanto na plataforma como na aplicação web tratando-se de uma constante no ecossistema Wibeee.

3.3 MÉTRICAS CALCULADAS PELO MIRUBEE

Conforme referido anteriormente, O Mirubee distingue-se de outras alternativas semelhantes ao fazer o cálculo em tempo real de certas métricas da rede elétrica. De seguida será feita a enumeração e análise de todas as métricas transmitidas pelo Mirubee:

- **Voltagem**: mede a tensão (ou diferença de potencial) elétrica em Volts entre dois pontos da corrente. Através do Mirubee é possível parametrizar a tensão da corrente elétrica onde este se encontra instalado e verificar se esta se encontra dentro de um limite aceitável;

- **Corrente**: mede, em Amperes, o fluxo de eletrões da corrente elétrica. A corrente elétrica é medida pelo Mirubee através da interação entre as pinças deste e o campo eletromagnético gerado pelo cabo fase;
- **Potência Ativa**: também denominada de potência útil, mede em Watts a potência utilizada por um equipamento para realizar trabalho útil (Boylestad, 2015);
- **Potência Reativa (Indutiva e Capacitiva)**: potência que não é utilizada para realizar trabalho útil, ocorre quando a voltagem e a corrente não estão em fase e se verifica um desfasamento entre os dois. A potência reativa pode ser indutiva (quando a corrente apresenta um atraso em relação à tensão) ou capacitiva (quando a corrente se adianta em relação à tensão) (Boylestad, 2015);
- **Potência Aparente**: potência total utilizada. Calculada com base na potência ativa e na potência reativa (Potência Aparente² = Potência Ativa² + Potência Reativa²) (Boylestad, 2015);
- **Frequência**: a frequência, em Hertz, regista o número de vezes que a corrente alternada inverte o sentido da sua polaridade num sentido e volta a invertê-lo para o sentido original num segundo. Em Portugal a frequencia da rede elétrica é de 50 Hertz;
- **Fator de Potência**: o rácio entre a potência ativa e a potência aparente. Este rácio pode variar entre -1 e 1 dependendo do desfasamento entre a corrente e a tensão (Boylestad, 2015);
- **Harmónicos de Tensão e de Corrente**: regista a distorção no sinal da Tensão ou da Corrente elétrica;
- **Energia Ativa**: mede, em kWh, a energia que foi consumida para realizar trabalho útil;
- **Energia Reativa**: mede, em kWh, a energia que foi consumida para realizar trabalho não útil.

A monitorização destas métricas pode revelar-se uma funcionalidade fundamental para certos utilizadores interessados numa análise mais granular da rede elétrica e diferencia do Mirubee da maioria dos outros *smart meter* disponíveis ao público.

4

ANÁLISE DE PADRÕES DE CONSUMO

Este Capítulo servirá para validar o conceito de analisar os padrões de consumo da rede eléctrica de um utilizador para inferir os seus hábitos quotidianos e as implicações que esta análise pode vir a ter na privacidade dos utilizadores de *smart meters*.

O trabalho realizado por Cuijpers e Koops (2012) refere o estudo encomendado pela Associação de Consumidores Holandeses acerca do potencial impacto que a implementação obrigatória de sistemas de *smart metering* poderia causar na privacidade dos seus utilizadores. Este estudo refere não só o possível risco de segurança causado pela recolha de leituras energéticas, uma vez que com base nestas seria possível derivar os estilos de vida dos moradores de uma habitação, mas também destaca as obrigações de segurança impostas aos gestores da rede elétrica que passariam a receber os relatórios energéticos gerados pelos *smart meters* instalados nas habitações dos cidadãos holandeses.

Os possíveis riscos de segurança levantados por este estudo foram, em parte, responsáveis pela reformulação do projeto de lei proposto pelo governo dos Países Baixos em relação à implementação a nível nacional de *smart meters* nas habitações holandesas. Assim, este projeto de lei passou a acomodar um maior leque de escolhas aos cidadãos holandeses com o intuito de zelar pela privacidade destes.

Segundo Erdemir et al. (2020), é ainda possível determinar as «assinaturas elétricas» de eletrodomésticos específicos usando técnicas de monitorização de carga não intrusivas (do inglês: Nonintrusive Load Monitoring (NILM)). Este tipo de análise ainda mais granular dos hábitos de consumo do utilizador permite aos fornecedores de energia ou mesmo a entidades externas com acesso a estes dados compreender ainda melhor os hábitos quotidianos do utilizador em tempo real. Os autores referem também que estas potenciais violações da privacidade dos utilizadores podem vir a ser um dos maiores entraves no que toca à adoção generalizada de *smart meters* a nível mundial.

Este artigo faz ainda o levantamento de várias técnicas que têm como objetivo zelar pela privacidade dos hábitos quotidianos dos utilizadores destes dispositivos de monitorização. Segundo o artigo estas técnicas podem ser divididas em dois grupos:

1. Manipulação de dados: envolve o uso de técnicas de ofuscação dos dados (ao adicionar ruído), agregação de dados (ao agregar várias leituras em simultâneo e cifrá-las), anonimização dos dados (ao não utilizar diretamente a identidade dos utilizadores) ou ainda técnicas de *downsampling* dos dados gerados pelos smart *meters*
2. Manipulação da procura energética: estas técnicas envolvem o uso de fontes de energia renováveis ou de baterias recarregáveis de forma a garantir que os eletrodomésticos do utilizador apenas usem parcialmente a rede elétricas ofuscando deste modo o comportamento energético do utilizador.

Apesar de garantirem uma maior privacidade dos hábitos quotidianos do utilizador, ambos os grupos de técnicas referidas em cimas apresentam desvantagens claras no que toca à normal monitorização dos consumos energéticos do utilizador uma vez resultam na adulteração das leituras originalmente geradas pelos *smart meters*

No âmbito da análise realizada neste Capítulo, foram processados os dados dos últimos dois anos (entre julho de 2018 e julho de 2020) registados por um *smart meter* Mirubee instalado no quadro elétrico de uma habitação cujas características são apresentadas na Tabela 3.

Tabela 3: Especificações da habitação monitorizada

Tipo de Habitação	Composição do Agregado Familiar	Potência Contratada (kVA)
Moradia familiar	4 pessoas	10,35 KVA, trifásico, bi-horário

Uma vez que os dados processados ao longo deste Capítulo se encontravam armazenados na plataforma da Circutor e que esta plataforma por omissão não permite a recolha em massa das suas leituras com uma granularidade de minuto a minuto, tornou-se necessário compreender o tipo de pedido aceite pela plataforma para obter os dados pretendidos. Através das ferramentas de desenvolvedor do navegador web Firefox foi identificado e exportado (em formato cURL) um dos pedidos realizados pela página web da Circutor para aceder aos dados hospedados na sua plataforma. Verificou-se então que estes pedidos eram compostos por quatro campos:

1. **`datetime_ini`**: a data de inicio da recolha de dados (*e. g.* «01/01/2020 00:00:00»);
2. **`datetime_fin`**: a data de fim da recolha de dados (*e. g.* «07/01/2020 23:59:59»);

Listagem 2: Excerto do ficheiro CSV gerado com base nas leituras da voltagem

```
V1;V2;V3;data;Vt
233.282;233.325;232.371;2018-07-01T00:01:00Z;232.993
233.394;233.809;232.466;2018-07-01T00:02:00Z;233.223
234.288;233.853;229.093;2018-07-01T00:03:00Z;232.412
228.875;233.978;232.336;2018-07-01T00:04:00Z;231.730
229.401;233.938;232.254;2018-07-01T00:05:00Z;231.864
233.264;233.577;229.238;2018-07-01T00:06:00Z;232.027
233.411;232.870;227.981;2018-07-01T00:07:00Z;231.421
234.403;232.264;228.862;2018-07-01T00:08:00Z;231.843
232.379;232.600;229.493;2018-07-01T00:09:00Z;231.491
```

3. **`measure_type`**: a métrica pretendida;
4. **`granularity`**: a granularidade pretendida (*e. g.* «minute», «quarter», «hour», «day»).

Uma vez que era pretendida uma granularidade de minuto a minuto entre leituras e que por omissão a página web utiliza uma granularidade diária (quando o intervalo entre a data de início e a data de fim assume um valor superior a três meses), Este pedido foi importado para a ferramenta Postman para facilitar a manipulação dos pedidos enviados.

Deste modo, foram recolhidos os dados (em formato JSON) de minuto a minuto para seis métricas distintas da rede elétrica (energia ativa, voltagem, corrente elétrica, potência ativa, potência aparente e frequência) sendo que cada um dos registos destas métricas é composto por quatro valores (fase 1, fase 2, fase 3 e o total das três fases sumadas). Estes dados foram de seguida convertidos para formato CSV e processados através da ferramenta RapidMiner desenvolvida especificamente para tratar, processar e analisar um grande volume de dados. Com efeito foram processados 1 052 640 registos por cada métrica analisada.

A Listagem 2 ilustra o conteúdo do ficheiro CSV gerado com base nas leituras da voltagem, realizadas pelo Mirubee, que foram processadas pelo RapidMiner para este projeto.

Uma vez importados e validados os dados pelo RapidMiner, foi feita uma análise inicial da média do consumo energético do utilizador em função da hora do dia. Para calcular os consumos horários de cada dia, os registos da energia ativa foram agrupados por hora para cada dia distinto analisado e foi calculado o seu somatório, sendo de seguida calculada a média horária destes somatórios. O resultado encontra-se ilustrado na Figura 26, nesta encontram-se discriminadas as média do consumo da energia ativa registada pelo Mirubee ao longo dos dois anos analisados para a Fase 1, Fase 2, Fase 3 e para o Total das três fases. As barras de erro presentes

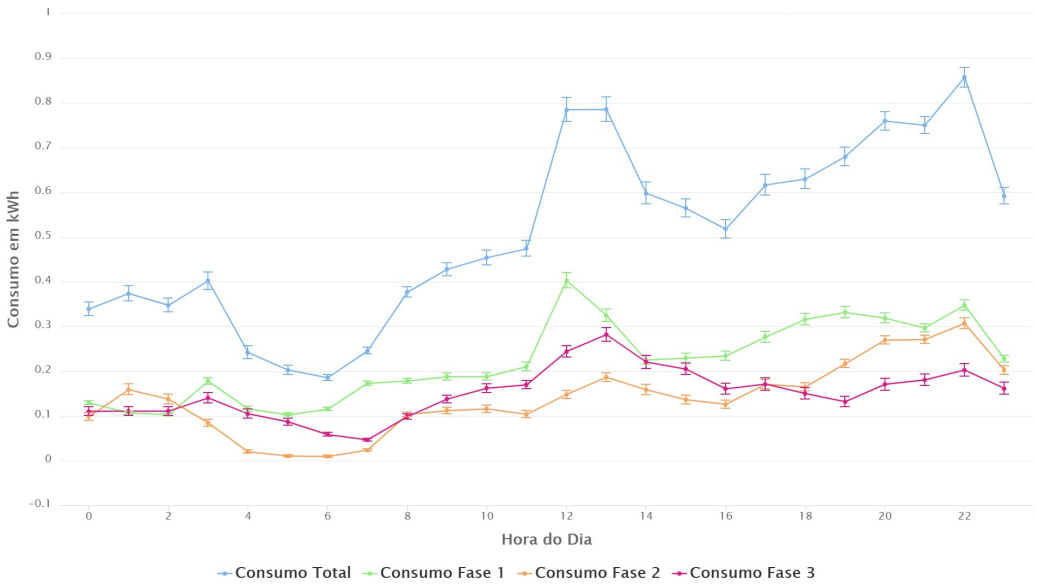


Figura 26: Gráfico da média de consumo energético ao longo do dia

neste gráfico, bem como nos restantes gráficos deste Capítulo, são representativas do erro padrão das médias e foram calculadas através da seguinte fórmula:

$$Erro\ padrão\ da\ média = \frac{Desvio\ Padrão\ da\ amostra}{\sqrt{Tamanho\ da\ amostra}}$$

É possível constatar com base na figura 26 que mesmo sem grande refinamento dos dados é possível retirar algumas ilações acerca dos hábitos quotidianos do utilizador:

1. Com base no aumento gradual do consumo energético a partir das oito horas da manhã é razoável assumir que o utilizador começa o seu dia a partir desta hora;
2. O pico assistido entre as doze e treze horas pode ser um bom indicador de que não só o utilizador almoça a estas horas mas também que este almoça regularmente em sua casa;
3. O aumento do consumo assistido a partir das dezassete horas pode ser causado pelo regresso a casa por parte do utilizador e consequente utilização de aparelhos ligados à rede elétrica deste;
4. Após as vinte e duas horas observa-se uma queda brusca no consumo energético do utilizador provavelmente relacionado com a hora de deitar deste.

Para além desta análise foi também possível calcular que a média de consumo diário durante os dois anos analisados foi de 12,203 kWh.

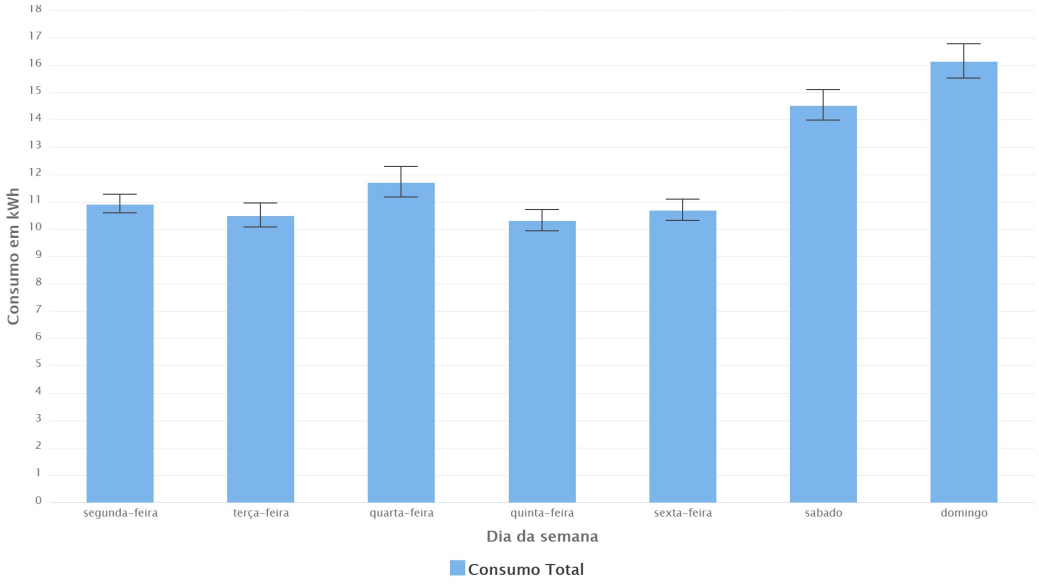


Figura 27: Evolução semanal dos consumos médios

De seguida foi feita uma análise à evolução semanal dos consumos médios do utilizador. A Figura 27 ilustra o gráfico traçado para o efeito.

A partir deste gráfico, chega-se à conclusão que durante o fim de semana (sábado e domingo) os consumos energéticos do utilizador aumentam substancialmente, aumentando em alguns casos mais do que quatro kWh quando comparado com os dias úteis da semana.

Posto isto foi decidido analisar a diferença entre os perfis de consumo nos dias de semana e nos dias de fim de semana. Como seria expectável, no fim de semana, foi observado um aumento considerável da energia consumida, o que pode ser atribuído ao facto de ser mais provável que o utilizador se encontre em casa durante o fim de semana e que consequentemente faça mais uso dos seus equipamento elétricos.

Após a separação entre os dias de semana e os de fim de semana, foram calculadas as médias de consumo destes, respetivamente 10,913 e 15,418 kWh registando-se um aumento de quase 50% do consumo energético durante os fins de semana.

Com base nesta análise foram ainda gerados os gráficos presentes na Figura 28. Esta figura é composta por dois gráficos:

1. O gráfico do consumo total médio, em kWh, ao longo do dia para o fim de semana e para os dias úteis;
2. O gráfico da diferença entre o consumo médio registado ao longo dos dois anos em função das horas do dia (cujos valores foram ilustrados previamente

na Figura 26) e o consumo ao longo do dia para o fim de semana e para os dias úteis.

Ao analisar estes gráficos, é possível observar que, apesar de em certas alturas do dia (nomeadamente durante a noite) os perfis de consumo observados serem semelhantes, durante o dia (principalmente após as doze horas) o consumo energético é bastante superior durante os fins de semana, chegando em certos casos a duplicar face aos dias de semana. É ainda possível observar que o subgrupo de registos relativos aos fins de semana apresentam um erro do desvio padrão superior, facto este que pode ser atribuído a este subgrupo ser composto por um número inferior de registos e ainda a uma (provável) maior variação dos consumos energéticos durante este período da semana.

A comparação do consumo energético entre os dias de semana e os de fim de semana torna-se ainda mais útil para um eventual agente malicioso que queira tentar perceber se o utilizador se encontra ou não em casa. O agente malicioso, com base nos perfis de consumo traçados durante esta comparação, pode facilmente tentar enquadrar os dados emitidos pelo Mirubee do utilizador em tempo real num dos dois perfis traçados e mais facilmente estimar se o utilizador se encontra ou não em casa.

Para aprofundar a análise feita ao consumo energético deste utilizador foi ainda feito o estudo da energia consumida em função da estação do ano. Esta análise tem como objetivo tentar perceber se o utilizador apresenta padrões de uso diferentes ao longo do ano.

A Tabela 4 foi gerada com base no processamento dos dados registados durante os dois anos em que o Mirubee recolheu os dados do utilizador em questão. Nesta Tabela é possível consultar o consumo total ao longo dos dois anos e o consumo médio diário (ambos em kWh) em função da estação do ano.

Tabela 4: Consumo energético do utilizador anos consoante a estação do ano

Estação do Ano	Consumo Total (kWh)	Média do Consumo Diário (kWh)
Inverno	2416,600	13,362
Primavera	2079,731	11,311
Verão	2124.296	11,989
Outono	2200,813	12,166

Figura 28: Comparação do consumo diário durante a semana e o fim de semana

Com base nestes dados chega-se à conclusão que os consumos energéticos do utilizador aumentam principalmente durante o Inverno provavelmente devido à baixa da temperatura assistida durante este período do ano e consequente uso de equipamento de aquecimento elétrico por parte do utilizador.

Foi ainda feito o estudo da evolução do consumo energético ao longo do dia para cada uma das estações do ano, ilustrado na Figura 29 análoga à Figura 28. Como seria de esperar, o Inverno apresenta uma curva com uma evolução mais acentuada mas em geral os consumos entre estações apresentam um perfil bastante semelhante com a exceção do Verão que apresenta um pico de consumo entre as três e quatro horas da manhã (provavelmente devido a um equipamento programado para atuar a estas horas).

Face às recentes alterações dos hábitos quotidianos da maioria da população mundial face à pandemia causada pela Covid-19 foi ainda considerado relevante determinar se seria possível identificar eventuais divergências nos padrões de consumo do utilizador resultantes do confinamento causado por esta epidemia.

Assim, foram criados dois subgrupos de registos:

1. Um subgrupo composto pelos registo gerados entre 01/04/2019 e 31/07/2019 representativo dos consumos anteriores ao confinamento;
2. Um subgrupo composto pelos registo gerados entre 01/04/2020 e 01/07/2020 representativo dos consumos durante ao confinamento;

Com base nesta divisão, foi calculada a média do consumo total diário para cada um dos dois grupo em cima sendo que, para o primeiro grupo, foi registada uma média diária de 10,475 kWh enquanto que o segundo apresenta uma média diária de 13,176 kWh.

Em primeira análise é possível observar uma divergência clara no padrão de consumo do utilizador, posto isto foram mais uma vez gerados o gráficos da evolução diária dos valores do consumo média para os dois grupos gerados representado pela Figura 30.

Apesar de durante a maioria do dia ambos os grupos apresentarem valores semelhantes os registos relativos ao confinamento apresenta dois picos significativos às doze horas e vinte e duas horas.

Com base nos resultados expostos ao longo deste Capítulo chega-se à conclusão que os dados do consumo elétrico de um utilizador refletem os hábitos diários deste e podem ser analisados por um agente externo para extrapolar o comportamento do utilizador na privacidade da sua própria casa. Assim, apesar de inicialmente

Figura 29: Comparação do consumo energético ao longo do dia entre estações do ano

Figura 30: Evolução média dos consumos diários durante o confinamento

os dados do consumo elétrico de um utilizador poderem parecer inócuos, torna-se importante garantir a confidencialidade destes de forma a zelar pela privacidade dos utilizadores de uma plataforma de monitorização de consumos energéticos.

5

SEGURANÇA NO DESENVOLVIMENTO DE APLICAÇÕES WEB

Neste Capítulo será feito um enquadramento às principais vulnerabilidades de seguranças encontradas hoje em dia em aplicações web. Mais, serão apontadas as soluções adotadas durante o desenvolvimento deste projeto para combater estas vulnerabilidades.

5.1 TOP 10 VULNERABILIDADES SEGUNDO A OWASP

A Open Web Application Security Project (OWASP) é, segundo a mesma: «uma comunidade aberta dedicada a permitir que as organizações desenvolvam, adquiram e mantenham aplicações e APIs confiáveis». Criada em 2001 por Mark Curphey, esta comunidade é responsável pela publicação de diversos documentos e recursos disponibilizados gratuitamente através da Internet.

Periodicamente esta organização publica um documento (OWASP, 2017) no qual é feito o levantamento das 10 principais vulnerabilidades de segurança ordenadas segundo a metodologia de classificação de riscos desenvolvida pela OWASP.

Esta metodologia de classificação de riscos, parametriza seis características para cada vulnerabilidade:

1. Agente ameaça;
2. Facilidade de abuso do vetor de ataque;
3. Prevalência da falha de segurança;
4. Facilidade da detetabilidade da fraqueza;
5. Impacto técnico;
6. Impacto no negócio.

Com base no nível atribuído a cada característica, torna-se possível comparar e ordenar as vulnerabilidades analisadas segundo o seu potencial impacto. A tabela

5 ilustra os vários níveis que podem ser atribuídos a cada característica analisada segundo esta metodologia.

Tabela 5: Modelo de classificação de risco da OWASP (adaptado do trabalho realizado pela OWASP (2017))

Agente Ameaça	Abuso	Prevalência da Falha	Detetabilidade	Impacto Técnico	Impacto Negócio
Específico da Aplicação	Fácil: 3 Moderado: 2 Difícil: 1	Predominante: 3 Comum: 2 Incomum: 1	Fácil: 3 Moderado: 2 Difícil: 1	Grave: 3 Moderado: 2 Reduzido: 1	Específico do Negócio

Tabela 6: Classificação de vulnerabilidades web segundo a OWASP

Vulnerabilidade	Abuso	Prevalência da Falha	Detetabilidade da fraqueza	Impacto Técnico
Injeção	Fácil: 3	Comum: 2	Fácil: 3	Grave: 3
Quebra de autenticação	Fácil: 3	Comum: 2	Moderado: 2	Grave: 3
Exposição de dados sensíveis	Moderado: 2	Predominante:3	Moderado: 2	Grave: 3
Entidades Externas de XML (XXE)	Moderado: 2	Comum: 2	Fácil: 3	Grave: 3
Quebra de Controlo de Acessos	Moderado: 2	Comum: 2	Moderado: 2	Grave: 3
Configurações de Segurança Incorretas	Fácil: 3	Predominante:3	Fácil: 3	Moderado: 2
Cross-Site Scripting (XSS)	Fácil: 3	Predominante:3	Fácil: 3	Moderado: 2
Desserialização Insegura	Difícil: 1	Comum: 2	Moderado: 2	Grave: 3
Utilização de componentes vulneráveis	Moderado: 2	Predominante:3	Moderado: 2	Moderado: 2
Registo e Monitorização Insuficiente	Moderado: 2	Predominante:3	Difícil: 1	Moderado: 2

Assim, de seguida serão analisadas em detalhe cada uma das vulnerabilidades de seguranças enumeradas na mais recente edição do top 10 da OWASP publicada em 2017.

5.1.1 *Injeção*

Este tipo de ataque ocorre quando um interpretador recebe um comando que leva à execução maliciosa de comandos a que o utilizador não teria privilégios para executar em condições legítimas. Os ataques de injeção podem ocorrer em qualquer aplicação que permita ao utilizador executar comandos ou mesmo *queries* em bruto sem validação ou processamento prévio destas.

As aplicações com código mais antigo que não foi devidamente mantido são particularmente vulneráveis a ataques de injeção. De facto, muitas destas aplicações com código «legado» não validam ou filtram os dados introduzidos pelo utilizador, enviando diretamente estes dados para o interpretador em questão independentemente do seu contexto ou conteúdo. A maioria destes ataques ocorrem em: «aplicações Structured Query Language (SQL), Lightweight Directory Access Protocol (LDAP), XPath, *queries* NoSQL , comandos Operating System (OS) (através da *shell*), *parsers* Extensible Markup Language (XML), cabeçalhos Simple Mail Transfer Protocol (SMTP), linguagens de expressão ou *queries* Object-Relational Mapping (ORM)».

O impacto causado por ataques de injeção é normalmente substancial uma vez que um utilizador malicioso pode potencialmente ganhar acesso privilegiado ao sistema atacado podendo levar à remoção, alteração ou consulta indevida dos dados da aplicação.

A título de exemplo considere-se a *query* SQL ilustrada na Listagem 3, executada dinamicamente por uma aplicação web na qual o parâmetro «`username`» é derivado pela aplicação e o parâmetro «`itemname`» é introduzido pelo utilizador:

Listagem 3: Exemplo de código SQL vulnerável a ataques de injeção

```
1 SELECT * FROM items WHERE owner = <userName> AND itemname = <itemName>;
```

Em condições normais o utilizador seria apenas capaz de consultar os «`items`» que este possuia, no entanto, uma aplicação que faça a concatenação da *string* «`itemname`» introduzida pelo utilizador com a *string* base da *query* sem fazer um processamento prévio desta (*e. g. escaping* de caracteres) torna-se vulnerável a este tipo de ataque.

Se neste caso o utilizador «`xpto`» introduzisse a *string* na Listagem 33:

Listagem 4: Exemplo de código a injetar

```
name' OR 'a'='a'
```

O resultado final da *query* construída dinamicamente seria:

Listagem 5: *Query* SQL realizada após a injeção

```
SELECT * FROM items WHERE owner = 'xpto' AND itemname = 'name' OR 'a'='a';
```

Com efeito, seria possível ao utilizador «`xpto`» adicionar a condição «`OR 'a'='a'`» à *query* em questão o que implicaria que a condição «`WHERE`» desta *query* seja sempre avaliada como verdadeira. Deste modo o utilizador «`xpto`» seria capaz de consultar todos os «`items`» armazenados na base de dados em questão independentemente do seu dono.

5.1.2 *Quebra de Autenticação*

A má implementação das funcionalidades de autenticação ou gestão de sessão de uma aplicação pode permitir que entidades maliciosas comprometam credenciais de autenticação dos utilizadores desta, podendo assim fazerem-se passar indevidamente por outros utilizadores.

Cada vez mais é possível encontrar na Internet *dumps* de credenciais de utilizadores de websites cuja integridade foi violada. Com base nestes *dumps* é possível criar uma listagem extensiva de credenciais e testá-las exaustivamente noutros websites ou serviços (*credential stuffing*). Para além disto é ainda possível aos atacantes realizar ataques de força bruta (com recurso a ferramentas que automatizam o processo de autenticação), ataques de dicionário e ainda a utilização das credenciais administrativas utilizadas por omissão nos mais variados sistemas. Assim, devido à quantidade substancial de recursos existentes online, a quebra de autenticação de uma aplicação web torna-se numa vulnerabilidade mais fácil de explorar do que seria desejável.

Uma vez autenticado indevidamente numa conta que não lhe pertence, uma entidade maliciosa passa a ter acesso a dados potencialmente confidenciais de outros utilizadores podendo inclusive executar comandos ou tarefas indevidamente.

Facto este que se agrava no caso de a conta comprometida se tratar de uma conta administrativa.

5.1.3 *Exposição de Dados Sensíveis*

A proteção não adequada dos dados dos utilizadores de uma aplicação web pode causar com que atacantes desta consigam aceder ou capturar dados de natureza sensível destes (*e. g.* dados do histórico de saúde, dados financeiros ou dados de identificação pessoal).

Este tipo de ataque pode acontecer tanto em dados estacionários (armazenados numa base de dados) como em dados em trânsito (entre os servidores da aplicação e o *browser* do utilizador). Os dados de natureza sensível nunca devem ser armazenados numa base de dados em *cleartext* devendo estes ser previamente cifrados e só então armazenados. Desta forma mesmo que a confidencialidade do conteúdo da base de dados seja comprometido torna-se mais difícil aos atacante extrair este tipo de dados, podendo mesmo ser praticamente impossível, dependendo do algoritmo criptográfico empregue para cifrar os dados.

No que toca aos dados em trânsito, para além da cifragem destes, devem ainda ser empregues protocolos de transporte de dados seguros (*e. g.* Transport Layer Security (TLS) ou SSL) tal como acontece nos protocolos HTTPS ou File Transfer Protocol Secure (FTPS) de forma a evitar ou mitigar ataques à confidencialidade dos dados transmitidos.

A exposição deste tipo de dados pode comprometer a vida pessoal dos utilizadores da aplicação. Mais, este tipo de dados, podem encontrar-se protegidos por legislação que visa proteger os mesmos, tal como acontece com o Regulamento Geral de Proteção de Dados (RGPD) na União Europeia (UE). Deste modo, a proteção não adequada deste tipo de dados pode ainda resultar em multas de valor substancial à entidade responsável pela aplicação web.

5.1.4 *Entidades Externas de XML*

O uso de processadores XML antigos ou mal configurados pode levar à exposição ou alteração de recursos internos da aplicação web através do uso da tag XML `<ENTITY>`.

Qualquer aplicação que permita o upload direto de ficheiros XML, a inserção de dados em ficheiros XML no servidor, ou que utilize uma versão do protocolo Simple Object Access Protocol (SOAP) inferior à 1.2 pode estar potencialmente vulnerável a este tipo de ataque uma vez que a *framework* deste procolo utiliza entidades XML. Esta falha de segurança pode levar à extração de dados internos do servidor, à execução de comandos remotos no servidor ou mesmo a ataques de DoS tais como o ataque «*Billion Laughs*». Este ataque consiste na declaração recursiva de entidades XML com base na multiplicação da entidade anterior o que leva a que a ultima entidade definida assuma mil milhões de vezes o valor da entidade original. Ao fazer o *parsing* da última entidade declarada, o computador em questão esgota os seus recursos o que resulta na negação do serviço prestado pela aplicação.

Listagem 6: Exemplo de código XML vulnerável

```
1 <!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxeSYSTEM
2 "file:///etc/passwd»]>
3 <foo>&xxe;</foo>
```

Ao executar o código em cima na listagem 6, o processador XML iria retornar o conteúdo do ficheiro «`etc/passwd`» do servidor o que poderia comprometer a segurança deste.

5.1.5 *Quebra de Controlo de Acessos*

Uma má implementação ou execução das restrições de acesso dos utilizadores às funcionalidades e dados de uma aplicação web pode levar ao acesso ilegítimo destes por parte de um utilizador mal intencionado.

Quanto mais complexa uma aplicação web se torna, mais difícil se torna garantir que os controlos de acesso aos recursos desta se encontram devidamente configurados, tornando-se necessário a realização extensiva de testes funcionais a estes controlos.

Ao detetar eventuais lacunas no controlo de acesso de uma aplicação, quer manualmente, quer utilizando ferramentas que automatizam o processo de descoberta, um atacante pode conseguir aceder a recursos aos quais não teria acesso em condições normais. Uma vez contornados estes controlos, os atacantes passam a poder fazer-se passar por outros utilizadores legítimos ou mesmo utilizadores com acessos privilegiados, podendo assim assumir o controlo total sobre os dados e funcionalidades da aplicação.

5.1.6 *Configurações de Segurança Incorretas*

A incorreta configuração de segurança de uma aplicação é atualmente o tipo de vulnerabilidade mais prevalente em aplicações web.

Um dos factores que mais contribui para a prevalência desta vulnerabilidade prende-se com o facto desta poder assumir um sentido bastante lato uma vez que praticamente qualquer componente de uma aplicação pode ser mal configurado. Quer seja uma biblioteca importada, a base de dados utilizada, cabeçalhos HTTP, *frameworks*, ou mesmo a configuração dos serviços utilizados pelo servidor, praticamente qualquer componente de uma aplicação, independentemente do nível da camada aplicacional em que este atue, é passível de ser indevidamente configurado e comprometer a segurança da aplicação em questão.

Uma vez detetada este tipo de vulnerabilidade, o atacante pode tentar explorá-la e eventualmente ganhar acesso ilegítimo a recursos protegidos do servidor, comprometendo os dados da aplicação.

Um exemplo bastante comum da exploração desta vulnerabilidade passa pela análise de mensagens de erro demasiado verbosas implementadas pelos programadores da aplicação (normalmente por motivos de *debugging*). Com efeito, a inclusão de informação sensível nas mensagens de erro de uma aplicação (*e. g.* a *stack trace* da aplicação, a versão dos vários componentes utilizados por esta ou mesmo variáveis de sistema) pode resultar na exposição de outras vulnerabilidades susceptíveis de serem exploradas pelo atacante.

5.1.7 *Cross-Site Scripting*

O Cross-Site Scripting (XSS) é um tipo específico de injeção na qual é injetado código malicioso numa página web legítima. Geralmente este tipo de ataques ocorre quando um atacante leva um utilizador a executar código malicioso que será executado pelo *browser* da vítima (uma vez que é impossível para o *browser* verificar a autenticidade dos *scripts* executados). Ao executar o código malicioso, o *browser* pode ser configurado para transmitir para o atacante os dados da sessão do utilizador (*e. g. cookies* ou *tokens*) ou mesmo alterar o conteúdo da página web da vítima. Estima-se que cerca de dois terços de todas aplicações web sejam vulneráveis a este tipo de ataques.

Existem três tipos de ataques de Cross-Site Scripting:

- **XSS Refletido**: os ataques de XSS refletidos podem ocorrer em aplicações que devolvam o *input* enviado pelos seus utilizadores sem validação prévia deste. Estes tipo de ataques podem ser dissecados em quatro fases, ilustradas na figura 31:

  1. O atacante envia à vítima uma hiperligação injetada com código malicioso para um *website* legítimo;
  2. Ao clicar na hiperligação o *browser* da vítima envia para o servidor do *website* o *script* malicioso;
  3. O servidor recebe o *input* malicioso enviado pela vítima, processa-o e envia para o *browser* da vítima o conteúdo requisitado em conjunto com o *input* malicioso que é devolvido (ou refletido) de volta para a vítima;
  4. Ao receber o código malicioso, refletido pelo *website* legítimo, o *browser* executa o código, resultando tipicamente no envio de dados sensíveis ao atacante.

- **XSS Armazenado ou Persistente**: este tipo de ataque é normalmente considerado mais perigoso uma vez que o código malicioso é persistido no *website* e potencialmente transmitido para outros utilizadores.

  Este tipo de ataque ocorre quando o *input* enviado por um utilizador é armazenado pelo servidor sem validação ou processamento prévio (*escaping*) (*e. g.* valores armazenados na base de dados, mensagens de utilizadores em fóruns, *etc*). Uma vez armazenado, o *script* malicioso passa a poder ser executado pelos *browsers* de qualquer utilizador que receba do *website* o conteúdo infetado;

- **Document Object Model (DOM) XSS**: o DOM XSS ocorre principalmente em aplicações *single page* e/ou *frameworks* de javascript que permitam a edição do seu DOM por parte do atacante.

A título de exemplo de um ataque de XSS refletido, considere-se o excerto de código HTML duma página web presente na Listagem 7:

Listagem 7: Exemplo de conteúdo de uma página HTML vulnerável

```
1 (String) page += "<input name='creditcard' type='TEXT'  value='" +
↪  request.getParameter("CC") + "'>";
```

Considere-se ainda que o utilizador é redirecionado para esta página após enviar para o servidor um pedido HTTP contendo o campo «`CC`» e que o servidor devolveu

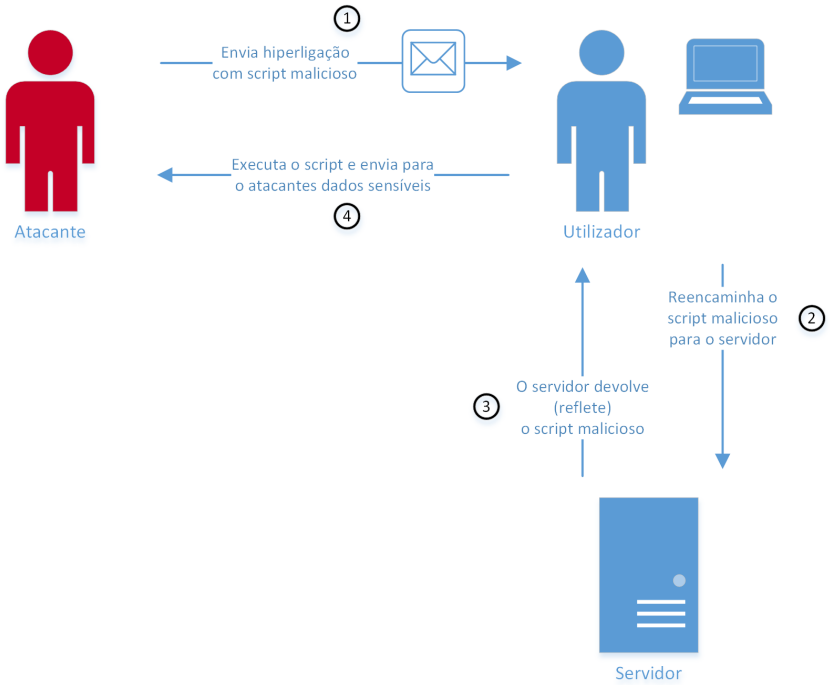


Figura 31: Diagrama de uma ataque de XSS refletido

de volta este pedido na sua íntegra para o utilizador, sem validação ou processamento prévio (tornando-se vulnerável a ataques de XSS refletido).

Em situações normais o código em cima acrescentaria um campo de texto à página web que assumiria o valor do parâmetro «`CC`» do pedido HTTP enviado originalmente pelo utilizador. No entanto um atacante pode aproveitar-se desta vulnerabilidade e enviar ao utilizador uma hiperligação que, quando acedida envie para o servidor da aplicação vulnerável um pedido em que campo «`CC`» assume o seguinte valor:

Listagem 8: Valor do campo CC no pedido HTTP partilhado pelo atacante

```
1 ><script>
2     document.location="http://www.attacker.com/cgi-bin/cookie.cgi?foo="+document
      ↪  .cookie
3 </script>
```

Ao receber de volta este pedido, o web *browser* da vítima é redirecionado para o domínio «`http://www.attacker.com`» pertencente ao atacante. Mais, ao ser redirecionado, o web *browser* efetua um pedido GET com o campo «`foo`» a assumir o valor do conteúdo da *cookie* do *website* vulnerável.

Ao enviar o conteúdo da *cookie* para o atacante, a vítima arrisca-se a expor o ID da sua sessão, efetivamente dando ao atacante o controlo da sua sessão atual.

5.1.8 *Desserialização Insegura*

A serialização e desserialização de objetos é uma funcionalidade fundamental para qualquer aplicação web nos tempos que correm.

Apesar da popularidade de formatos universais de dados serializados como o XML ou JavaScript Object Notation (JSON), diferentes linguagens de programação ou APIs oferecem muitas vezes formatos de serialização e desserialização nativos, que normalmente permitem personalizar este processo. No entanto, apesar das vantagens proporcionadas por estes formatos nativos, muitas vezes estes podem ser abusados por atacantes como vetor de ataque ao misturar os dados serializados com a lógica de programação da aplicação.

Mesmo sendo uma vulnerabilidade mais complexa de explorar por parte do atacante, devido à especificidade de cada aplicação, a desserialização não segura de dados é ainda uma vulnerabilidade bastante prevalente que pode levar à injeção de código e à escalada de privilégios do atacante, podendo resultar na exposição de dados sensíveis e na negação do serviço da aplicação.

5.1.9 *Utilização de Componentes Vulneráveis*

Quando os programadores da aplicação utilizam ou importam componentes «externos», com vulnerabilidade bem conhecidas, que serão executados com o mesmo nível de privilégio que a aplicação web, expõem os restantes componentes da aplicação a brechas de segurança provenientes destes componentes vulneráveis.

Assim, por muito segura que a aplicação seja, a vulnerabilidade de um único componente pode ser explorada para comprometer todo o sistema (atuando como o elo mais fraco da aplicação do ponto de vista da segurança).

Uma vez que as vulnerabilidades de certos componentes são bem conhecidas, é muitas vezes possível encontrar ferramentas ou recursos online que facilitam a exploração destas vulnerabilidades. Mesmo estando ciente das potenciais vulnerabilidades presentes num componente utilizado, a equipa de desenvolvimento da aplicação pode optar por menosprezar o possível impacto de segurança em prole

das funcionalidades prestadas pelo componente e pela facilidade de integração deste, quando comparado com a implementação das suas funcionalidades de raíz.

O impacto causado por esta vulnerabilidade pode variar consoante a integração do componente em questão na aplicação. Apesar de na maioria dos casos o abuso desta vulnerabilidade ter um impacto reduzido, certos casos podem comprometer a segurança de toda a aplicação.

5.1.10 *Registo e Monitorização Insuficiente*

A criação de *logs* durante a monitorização e registo de eventos num sistema informático é um pilar indispensável para diagnosticar e combater ataques a uma aplicação web.

A aplicação deve ser configurada de forma a criar registos com informação pertinente aquando da deteção de vestígios de uma possível intrusão que podem e devem ser definidos antes e após os testes de penetração efetuados à aplicação. Idealmente, devem ser criados registos sempre que um atacante ou uma entidade externa tente descobrir vulnerabilidades no sistema.

A falha ao fazer um registo e monitorização completo e detalhado da atividade da aplicação resulta na não deteção atempada de um ataque, proporcionando ao atacante mais tempo e mais oportunidades para desenvolver e propagar o seu ataque.

5.2 SOLUÇÕES

Feito o enquadramento teórico de cada uma das dez vulnerabilidades abordadas na lista da OWASP, de seguida serão abordadas as medidas tomadas para combater cada destas vulnerabilidades.

5.2.1 *Injeção*

Uma vez que a aplicação desenvolvida no âmbito deste projeto utiliza uma base de dados MySQL, tornou-se necessário adotar medidas preventivas para evitar ataques de injeção. Mais concretamente, foi utilizada a ferramenta ORM Eloquent da *framework* Laravel para realizar todas as consultas, inserções, alterações ou remoções na base de dados que utilizem dados de entrada provenientes de utilizadores.

Através do Eloquent torna-se possível realizar operações SQL a partir da linguagem de programação PHP: Hypertext Preprocessor (PHP), evitando-se assim ter que construir *statements* dinâmicos expondo a aplicação a ataques de injeção.

Caso se tornasse necessária a criação de um comando SQL gerado dinamicamente com base em valores de entrada introduzidos pelo utilizador, estes valores de entrada seriam devidamente validados e seria realizado o *escape* destes.

5.2.2 *Quebra de Autenticação*

Para combater ataques de força bruta ou de dicionário, foi implementada a autenticação por múltiplos fatores no website. Deste modo, mesmo que as credenciais do utilizador sejam comprometidas ou descobertas, através de ferramentas que automatizam ataques de força bruta como o Hydra, o utilizador necessita de comprovar a sua autenticidade mais uma vez para ter acesso à sua conta.

A autenticação multi fator normalmente é feita ao combinar dois ou mais tipos de credenciais distintas. Existem três tipos distintos de credenciais que podem ser utilizadas:

- **Algo que o utilizador sabe**: normalmente uma *password* ou uma pergunta de segurança;
- **Algo que o utilizador possui**: por exemplo, uma chave de segurança;
- **Algo que o utilizador é**: tal como a verificação dos dados biométricos do utilizador.

No caso da aplicação desenvolvida, uma vez que o uso da *password* se enquadra o tipo de credencial que o utilizador sabe, torna-se necessário que as restantes credencias utilizadas no seguintes passos de autenticação sejam ou algo que o utilizador possui ou algo que o utilizador é.

Posto isto, como segundo passo de autenticação no *website* o utilizador pode autenticar-se através das seguintes maneiras: da aplicação Google Authenticator, duma chave de segurança Universal 2nd Factor (U2F), do protocolo Secure, Quick, Reliable Login (SQRL) ou através de um *token* enviado por email. Esta funcionalidade será abordada em maior detalhe no Capítulo 6: «Solução Proposta» deste documento na secção «Funcionalidades de Segurança».

5.2.3 *Exposição de Dados Sensíveis*

Apesar de, à partida o *website* proposto não armazenar dados de natureza sensível (*e. g.* dados de saúde e dados financeiros), cada um dos vários tipos de dados processados foi classificado consoante a sua sensibilidade.

Através da classificação da sensibilidades dos dados processados chegou-se à conclusão que a localização dos dispositivos dos utilizadores pode conter informação sensível e como tal foram tomadas medidas consideradas necessárias para manter a confidencialidade deste campo. Para além deste campo, as leituras dos consumos energéticos foram também consideradas sensíveis uma vez que a partir destas pode um agente malicioso pode deduzir quando é que um utilizador se encontra em casa ou no local em que se encontra instalado o dispositivo. Assim, tanto o campo da localização do dispositivos do utilizado como o conteúdo das leituras energéticas passaram a ser cifrados antes de serem armazenados na base de dados.

5.2.4 *Entidades Externas de XML*

Como forma de evitar este tipo de vulnerabilidades, foi optado pelo uso do formato de dados JSON, devido à sua simplicidade e segurança quando comparado com o formato XML.

No entanto, caso o website, por qualquer razão, utilizasse o formato XML ou o protocolo SOAP seriam adotadas as medidas necessárias para combater a exploração desta vulnerabilidade. Mais concretamente seria desativado o processamento de entidades externas de XML e seria empregue a especificação mais recente do protocolo SOAP (versão 1.2).

5.2.5 *Quebra de Controlo de Acessos*

Para combater a quebra de controlo de acessos à aplicação, foram implementadas as boas práticas de segurança que em conjunto com as ferramentas de segurança da *framework* Laravel proporcionam um grau de segurança adequado à aplicação.

Mais concretamente foram adotadas as seguintes medidas:

- Negação de acesso a recursos por omissão;

- Verificação constante das permissões do utilizador durante o acesso aos recursos da aplicação;
- Negação do acesso à listagem de diretorias do servidor;
- Utilização de mecanismos de segurança do Laravel tais como o *middleware* para controlar e validar o acesso aos vários recursos do servidor.

5.2.6 *Configurações de Segurança Incorrectas*

Para garantir a correta configuração de segurança dos vários componentes do servidor da aplicação, o processo de instalação e configuração foi automatizado de forma a evitar eventuais erros de configuração provenientes da configuração manual dos vários componentes empregues.

Mais, foram utilizados, sempre que possível, *scripts* e boas práticas de *hardening* dos vários componentes individuais da aplicação. Por exemplo no caso da base de dados MySQL foi executado o *script*:

mysql_secure_installation

Que automatiza o processo de configuração de segurança da base de dados.

Foram também aplicadas as boas práticas e convenções de segurança da *firewall* iptables e do fail2ban, para garantir a segurança dos vários componentes da aplicação expostos à Internet

Também importante para o efeito, foi o facto de não terem sido incluídos ou ativados componentes não fossem estritamente necessários ao correto funcionamento da plataforma.

5.2.7 *Cross-Site Scripting*

Sempre que considerado necessário é realizada a validação e *escaping* dos valores enviados pelo utilizador por parte do servidor do website.

Para além disto, através da *framework* de javascript Vue, qualquer valor introduzido pelo utilizador que seja posteriormente utilizado no DOM da página web ou *scripts*, é automaticamente *escaped* evitando assim a injeção de código malicioso através de Cross-Site Scripting.

5.2.8 *Desserialização Insegura*

A desserialização não segura de dados é combatida ao usar apenas formatos de dados primitivos, mais concretamente JSON.

No entanto, na eventualidade de se tornar necessário desserializar dados a partir de formatos não primitivos, seriam empregues verificações da autenticidade e integridades dos dados a desserializar, por exemplo a partir de uma assinatura digital. Este processo seria ainda devidamente monitorizado e registado de forma a detetar possíveis anomalias no seu normal comportamento.

5.2.9 *Utilização de Componentes Vulneráveis*

Com o intuito de combater a exposição do *website* a partir de componentes vulneráveis é realizada periodicamente a verificação de vulnerabilidades nos vários componentes, tanto do *front-end* como do *back-end*, deste.

Para tal são utilizados repositórios de vulnerabilidades conhecidas tais como o Common Vulnerabilities and Exposures (CVE) e o National Vulnerability Database (NVD) ou mesmo o próprio GitHub que automaticamente deteta e verifica a existência de novas vulnerabilidades nos vários componentes utilizados.

Foi ainda feita uma subscrição por email aos alertas de segurança dos vários componentes utilizados nas fontes de vulnerabilidades listada em cima. Para além disto, é feita uma monitorização constante a possíveis actualizações aos componentes utilizados e sempre que possível é realizada a actualização destes.

5.2.10 *Registo e Monitorização Insuficiente*

Apesar da *framework* Laravel por omissão fazer uma monitorização e registo adequado dos principais eventos de segurança no website, foram ainda configurados mais eventos a partir dos quais são gerados *logs* com o objetivo de parametrizar e detetar eventuais abusos de acesso por parte dos utilizadores nomeadamente durante falhas no controlo de acessos a recursos do *website* e aquando do envio de dados de entrada mal formatados por parte do utilizador.

Foi ainda tomado o cuidado de tentar assegurar a uniformidade dos vários *logs* apesar da heterogeneidade dos vários eventos geradores destes.

6

SOLUÇÃO PROPOSTA

Uma vez abordadas as principais vulnerabilidades que afetam as aplicações web hoje em dia, será apresentada e escrutinada a solução proposta.

Inicialmente será feito um enquadramento ao modo de funcionamento original do Mirubee, será analisado o processo de emparelhamento deste dispositivo com o *access point* Wi-Fi do utilizador e o modo como este envia os seus dados para a plataforma do fabricante (Smilics ou Circutor).

O enquadramento do processo de instalação original servirá para fundamentar a arquitetura da solução desenvolvida que será abordada posteriormente neste Capítulo. Será feita uma análise aos vários componentes individuais que integram a solução bem como ao modo como estes interagem entre si. De seguida serão documentadas as principais funcionalidades da plataforma web que foi concebida no âmbito deste projeto.

6.1 ENQUADRAMENTO

O processo de *set-up* do Mirubee realiza-se através da aplicação móvel do fabricante. O utilizador deverá começar por fazer o *reset* ao dispostivo o que levará a que este crie um *access point* ao qual o utilizador se deve ligar para começar o processo de configuração inicial (ver Figura 32).

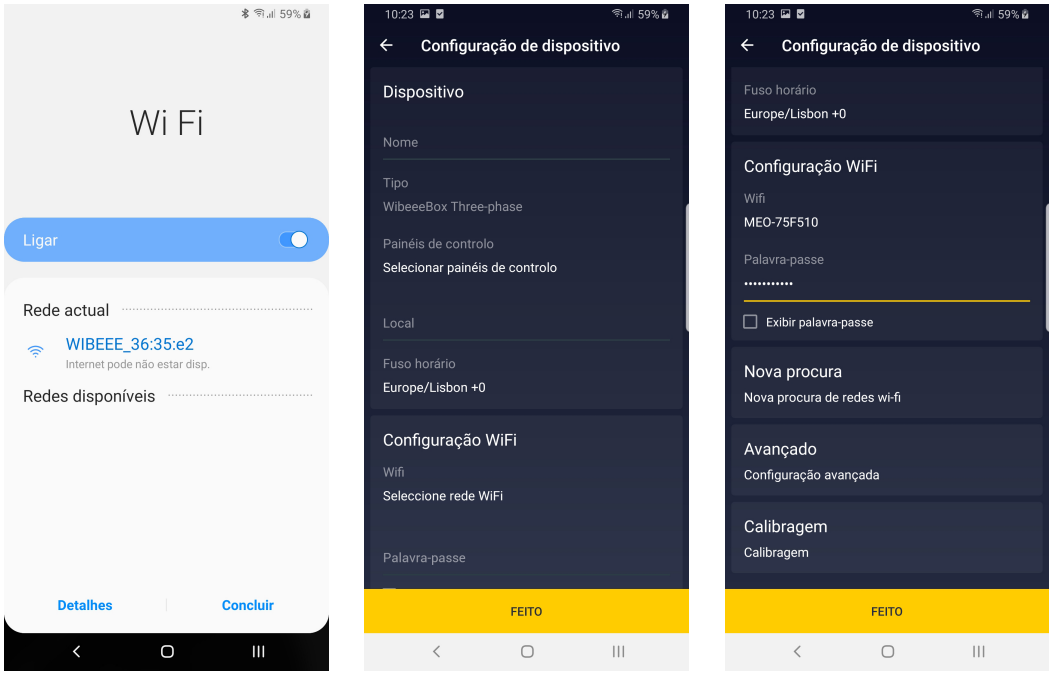


Figura 32: Processo de configuração do Mirubee (da esquerda para a direita: ligação ao *access point* gerado pelo Mirubee, ecrã de configuração inicial e configuração do access point Wi-Fi ao qual o Mirubee se irá conectar para enviar as suas leituras pela Internet)

Uma vez conectado ao *access point* do Mirubee, o utilizador poderá, através da aplicação móvel, configurar o Mirubee consoante as suas necessidades. Mais concretamente, o utilizador deverá definir o nome do dispositivo, a localização deste e as credenciais de autenticação que o Mirubee deve utilizar para se conectar ao *access point* Wi-Fi do utilizador de forma a que este possa enviar as suas leituras para a plataforma do fabricante.

Concluído o processo de emparelhamento ao *access point* Wi-Fi do utilizador, o Mirubee irá periodicamente enviar o valor das suas leituras para o endereço e porto especificados durante o processo de configuração deste (por omissão: `wibeee.smilics.com:8000`). Este dispositivo envia as suas leituras através do protocolo HTTP, um protocolo que, conforme referido anteriormente, não garante a confidencialidade dos dados transmitidos por este.

Assim, uma vez que é impossível alterar o comportamento do Mirubee sem ter acesso ao *firmware* deste, de forma a garantir a confidencialidade dos dados enviados para a plataforma desenvolvida neste projeto, foi decidido (durante a especificação desta) criar uma Virtual Local Area Network (VLAN) (para cada dispositivo do utilizador) para isolar o envio dos pedidos GET efetuados pelo Mirubee. Deste

modo é possível evitar tanto quanto possível que entidades externas interceptem os pacotes não cifrados enviados pelo Mirubee.

6.2 ARQUITETURA DA SOLUÇÃO PROPOSTA

A solução proposta, cuja a arquitetura se encontra ilustrada resumidamente na Figura 33, é composta por três elementos distintos:

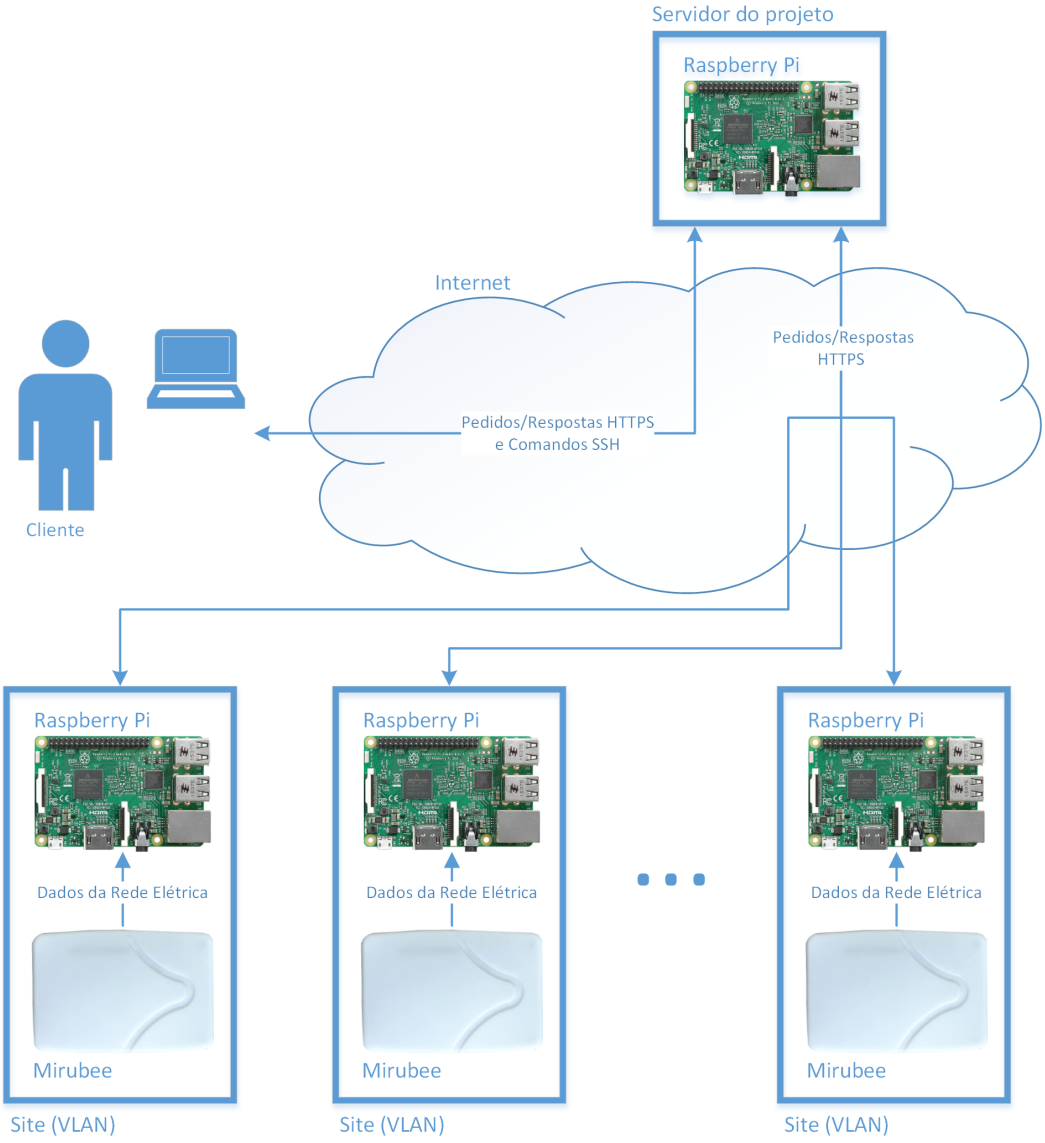


Figura 33: Arquitetura geral da solução proposta

- **Os clientes**: através de um web *browser*, o utilizador pode aceder ao *website* hospedado pelo servidor e consultar os dados armazenados por este. O utiliza-

dor pode ainda aceder remotamente ao servidor por Secure Shell (SSH) caso seja necessário realizar qualquer tipo de manutenção;

- **Os *sites* dos utilizadores**: cada utilizador deverá utilizar o *website* para configurar pelo menos um local, ou «*site*», cuja rede elétrica este queira monitorizar (*e. g.* a sua casa ou local de trabalho).

  Em cada um dos *sites* do utilizador, deverá ser instalado um Mirubee e um Raspberry Pi, numa VLAN privada de forma a isolar a transmissão de dados não segura do Mirubee. O Mirubee será então responsável por monitorizar os dados da rede elétrica deste local que serão de seguida encaminhados para o Raspberry Pi presente na sua rede local. Uma vez validados e processados, o Pi irá atuar como *proxy* e transmitir os dados recebidos de forma segura para o servidor desta aplicação;

- **O servidor da aplicação web**: composto por um Raspberry Pi, é responsável por hospedar o *website* através dos quais os utilizadores irão consultar os dados da rede elétrica de cada um dos *sites* dos utilizadores.

Convém ainda referir que, a título de conveniência, tanto o servidor da plataforma como o *proxy* do *site* do utilizador podem ser instanciados no mesmo Raspberry Pi, consoante as necessidades do utilizador.

Uma vez explicada a arquitetura geral da solução proposta, bem como a maneira como os vários componentes interagem entre si, será de seguida explicado em maior detalhe a arquitetura do servidor da aplicação e dos *sites* dos utilizadores.

6.2.1 *Arquitetura do Servidor*

A arquitetura do servidor desta aplicação encontra-se ilustrada em maior detalhe na Figura 34.

Visto que este servidor estará exposto à Internet, tanto o acesso à plataforma web como o acesso por SSH estarão protegidos pela *firewall* iptables (responsável por filtar tráfego da rede) e pela *framework* fail2ban (responsável por evitar ataques de *credential stuffing*).

O *website* desta aplicação foi implementado através da *framework* Laravel, devido à robustez desta, ao ênfase colocada à segurança, e às ferramentas que disponibiliza para proteger o acesso aos vários recursos do *website*.

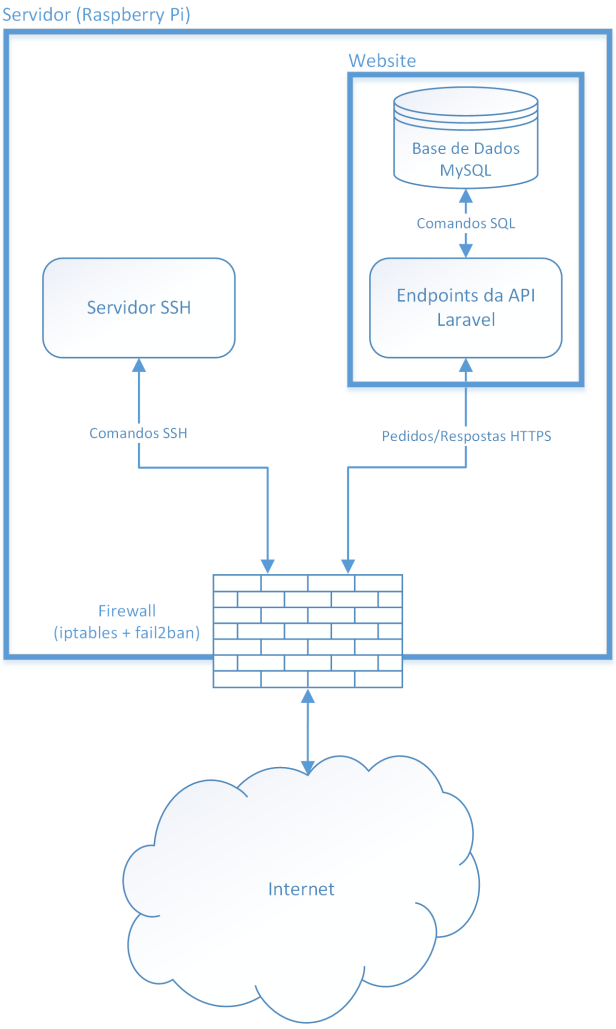


Figura 34: Arquitetura do servidor da aplicação

Quando o utilizador acede à plataforma web para visualizar os dados registados pelo Mirubee esta faz as respetivas *queries* à base de dados para desenhar os gráficos para o utilizador consultar.

Conforme referido anteriormente alguns dos dados armazenados na base de dados foram considerados de natureza sensível e como tal são cifrados antes de serem persistidos na base de dados, tal como acontece com as leituras energéticas do utilizador.

O servidor permite ainda que o utilizador se conecte a este remotamente através do protocolo SSH que se encontra a correr no porto 22 deste. Ao ligar-se remotamente à *shell* do servidor o utilizador pode, de maneira segura e cómoda, ajustar o modo de funcionamento deste e ainda realizar a manutenção dos vários componentes da plataforma web hospedada neste conforme necessário. É aconselhado, sempre que possível que a autenticação SSH seja feita através de chaves assimétricas, mais

concretamente através de uma função de curva elíptica Diffie–Hellman, tal como acontece por exemplo com a curva EC25519 (Bernstein, 2006).

6.2.2 *Arquitetura dos Sites do Utilizador*

A Figura 35 ilustra, em detalhe, a arquitetura de um *site* bem como a maneira como os vários componentes deste interagem entre si.

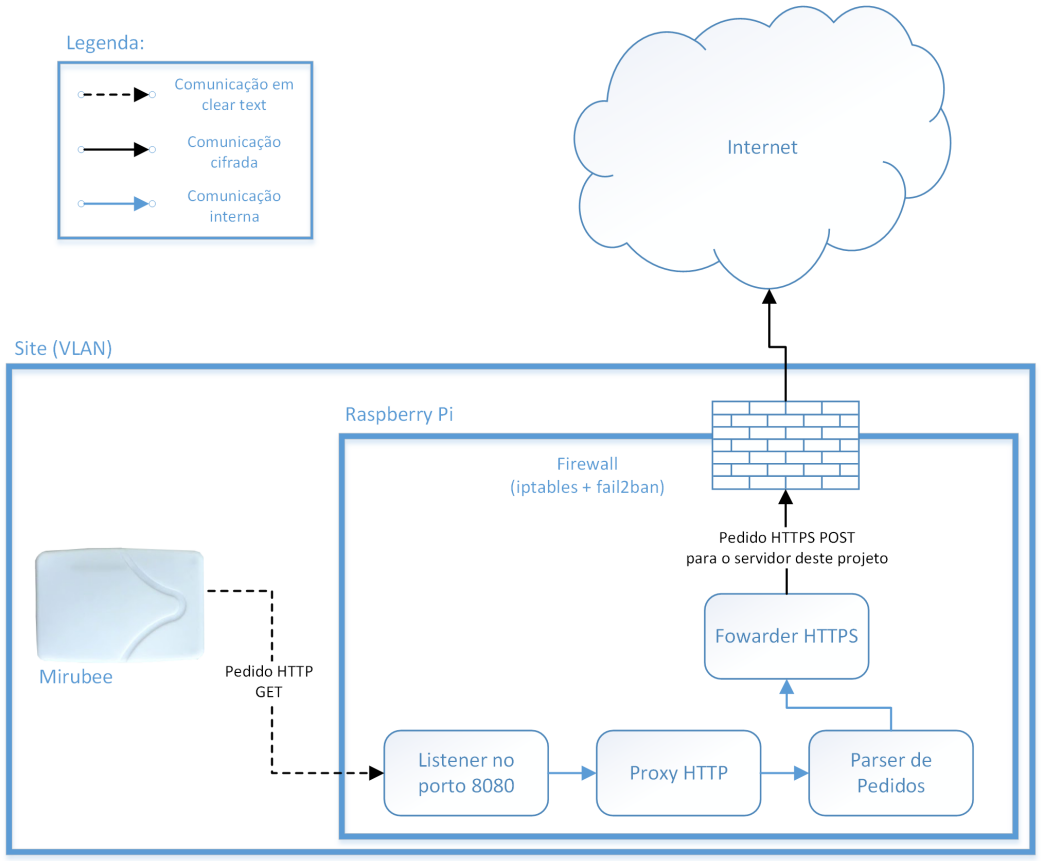


Figura 35: Arquitetura de um *site*

Uma vez que o Mirubee envia os seus dados de maneira não segura, tornou-se necessário arranjar uma solução que garantisse a confidencialidade e autenticidade dos dados enviados para o servidor desta aplicação. Para tal foi decidido utilizar uma VLAN composta pelo Mirubee e um Raspberry Pi. Assim, o Mirubee, através da aplicação móvel da Smilics, é configurado para passar a enviar os seus dados para o Raspberry Pi presente na sua VLAN que estará a correr um serviço que escuta, no porto 8080, pelos pedidos HTTP realizados pelo Mirubee (ver Figura 36).

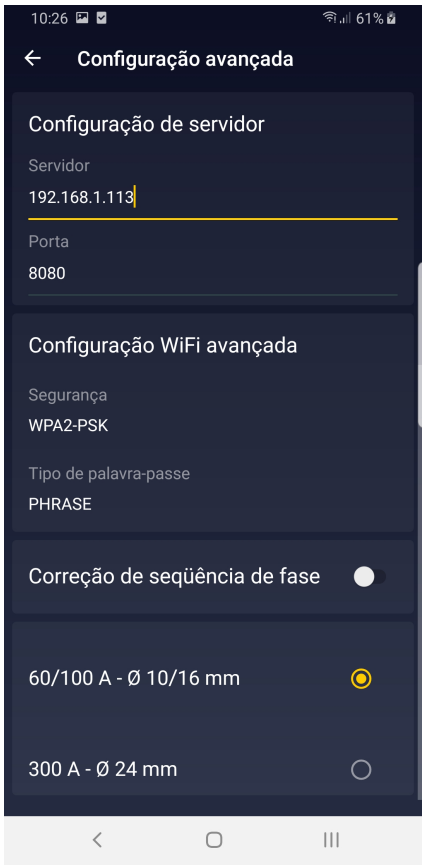


Figura 36: Configuração do endereço e porto do servidor do Mirubee

Ao ser instanciado, o Raspberry Pi autentica-se no servidor desta aplicação e recebe um *token* que será utilizado durante o resto do ciclo de vida deste serviço para autenticar os pedidos realizados pelo Raspberry Pi perante o servidor.

O componente «Proxy HTTP», ao receber dados do Mirubee, processa o pedido recebido e constrói um pedido POST que será enviado por HTTPS para o servidor desta aplicação, devidamente autenticado através do *token* recebido durante a instanciação deste serviço.

Desta forma, torna-se possível isolar o processo de envio de dados do Mirubee e mitigar, tanto quanto possível, as vulnerabilidades causadas pelo uso de protocolos e práticas não seguras deste *smart meter*.

6.3 FUNCIONALIDADES DA PLATAFORMA WEB DESENVOLVIDA

De seguida serão enumeradas as principais funcionalidades implementadas na plataforma web com base no levantamento e especificação de requisitos da plataforma.

As funcionalidades implementadas podem ser divididas em dois grupos distintos, as funcionalidades de segurança e as funcionalidades de monitorização dos consumos energéticos dos utilizadores.

6.3.1 *Funcionalidades de Segurança*

Com base tanto na análise de segurança feita no Capítulo 3 à aplicação móvel e à plataforma web da Smilics e da Circutor como no enquadramento realizado no Capítulo 5 às principais vulnerabilidades de segurança em aplicações web segundo a OWASP, foram implementados funcionalidades e mecanismos de segurança que colmatam as principais vulnerabilidades de segurança da solução de desenvolvida originalmente pela Smilics.

- **Implementação do protocolo HTTPS**: através do serviço «let's encrypt»[1] foi gerado um certificado SSL para o domínio «`securesmartmonitor.ga`», Figura 37, que é utilizado pelo protocolo HTTPS para cifrar os dados transmitido pela plataforma e para autenticá-la perante os seus utilizadores.

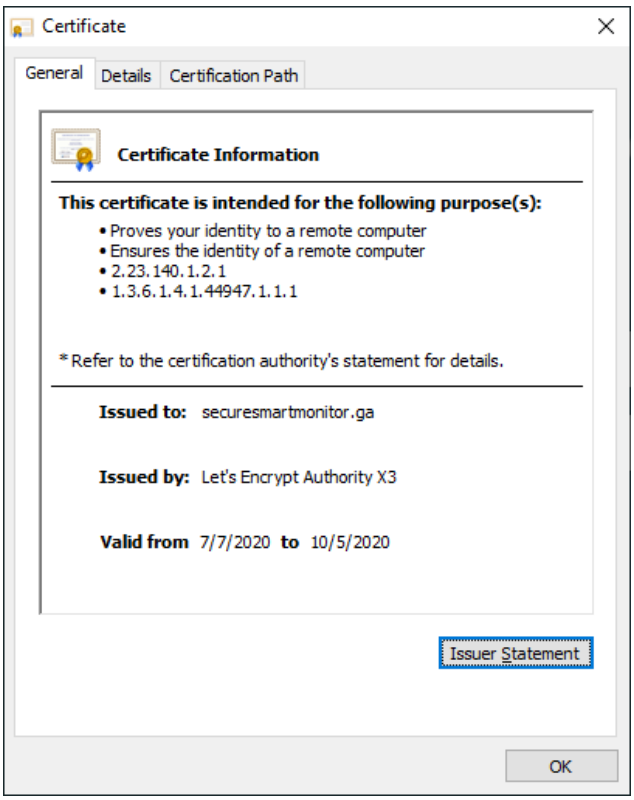
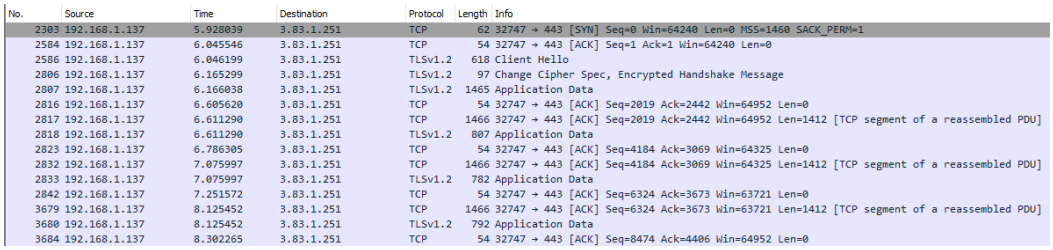


Figura 37: Certificado SSL gerado

1 https://letsencrypt.org/

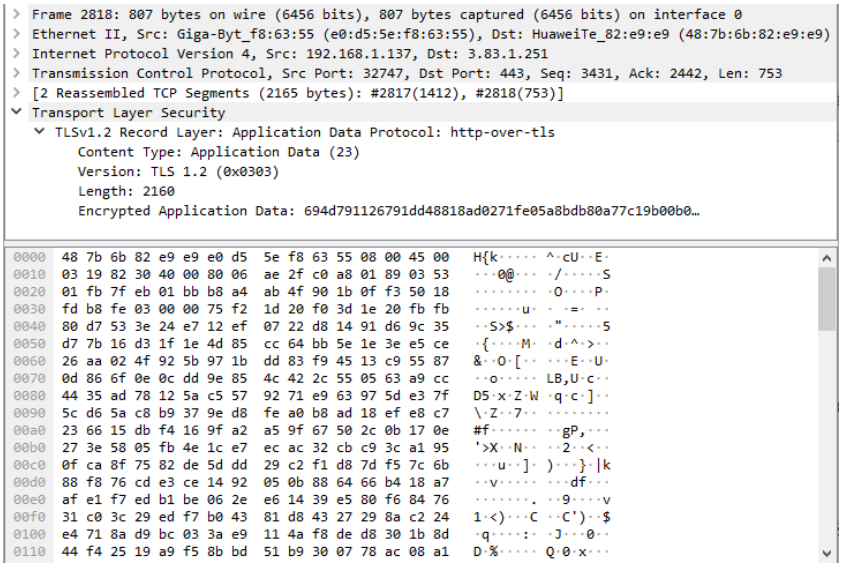
O serviço «let's encrypt» foi utilizado neste projeto devido ao facto deste permitir criar certificados SSL de forma gratuita e automatizada o que facilita a implementação do protocolo HTTPS em praticamente qualquer *website* ou plataforma web, independentemente do seu âmbito ou orçamento.

Deste modo, os pacotes enviados tanto pelo *website* como pelos seus utilizadores, mesmo que interceptados por terceiros, encontrar-se-ão cifrados tornando-se assim praticamente impossível de aceder ao conteúdo destes. As Figuras 38 e 39 obtidas através da captura de pacotes, usando o aplicativo Wireshark, ilustram o processo de transmissão de dados cifrados entre o *website* e o browser do utilizador.



No.	Source	Time	Destination	Protocol	Length	Info
2303	192.168.1.137	5.928039	3.83.1.251	TCP	62	32747 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
2584	192.168.1.137	6.045546	3.83.1.251	TCP	54	32747 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
2586	192.168.1.137	6.046199	3.83.1.251	TLSv1.2	618	Client Hello
2806	192.168.1.137	6.165299	3.83.1.251	TLSv1.2	97	Change Cipher Spec, Encrypted Handshake Message
2807	192.168.1.137	6.166038	3.83.1.251	TLSv1.2	1465	Application Data
2816	192.168.1.137	6.605620	3.83.1.251	TCP	54	32747 → 443 [ACK] Seq=2019 Ack=2442 Win=64952 Len=0
2817	192.168.1.137	6.611290	3.83.1.251	TCP	1466	32747 → 443 [ACK] Seq=2019 Ack=2442 Win=64952 Len=1412 [TCP segment of a reassembled PDU]
2818	192.168.1.137	6.611290	3.83.1.251	TLSv1.2	807	Application Data
2823	192.168.1.137	6.786305	3.83.1.251	TCP	54	32747 → 443 [ACK] Seq=4184 Ack=3069 Win=64325 Len=0
2832	192.168.1.137	7.075997	3.83.1.251	TCP	1466	32747 → 443 [ACK] Seq=4184 Ack=3069 Win=64325 Len=1412 [TCP segment of a reassembled PDU]
2833	192.168.1.137	7.075997	3.83.1.251	TLSv1.2	782	Application Data
2842	192.168.1.137	7.251572	3.83.1.251	TCP	54	32747 → 443 [ACK] Seq=6324 Ack=3673 Win=63721 Len=0
3679	192.168.1.137	8.125452	3.83.1.251	TCP	1466	32747 → 443 [ACK] Seq=6324 Ack=3673 Win=63721 Len=1412 [TCP segment of a reassembled PDU]
3680	192.168.1.137	8.125452	3.83.1.251	TLSv1.2	792	Application Data
3684	192.168.1.137	8.302265	3.83.1.251	TCP	54	32747 → 443 [ACK] Seq=8474 Ack=4406 Win=64952 Len=0

Figura 38: Processo de *handshake* e troca de chaves



```
> Frame 2818: 807 bytes on wire (6456 bits), 807 bytes captured (6456 bits) on interface 0
> Ethernet II, Src: Giga-Byt_f8:63:55 (e0:d5:5e:f8:63:55), Dst: HuaweiTe_82:e9:e9 (48:7b:6b:82:e9:e9)
> Internet Protocol Version 4, Src: 192.168.1.137, Dst: 3.83.1.251
> Transmission Control Protocol, Src Port: 32747, Dst Port: 443, Seq: 3431, Ack: 2442, Len: 753
> [2 Reassembled TCP Segments (2165 bytes): #2817(1412), #2818(753)]
v Transport Layer Security
  v TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 2160
      Encrypted Application Data: 694d791126791dd48818ad0271fe05a8bdb80a77c19b00b0…
```

```
0000  48 7b 6b 82 e9 e9 e0 d5  5e f8 63 55 08 00 45 00   H{k····· ^·cU··E·
0010  03 19 82 30 40 00 80 06  ae 2f c0 a8 01 89 03 53   ···0@··· ·/·····S
0020  01 fb 7f eb 01 bb b8 a4  ab 4f 90 1b 0f f3 50 18   ········ ·O····P·
0030  fd b8 fe 03 00 00 75 f2  1d 20 f0 3d 1e 20 fb fb   ······u· · ·=· ··
0040  80 d7 53 3e 24 e7 12 ef  07 22 d8 14 91 d6 9c 35   ··S>$··· ·"·····5
0050  d7 7b 16 d3 1f 1e 4d 85  cc 64 bb 5e 1e 3e e5 ce   ·{····M· ·d·^·>··
0060  26 aa 02 4f 92 5b 97 1b  dd 83 f9 45 13 c9 55 87   &··O·[·· ···E··U·
0070  0d 86 6f 0e 0c dd 9e 85  4c 42 2c 55 05 63 a9 cc   ··o····· LB,U·c··
0080  44 35 ad 78 12 5a c5 57  92 71 e9 63 97 5d e3 7f   D5·x·Z·W ·q·c·]··
0090  5c d6 5a c8 b9 37 9e d8  fe a0 b8 ad 18 ef e8 c7   \·Z··7·· ········
00a0  23 66 15 db f4 16 9f a2  a5 9f 67 50 2c 0b 17 0e   #f····· ··gP,···
00b0  27 3e 58 05 fb 4e 1c e7  ec ac 32 cb c9 3c a1 95   '>X··N·· ··2··<··
00c0  0f ca 8f 75 82 de 5d dd  29 c2 f1 d8 7d f5 7c 6b   ···u··]· )···}·|k
00d0  88 f8 76 cd e3 ce 14 92  05 0b 88 64 66 b4 18 a7   ··v····· ···df···
00e0  af e1 f7 ed b1 be 06 2e  e6 14 39 e5 80 f6 84 76   ········ ··9····v
00f0  31 c0 3c 29 ed f7 b0 43  81 d8 43 27 29 8a c2 24   1·<)···C ··C')··$
0100  e4 71 8a d9 bc 03 3a e9  11 4a f8 de d8 30 1b 8d   ·q····:· ·J···0··
0110  44 f4 25 19 a9 f5 8b bd  51 b9 30 07 78 ac 08 a1   D·%····· Q·0·x···
```

Figura 39: Transmissão de dados cifrados através do protocolo TLS v1.2

O facto do servidor poder ainda comprovar a sua autenticidade perante os seus utilizadores ajuda ainda a combater ataques de *phishing*, assumindo que o utilizador comprova a veracidade do certificado utilizado pelo *website* visitado.

- **Autenticação multi fator**: para prevenir o comprometimento da conta de um utilizador caso a sua *password* seja exposta, foi implementada no *website*

a autenticação multi fator. Assim, ao autenticar-se pela primeira vez num novo dispostivo, o utilizador deverá ter que comprovar a sua autenticidade mais uma vez através de um dos seguintes tipos de credenciais:

1. **One-time password (OTP)**: o utilizador deverá utilizar uma aplicação que gera uma *password* HMAC-based One-time Password algorithm (HOTP) de uso único com base numa *seed* gerada pelo *website* (*e. g.* através Google Authenticator) (M'Raihi et al., 2005);
2. ***Token* enviado por email**: neste caso é gerado um *token* secreto aleatório que é posteriormente enviado por email ao utilizador;
3. **Chave de segurança Universal 2nd Factor (U2F)**: um dispositivo que utiliza criptografia assimétrica em conjunto com o modelo *challenge-response* para autenticar o utilizador (Srinivas et al., 2017);
4. **Protocolo Secure, Quick, Reliable Login (SQRL)**: um protocolo *open source* que gera um URL através de um *nonce* utilizado para a autenticação dos utilizadores através de protocolos criptográficos de curva elíptica (Gibson Research Corporation, 2019).

Em cada uma das quatro formas de autenticação em cima é utilizado sempre algo que, em teoria, apenas o utilizador possui. Quer seja a *seed* utilizada pelo Google Authenticator, o dispositivo U2F único, a chave mestre de 256 bits de cada utilizador SQRL ou o acesso à sua conta email, cada um destes métodos de autenticação requer o uso de algo a que apenas o utilizador tem acesso e complementa o método de autenticação por omissão do *website*, o uso da *password* do utilizador. É conveniente ainda referir que tanto o terceiro como o quarto método de autenticação na lista em cima (respectivamente a chave de segurança Universal 2nd Factor (U2F) e o protocolo SQRL) conseguem prevenir ataques de MITM durante o processo de autenticação do utilizador.

- **Autenticação *clientside* dos *smart meters***: tal como foi referido no Capítulo 3, uma das grandes vulnerabilidades da solução da Smilics passa pela não autenticação dos *smart meters* perante o servidor. Isto leva a que a identificação destes dispositivos seja feita a partir do endereço MAC enviado como parâmetro dos pedidos GET onde são transmitidos os valores das leituras efectuadas pelo Mirubee para o servidor. Uma vez que os parâmetros de um pedido GET podem facilmente ser manipulados, um atacante pode, facilmente, fazer o *spoofing* do endereço MAC de um pedido GET enviado ao servidor da

Smilics e enviar para a conta de um utilizador legítimo leituras incorretas ou forjadas.

Para combater esta vulnerabilidade os Raspberry Pi nos vários *sites* do utilizador, ao inicializarem o processo de *forwarding* dos pacotes para o servidor deste projeto, terão necessariamente de se autenticar perante o servidor, utilizando as suas credenciais para efeito, transmitidas de maneira segura por HTTPS. A Figura 40 ilustra o processo de inicialização do *forwarder* mais especificamente o pedido de credenciais e respetiva autenticação deste perante o servidor.

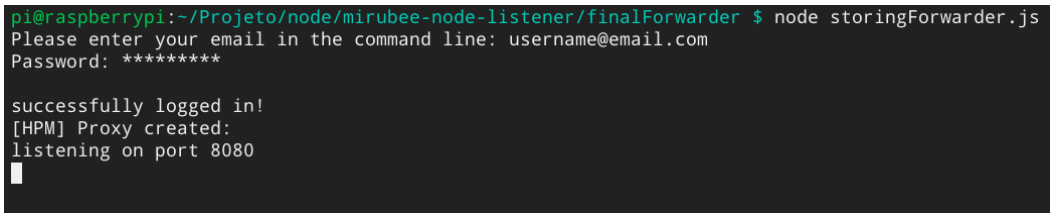


Figura 40: Autenticação do *forwarder* do Raspberry Pi

Desta forma, é possível assegurar a autenticidade das leituras recebidas pelo servidor e a integridade dos valores registados em cada *site* do utilizador.

- **Utilização de Object-Relational Mapping (ORM)**: a implementação do ORM Eloquent, protege o servidor deste projeto de ataques de injeção SQL.

  O excerto de código 9, em baixo, ilustra um dos vários cenários de uso do Eloquent nesta aplicação. Neste é feita uma consulta à tabela «`Devices`» pelo dispositivo cujo «`id`» corresponde ao «`id`» do pedido realizado pelo utilizador. Após realizada a consulta à base de dados, o ORM faz a conversão automática dos vários campos da tabela para o objecto PHP «`$device`» que após as validações devidas é editado e posteriormente são persistidas as alterações efetuadas na base de dados, tudo isto sem utilizar diretamente um único *statement* SQL.

Listagem 9: Exemplo de uso do ORM Eloquent (excerto retirado do ficheiro «DeviceController.php»)

```
79          $device = Device::where('id', $request->id)->first();

104         $device->name = $name;
105         $device->site_id = $site['id'];

125         $device->save();
```

Uma vez que todos os acessos à base de dados são realizados através do ORM Eloquent, torna-se desnecessária a criação de *queries* SQL geradas dinamicamente em *runtime* com base nos *inputs* enviados pelo utilizador. Mais, mesmo que um atacante envie *inputs* maliciosos para o servidor, com o intuito de quebrar a lógica do interpretador SQL, o Eloquent irá automaticamente fazer o devido processamento dos campos recebidos e mitigar qualquer tipo de ataque injeção SQL.

6.3.2 *Funcionalidades de Monitorização de Consumos Energéticos*

- **Consulta das Métricas Monitorizadas**: as leituras armazenadas na base de dados podem ser consultadas em qualquer altura ao consultar a página do *site* de onde estas são provenientes.

  Uma vez escolhido o *site* a consultar, o utilizador deverá filtrar as leituras a exibir pelo dispositivo associado, pela data em que estas foram registadas e ainda pela métrica a consultar. Após obter do servidor as leituras pedidas, o navegador do utilizador irá desenhar o respetivo gráfico de linhas permitindo ainda ao utilizador escolher quais as fases a exibir ou esconder. A Figura 41 ilustra o gráfico desenhado a partir dos valores da voltagem da fase 1 e 3, registados no dia 15 de janeiro de 2020 no *site* «ESTG (Leiria)».

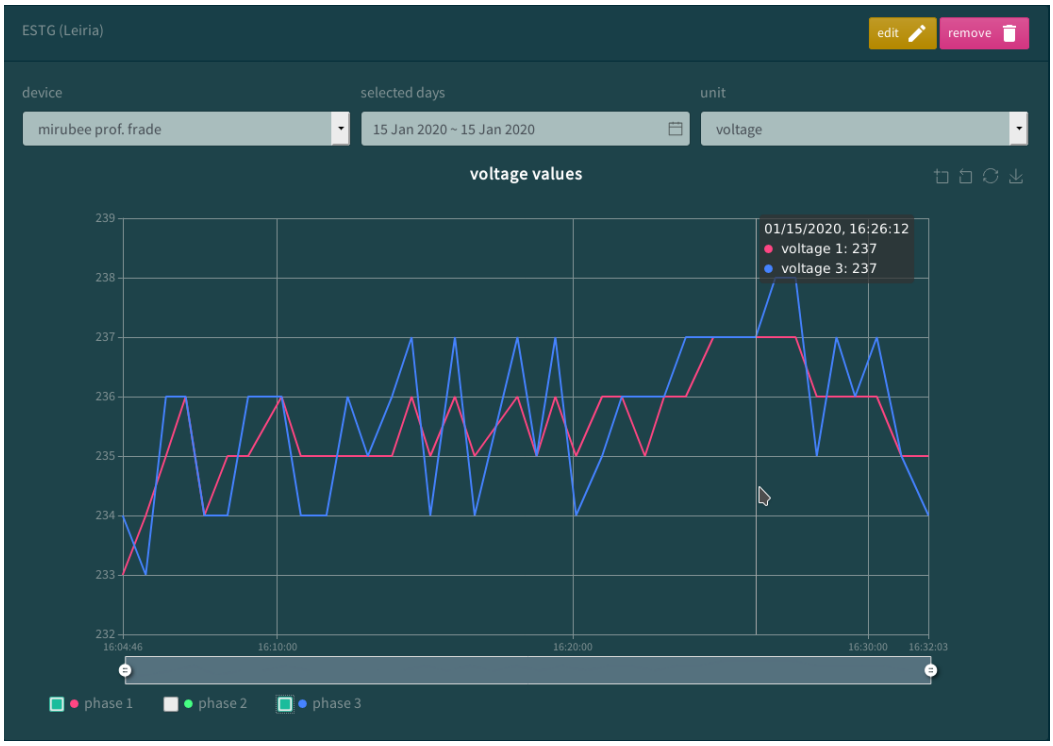


Figura 41: Consulta da voltagem da rede elétrica

No canto superior direito do gráfico o utilizador pode ainda definir o nível de zoom do gráfico, posicionar o gráfico consoante as suas necessidades e ainda fazer o download dos valores deste em formato Comma-separated values (CSV). Para além destas ferramentas, o utilizador pode utilizar também a barra em baixo para definir o nível de zoom e o posicionamento do gráfico.

- **Gestão de Tarifas**: para permitir ao utilizador simular os seus custos energéticos, torna-se necessário permitir a este definir os detalhes das tarifas energéticas de cada *site* que este monitorize através desta plataforma. Assim, para cada um dos *sites* do utilizador este deverá definir os seguintes parâmetros:
  - Potência contratada: a potência contratada pelo utilizador irá definir o número máximo de equipamentos ligados em simultâneo à rede elétrica do *site* monitorizado. Quanto maior a potência contratada pelo utilizador (em Quilovoltampere (kVA)) maior será a taxa diária paga por este;
  - Preço diário da potência contratada: este campo irá definir o custo diário da potência contratada no *site* monitorizado;

- Imposto: define a taxa, em percentagem, paga pelo utilizador na forma de imposto ao Estado. Em Portugal o imposto energético tipicamente assume o valor de 23%;
- Tipo de tarifa: define o tipo de tarifa energética bem como o número de horários de consumo do *site*. Pode assumir três tipos de tarifas diferentes: **simples** (em que as vinte e quatro horas do dia estão sujeitas ao mesmo custo de consumo), **bi-horária** (divide as vinte e quatros do dia em dois horários: horas fora de vazio e horas de vazio, ambas com custos de consumo distintos) e **tri-horária** (divide as vinte e quatros do dia em dois horários: horas de ponta, horas cheias e horas de vazio, todas com custos de consumo distintos);
- Custo de consumo: define o custo de consumo em Euros para cada um dos horários estabelecido pelo tipo de tarifa definido anteriormente;
- Horas de começo dos regimes horários da tarifa: define a hora que cada regime horário da tarifa começa, necessário para o cálculo do custo do consumo elétrico.

A Figura 42 ilustra a UI do *site* desenvolvido através do qual o utilizador deverá definir os detalhes da tarifa energética dos seus *sites*.

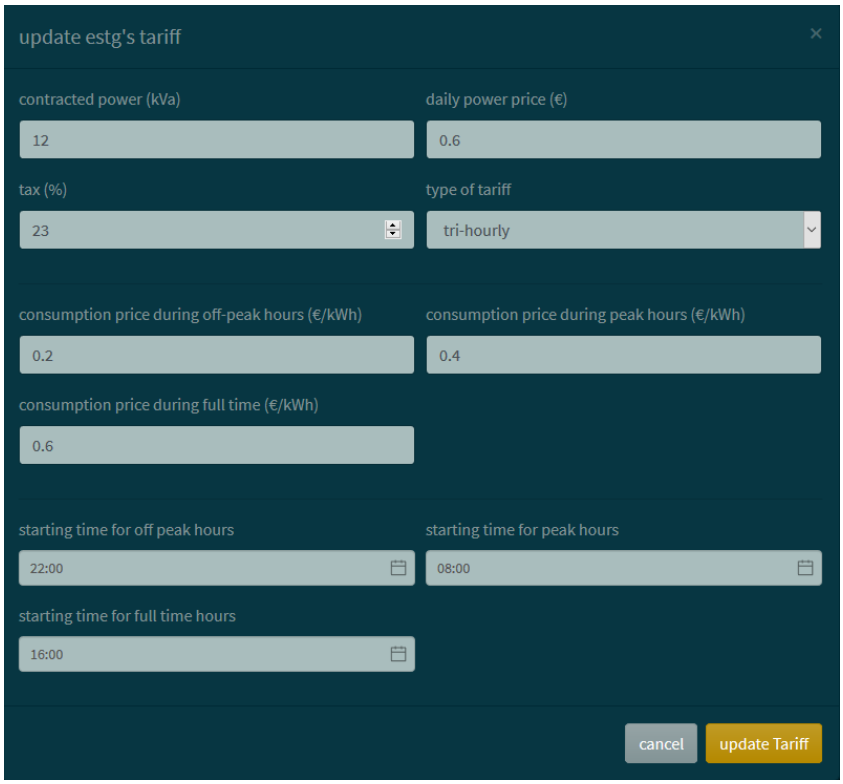


Figura 42: Janela de gestão da tarifa energética

Os dados inseridos pelo utilizador serão então utilizados pela plataforma para fazer o cálculo do consumo energético mensal final através da seguinte fórmula:

$$\textit{Preço do consumo elétrico } (€) = [(\textit{Consumo energético } (kWh) \times \textit{Preço da Tarifa } (€/kWh)) + (\textit{Preço diário da potência contratada} \times \textit{Número de dias do mês})] \times [(1 + (\textit{Imposto associado} \times 0{,}01))]$$

- **Relatório mensal da rede elétrica**: o utilizador pode, atráves desta funcionalidade, consultar comodamente uma síntese mensal das leituras elétricas para cada um dos seus *sites*. Para além disto o utilizador pode ainda consultar o seu consumo mensal (em kWh) e o seu custo mensal (em Euros) tendo em conta os dados da tarifa associada ao *site* em questão.

  O relatório mensal é automaticamente ajustado consoante o tipo de tarifa energética (simples, bi-horária ou tri-horária) para mostrar os gráficos e tabelas que melhor se ajustam a cada cenário.

  As figuras 43 e 44 são respetivamente um gráfico e uma tabela de simulação do custo mensal gerados automaticamente pelo *website*. A partir destas ferramentas de visualização o utilizador consegue mais facilmente perceber a natureza dos seus consumos, sendo discriminado o consumo em cada dia e em cada regime horário da sua tarifa energética;

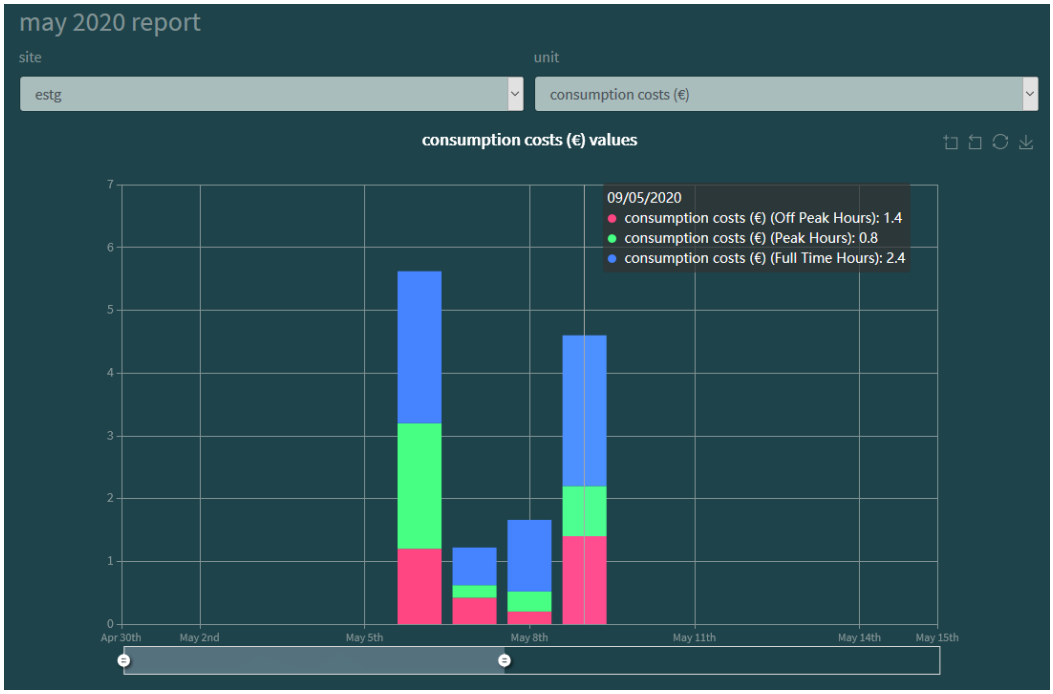


Figura 43: Gráfico do custo energético mensal de uma tarifa tri-horária

Figura 44: Simulação dos custos mensais de uma tarifa tri-horária

- **Gestão de Alertas**: através do *website*, o utilizador pode ainda criar e gerir alertas para as várias métricas monitorizadas, conforme ilustrado na Figura 45.

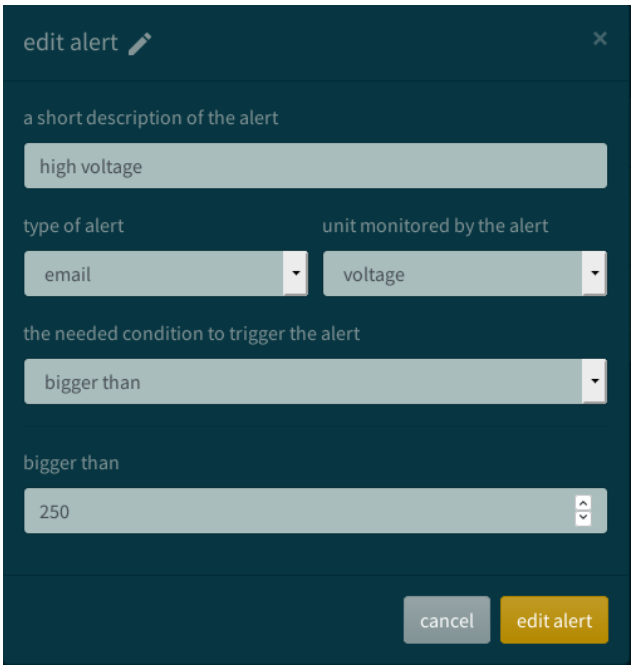


Figura 45: Janela de gestão de alertas

Os alertas, quando despoletados, podem ser transmitidos através do *website* ou através do envio de um email para o endereço de correio eletrónico do utilizador.

Os novos utilizadores da plataforma terão ainda uma série de alertas pré definidos, considerados relevantes para a monitorização da rede elétrica (*e. g.* alertas para monitorizar quando a tensão da rede elétrica desce os 195 Volts ou ultrapassa os 253 Volts (ERSE, 2017) (CENELEC, 2001)).

Ao criar o alerta o utilizador deve escolher a métrica a monitorizar, e ainda definir a condição que deve ser satisfeita para despoletar o alerta. A condição criada pode ser de quatro tipos distintos: «maior do que», «menor do que», «igual a» ou «entre» sendo que o após estabelecida o tipo a condição o utilizador deve ainda definir o(s) valor(es) limite da condição;

- **Gestão de Projetos e de *smart meters***: ao receber os dados de monitorização da rede elétrica de um dispositivo com um endereço MAC novo, o servidor irá automaticamente registar um novo dispositivo na sua base de dados e associá-lo-á à conta do utilizador em questão, bem como às leituras enviadas pelo *smart meter*.

  Conforme ilustrado na Figura 46, cabe então ao utilizador definir o *site* ao qual o novo dispositivo pertence, podendo ainda escolher um nome a associar a este para poder mais facilmente identificá-lo no futuro.

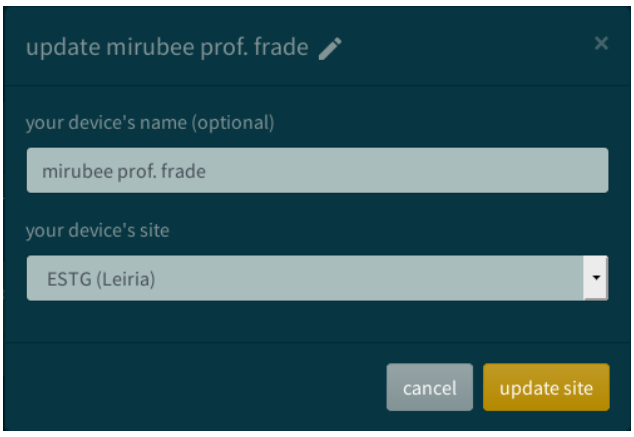


Figura 46: Janela de gestão de dispositivos

Uma vez configurado, o dispositivo passará a aparecer na lista de dispositivos na página do *site* associado a este permitindo ao utilizador consultar os dados registados por este. O utilizador pode ainda verificar o estado do dispositivo na página de detalhes deste conforme ilustrado na Figura 47.

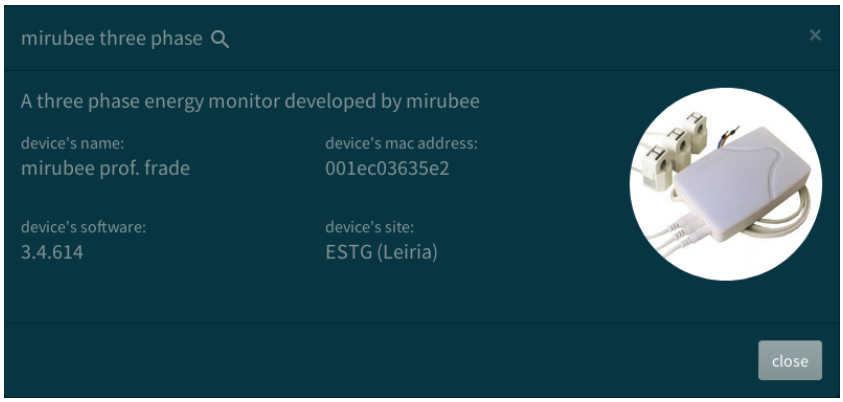


Figura 47: Janela de detalhes de dispositivos

6.4 SÍNTESE

A arquitetura proposta neste Capítulo em conjunção com os mecanismos de segurança implementados traduzem-se num nível de segurança maior de que aquele proporcionado pela solução original da Smilics. Mais, as funcionalidades de monitorização da rede elétrica deste *site* oferecem um nível de controlo maior ao utilizador, permitindo a este monitorizar e controlar com maior granularidade as leituras realizadas pelos seus *smart meters*.

7

DESENVOLVIMENTO DA PLATAFORMA WEB

Este Capítulo servirá para documentar em maior detalhe o processo de desenvolvimento dos vários componentes da plataforma web.

Uma vez que é impossível abordar na sua íntegra o processo de desenvolvimento desta plataforma, serão utilizados os exemplos considerados mais relevantes para melhor ilustrar e fundamentar as escolhas tomadas ao longo deste projeto, para cada um dos componentes desta. Serão ainda listadas e fundamentadas as principais tecnologias utilizadas durante o desenvolvimento desta plataforma.

7.1 IMPLEMENTAÇÃO DO *proxy* MIRUBEE

O *proxy* do Mirubee, que será executado por um Raspberry Pi, foi implementado em Node.js, um ambiente de execução de javascript que permite criar soluções web capazes de correr em vários sistemas operativos e em diferentes arquiteturas de processador, sendo assim uma *framework* bastante flexível e fácil de utilizar.

Mais, o Node.js permite criar soluções escaláveis devido ao facto deste ser bastante leve em termos de processamento e de permitir gerir um grande número de ligações em simultâneo. Para além disto, o Node.js vem acompanhado pelo Node Package Manager (NPM), o gestor de pacotes deste que permite facilmente integrar novos pacotes e funcionalidades nos projetos desenvolvidos com esta *framework* e permite ainda a fácil manutenção dos pacotes integrados.

7.1.1 *Instanciação do Endpoint e Envio de Leituras*

Um dos principais pacotes disponibilizados pelo NPM é o «`Express`», que permite criar *endpoints* HTTP e HTTPS permitindo ao utilizador definir funcões de *callback* que serão executadas sempre que a aplicação Node receba um pedido no *endpoint* criado.

O excerto de código 10, em baixo, foi retirado do código fonte do *proxy* do Mirubee desenvolvido no âmbito deste projeto e ilustra a integração dos pacotes «`Express`» e «`Axios`» para respectivamente criar *endpoints* HTTP e fazer pedidos HTTPS. Conforme referido anteriormente, o *proxy* deve autenticar-se perante o servidor da plataforma com as credenciais do utilizador para garantir a autenticidade dos pedidos recebidos pelo servidor.

Listagem 10: Configuração do *endpoint* GET do *proxy* Mirubee (excerto do código fonte do *proxy* do Mirubee)

```
1  var express = require('express');
2  var app = express();
3  var axios = require('axios');
```

```
43 function login(userEmail, userPassword){
44     return axios.post(websiteIp + '/api/login', { email : userEmail, password :
       ↪  userPassword })
45     .then( success => {
46         let data = success.data;
47         let tokenType = data.token_type;
48         let token = data.access_token;
49 
50         console.log('successfully logged in!')
51 
52         axios.defaults.headers.common['Authorization'] = tokenType + " " + token;
53 
54         app.get('/',  onMirubeeReq);
55         app.listen(8080)
56         console.log('listening on port 8080')
57 
58     }).catch(error => {
59         console.log(error.response.data.msg);
60         process.exit()
61     });
62 }
63 
64 function onMirubeeReq(req, res) {
65     console.log('new request')
66     var url =req.path;
67     var getstr = url.split('?')[1].split('&');
68     var get = processGetStr(getstr);
69 
70     console.log('number of fields on the GET Request: ',
       ↪  Object.keys(get).length);
71 
72     axios.post(websiteIp+ '/api/readings/threephase', get).
73     then(response => {
74         console.log(response.data);
75     })
76     .catch(error => {
77         console.log(error.response);
78     });
79 }
```

Após o utilizador definir as suas credenciais, o *proxy* irá autenticar-se, através da função «`login`». Esta função utiliza o pacote «`Axios`» para realizar o pedido POST

de login ao servidor; caso as credenciais sejam válidas, o servidor irá responder com um *token* que será utilizado no cabeçalhos dos restantes pedidos efetuados ao servidor para garantir a autenticidade destes. De seguida é criado um *endpoint* GET, no porto 8080 e é definida a função de *callback* «`onMirubeeReq`» que será chamada sempre que o *proxy* receber um novo pedido GET com as leituras energéticas proveniente do Mirubee.

A função «`onMirubeeReq`» é responsável por fazer o *parsing* do pedido original e extrair de lá os parâmetros necessários para criar o novo pedido HTTPS que será enviado para a plataforma desenvolvida) para o *endpoint* «`api/readings/threephase`») através do pacote «`Axios`».

As aplicações desenvolvidas em Node.js podem ainda ser executadas através do pacote NPM «`Forever`» que permite executar aplicações Node em *background*, como se de um serviço se tratasse, e ainda detetar eventuais *crashes* de execução e automaticamente reiniciar a aplicação em questão. Para executar uma aplicação Node através do Forever o utilizador deverá utilizar o comando: «`forever start app.js`».

Na figura 48 encontra-se representado o diagrama de sequência que ilustra o normal fluxo de execução deste *proxy*, bem como a maneira como este interage com o Mirubee e com o servidor desta aplicação.

7.2 IMPLEMENTAÇÃO DO SERVIDOR (*back-end*) DA PLATAFORMA WEB

Implementado através da *framework* Laravel e de uma base de dados Mysql, o servidor deste projeto atua como o pilar da plataforma desenvolvida, é responsável por receber processar e armazenar todas as leituras energéticas enviadas pelos *proxies* dos Mirubees dos utilizadores, processar os pedidos dos clientes e responder com o conteúdo indicado, garantir que não é quebrada a lógica da aplicação e que são seguidas as políticas de segurança e de acesso a conteúdos desta, entre outras funções tão ou mais fulcrais para o correto funcionamento da plataforma.

Foi ainda utilizado o Nginx para instanciar o servidor desta aplicação devido a este ser um solução leve, robusta e *open source*. O Nginx permite facilmente configurar os vários *websites* hospedados por uma máquina a partir dos seus ficheiros de configuração; na Listagem 11 encontra-se o conteúdo do ficheiro de configuração do servidor localizado na diretoria «`/etc/nginx/sites-enabled`».

Listagem 11: Ficheiro de configuração Nginx

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration

        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;

        ssl_certificate     /etc/ssl/mirubee.pem;
        ssl_certificate_key     /etc/ssl/mirubee.key;

        include snippets/ssl-params.conf;

        root /var/www/laravel/public;

        index index.php index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ /index.php?$query_string;
        }

        # pass PHP scripts to FastCGI server

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        }
}
```

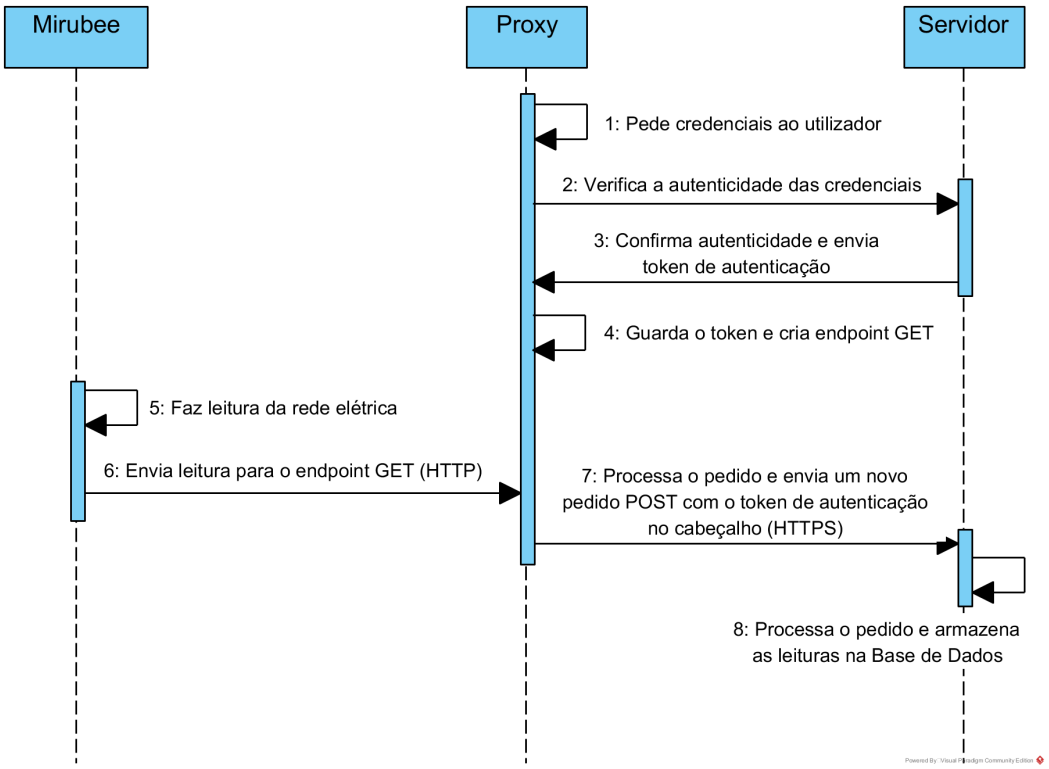


Figura 48: Diagrama de sequência do ciclo de vida do *proxy* do Mirubee

No ficheiro de configuração Nginx são estipulados os portos a utilizar pelo Nginx, a localização dos ficheiros hospedados e ainda as definições de SSL do servidor. Para além da localização do certificado e da chave SSL a utilizar pelo servidor, é ainda feita a inclusão de um *snippet* de configuração SSL adicional localizado na diretoria «`/etc/nginx/snippets/ssl-params.conf`», cujo conteúdo se encontra presente na Listagem 12. Neste *snippet* encontram-se presentes algumas das configurações consideradas como boas práticas no que toca ao *hardening* do protocolo SSL empregue por um servidor Nginx.

O Laravel é uma das *frameworks* PHP mais populares hoje em dia, tanto para projetos profissionais como académicos, principalmente devido às funcionalidades disponibilizadas por esta bem como à sua facilidade de utilização e implementação. Entre as principais funcionalidades disponibilizadas pelo Laravel destacam-se as seguintes:

- **Segurança**: O Laravel oferece aos desenvolvedores um conjuntos de soluções prontas a utilizar que ajudam a prevenir as principais vulnerabilidades de segurança encontradas hoje em dia em aplicações web. Quer sejam ataques de injeção SQL ou de Cross-Site Request Forgery (XSRF), o Laravel oferece

Listagem 12: Snippet de configuração SSL Nginx

```
ssl_protocols TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/ssl/certs/dhparam.pem;
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
ssl_session_timeout  20m;
ssl_session_cache shared:SSL:20m;
ssl_session_tickets off; # Requires nginx >= 1.5.9
ssl_stapling on; # Requires nginx >= 1.3.7
ssl_stapling_verify on; # Requires nginx => 1.3.7
resolver 1.1.1.1 8.8.8.8 valid=300s;
resolver_timeout 10s;
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

soluções de fácil implementação (muitas vezes já implementadas de raiz) que ajudam a prevenir estes tipos de vulnerabilidades;

- **Autenticação**: Quer seja através de *middleware* ou através de *guards*, ambos configurados pelo desenvolvedor, o Laravel permite configurar e proteger o acesso aos conteúdos hospedados por um servidor.

  Para além disto, o Laravel ainda oferece um bom grau de flexibilidade no que toca à autenticação dos utilizadores das plataformas implementadas através deste. Por exemplo através do serviço «Passport» o Laravel permite implementar a autenticação dos utilizadores através do *open standard* OAuth 2 (Hardt, 2012), indicado para projetos API nos quais a verificação da autenticidade dos utilizadores é feita através de *tokens* de autenticação enviados no cabeçalhos dos pedidos efetuados pelos clientes;

- ***Routing***: O sistema de routing do Laravel permite facilmente configurar os URI da aplicação e ainda associar cada um dos *endpoints* gerados a um *middleware* de forma a a proteger o acesso ao conteúdo disponibilizado pela rota em questão;

- ***Migration***: o sistema de *migrations* do Laravel facilita a configuração da estrutura da base de dados e ainda permite fazer o restauro desta de forma conveniente e segura.

Por outro lado foi decidido utilizar uma base de dados Mysql devido a esta ser uma solução *open source* e de utilização gratuita que oferece um grau robusto de segurança, flexibilidade e escalabilidade aos projetos implementados através desta.

7.2.1 *Rotas do Servidor*

O excerto de código 13 ilustra a configuração de algumas das rotas API do servidor desta plataforma. Neste excerto são configurados dois grupos de rotas aos quais são aplicados diferentes *middlewares*:

- O primeiro grupo, composto por rotas utilizadas durante a autenticação multi fator é protegido pelo *middleware* base do Laravel «`auth:api`» que verifica se o pedido do utilizador possui um *token* de autenticação válido (obtido através da autenticação por *password*). Este *middleware* é responsável por processar o pedido caso o *token* deste seja válido e recusá-lo caso não o seja;
- O segundo grupo, composto por rotas relativas aos *sites* dos utilizadores, é protegido pelo *middleware* criado no âmbito deste projeto «`multi_factor_authentication:api`». Para além de verificar se o pedido do utilizador é acompanhado por *token* de autenticação válido, este *middleware* verifica se o utilizador tem a autenticação multi fator ativada e, caso esta esteja ativada, verifica ainda se o utilizador se encontra autenticado através desta.

Listagem 13: Configuração das rotas API do servidor (excerto do código fonte do servidor da plataforma web)

```
47 // authenticate though mfa
48 Route::group(['middleware' => 'auth:api'], function () {
49         Route::post('/mfa/auth/email',
           ↪  'MfaMethodController@authenticateThroughEmail');
50         Route::post('/mfa/auth/email/code',
           ↪  'MfaMethodController@sendAuthenticationEmail');
51         Route::post('/mfa/auth/google',
           ↪  'MfaMethodController@authenticateThroughGoogle');
52 }
53
54 // sites
55 Route::group(['middleware' => 'multi_factor_authentication:api'], function () {
56         Route::get('/sites/{siteId}/readings/{start}/{end}',
           ↪  'SiteController@getReadings');
57         Route::get('/sites/{siteId}/devices/', 'SiteController@getSiteDevices');
58         Route::get('/sites/{siteId}/devices/{deviceId}/readings/{start}/{end}',
           ↪  'SiteController@getDeviceReadings');
59         Route::post('/sites/{siteId}/tariffs/', 'SiteController@setTariff');
60         Route::put('/sites/{siteId}/tariffs/', 'SiteController@updateTariff');
61         Route::delete('/sites/{siteId}/tariffs/', 'SiteController@deleteTariff');
62 }
```

O tipo de *middleware* empregue para cada rota deve ser determinado consoante o grau de sensibilidade do conteúdo disponibilizado por esta de forma a garantir que este se encontra adequadamente protegido.

Pode-se comprovar assim que, através do *middlware* do Laravel, é possível controlar o acesso aos recursos disponibilizados pelo servidor consoante o seu grau de sensibilidade e deste modo impedir utilizadores maliciosos de acederem indevidamente a estes.

7.2.2 *Alertas Definidos pelo Utilizador*

Conforme referido anteriormente, esta plataforma permite aos utilizadores criar alertas personalizados de forma a facilitar a monitorização da rede elétrica nos locais onde se encontram os seus dispositivos. Para tal foi criada a tabela «`alerts`» que irá armazenar os detalhes dos alertas criados, o código presente no excerto 14 é responsável pela criação da tabela «`alerts`».

Listagem 14: *Migration* responsável pela criação da tabela alerts (excerto do código fonte do servidor da plataforma web)

```
14      public function up()
15      {
16          Schema::create('alerts', function (Blueprint $table) {
17              $table->bigIncrements('id');
18              $table->timestamps();
19
20              $table->unsignedBigInteger('user_id');
21              $table->text('name');
22              $table->enum('unit', ['voltage', 'current', 'apparent power',
                    ↪  'active power',  'frequency', 'power factor']);
23              $table->double('threshold');
24              $table->double('threshold2')->nullable();
25              $table->enum('type', ['email', 'website']);
26
27              $table->enum('condition', ['bigger than', 'lesser than', 'between',
                    ↪  'equal']);
28
29              $table->timestamp('last_sent_email')->nullable();
30          });
31
32
33          Schema::table('alerts', function($table) {
34              $table->foreign('user_id')->references('id')->on('users')->onDelete(
                    ↪  'cascade');;
35          });
36      }
```

É possível constatar que a tabela «`alerts`» irá armazenar a unidade a monitorizar, os valores limite que irão despoletar o alerta, o tipo de alerta (notificação no *website* ou email) e ainda o tipo de condição que terá de ser observada para o alerta ser despoletado.

De seguida foi definido o evento Laravel «`ReadingInserted`» que é despoletado pelo servidor sempre que é recebida uma nova leitura válida através do comando: «`event(new ReadingInserted($reading, $user);`».

Associado ao evento «`ReadingInserted`», está o listener «`ReadingAlertTrigger`» que irá processar a leitura que despoletou o evento e verificar se esta despoleta algum dos alertas do utilizador. O excerto de código 15, retirado do listener «`ReadingAlertTrigger`», é responsável por ir buscar à base de dados todos os alertas do utilizador e verificar se algum destes é despoletados pela leitura associada. Finalmente, caso seja verificado que a leitura despoleta algum dos alertas dos utilizador é utilizada a função «`user->notify()`» para enviar o alerta para o utilizador através do método definido por este durante a criação do evento.

Listagem 15: Processamento do evento «`ReadingInserted`» (excerto do código fonte do servidor da plataforma web)

```
34      public function handle(ReadingInserted $event)
35      {
36          $this->user = ($event->user);
37
38          $this->checkReportCreation($event->reading);
39          foreach($event->user->alerts as $alert){
40              $this->checkTrigger($alert, $event->reading);
41          }
42      }
```

```
140     public function fireAlert(Alert $alert, $readingUnitValue){
141         $sendMail = ($alert->type == "email") ? true : false;
142         $this->user->notify(new ReadingAlert($alert, $sendMail,
            ↪  $readingUnitValue));
143     }
```

7.2.3 *Armazenamento de Leituras*

O controlador «`ReadingThreePhaseController`» é responsável por processar os pedidos de armazenamento das leituras da rede elétrica, podemos observar através do excerto de código 16 que o controlador começa por verificar se existe algum dispositivo pertencente ao utilizador na base de dados com o mesmo endereço MAC que o dispositivo que mandou a leitura a ser processada. Caso não exista, é criado um novo dispositivo na base de dados que será posteriormente associado ao utilizador que enviou o pedido.

De seguida, é criada uma nova leitura, representada pela instanciação da classe «`ReadingThreePhase`» no objeto «`$reading`» cujas propriedades serão preenchi-

Listagem 16: Processamento e armazenamento de leituras da rede elétrica (excerto do código fonte do servidor da plataforma web)

```
49      public function store(Request $request)
50      {
51          $deviceCreated = false;
52
53          if(is_null(Device::where('mac_address', $request->mac)->where('user_id',
            ↪  Auth::user()->id)->first())){
54              $device = new Device;
55              $device->type = 'three phase right';
56              $device->mac_address = $request->mac;
57              $device->model = $request->model;
58              $device->soft = $request->soft;
59              $device->user_id = Auth::user()->id;
60              $device->product_id = 2;
61
62              $device->save();
63              $deviceCreated = true;
64          }
65
66          $reading = new ReadingThreePhase;
67          $this->populateReading($reading, $request);
68
69          $reading->calc_time = 1;
70          $reading->calc_day_week = strtolower(date('l', $request->time));
71          $reading->calc_day_month = date('j', $request->time);
72          $reading->calc_year = date('Y', $request->time);
73          $reading->calc_month = date('m', $request->time);
74          $reading->calc_hour = date('H', $request->time);
75          $reading->calc_minute = date('i', $request->time);
76
77          if(!$deviceCreated){
78              $device = Device::where('mac_address',
                ↪  $request->mac)->where('user_id', Auth::user()->id)->first();
79          }
80          $reading->device()->associate($device);
81          $reading->save();
82
83          event(new ReadingInserted($reading, Auth::user()));
84
85          return response()->json(['message' => 'a new reading was created',
86          'reading' => new ReadingThreePhaseResource($reading),
87          'device' => new DeviceResource($device)],
88          201);
89      }
```

das através do método «`populateReading($reading, $request)`» com base nos valores do pedido POST enviado pelo utilizador.

São então calculadas as propriedades auxiliares deste objeto que serão utilizadas pela aplicação cliente para facilitar o processamento desta aquando da criação dos gráficos desenhados por esta. Estas propriedades são facilmente identificadas através do prefixo «`calc`» utilizado para sinalizar que se trata de uma propriedade calculada com base nos valores originais.

Posto isto, a leitura é persistida na base de dados através da função «`$reading->save();`» e é gerado um novo alerta «`ReadingInserted`» que como foi explicado anteriormente será processado pelo listener «`ReadingAlertTrigger`» para aferir se a leitura despoletou algum dos alertas do utilizador.

Finalmente, o servidor envia ao *proxy* do Mirubee uma resposta a confirmar o armazenamento da leitura na base de dados.

7.3 IMPLEMENTAÇÃO DO CLIENTE (*front-end*) DA PLATAFORMA WEB

No contexto global da plataforma criada, a aplicação cliente é responsável por processar os *inputs* do utilizador processá-los, fazer pedidos HTTPS ao servidor da plataforma e desenhar a UI com base nas respostas do servidor. Como tal, o *front-end* desta plataforma web (executado através do *browser* web dos utilizadores) foi desenvolvido, em parte, utilizando a *framework open source* Vue.js, uma *framework* javascript que é utilizada principalmente para desenvolver aplicações Single Page Application (SPA), como é o caso deste cliente. O facto da aplicação cliente ser uma SPA torna a navegação desta mais fluída e mais eficiente devido ao facto de, ao mudar de vista, o *browser* do utilizador só ter necessidade de carregar os componentes necessários à nova vista, evitando assim ter de voltar a carregar a página web de raiz.

O Vue.js permite ainda criar aplicações modulares devido ao facto de utilizar «componentes» que atuam como blocos de código HTML e javascript que podem facilmente ser reutilizados em outros cenários. Para além disto, o Vue facilita ainda a interação entre a vista (o DOM da página web) e o modelo (objetos javascript) da aplicação cliente ao permitir a passagem de variáveis (*props*) entre componentes e ao permitir aos programadores despoletar eventos entre vistas que serão capturados por *listeners* e devidamente processados.

7.3.1 *Rotas do Cliente*

Apesar de ser uma SPA, o cliente web desta plataforma permite aos utilizadores navegarem por esta através da barra de endereços do seu *browser* (bem como através da função de retroceder e avançar do *browser* tal como se de uma apliacação Multi Page Application (MPA) se tratasse) uma vez que é utilizado o Vue Router que permite associar componentes Vue a rotas e vice versa. O excerto de código 17 ilustra a configuração das rotas utilizadas pela aplicação cliente.

Listagem 17: Configuração das rotas da aplicação cliente (excerto do código fonte do cliente da plataforma web)

```
76  const routes = [
77      { path: '/login', component: login },
78      { path: '/register', component: register },
79      { path: '/example', component: example },
80      { path: '/', component: mainComponent,
81          children: [
82              { path: '/sites/retriever', component: siteRetriever, name:
                ↪  'siteRetriever'},
83              { path: '/sites', component: sitePicker, name: 'sitePicker'},
84              { path: '/sites/:siteName', component: sites, name: 'sites',
                ↪  props:true},
85              { path: '/devices', component: devices, name:'devices', props:true},
86              { path: '/tariffs', component: tariffs, name:'tariffs', props:true},
87              { path: '/alerts', component: alerts, name:'alerts', props:true},
88              { path: '/reports', component: reports, name:'reports', props:true,
89                  children: [
90                      { path: '/reports/:year', component: reportsList,
                        ↪  name:'reportsList', props: true},
91                      { path: '/reports/:year/:month', component: monthlyReport,
                        ↪  name:'monthlyReport', props: true}
92                  ]
93              },
94              { path: '/mfa/setup', component: mfaSetup, name:'mfaSetup',
                ↪  props:true},
95              { path: '/mfa/setup/email', component: mfaSetupEmail,
                ↪  name:'mfaSetupEmail', props:true},
96              { path: '/mfa/setup/google', component: mfaSetupGoogle,
                ↪  name:'mfaSetupGoogle', props:true},
97              { path: '/mfa/setup/u2f', component: mfaSetupU2F,
                ↪  name:'mfaSetupU2F', props:true},
98
99                      { path: '/mfa/authentication', component: mfaAuthentication,
                        ↪  name:'mfaAuthentication', props:true},
100                     { path: '/mfa/authentication/email', component:
                        ↪  mfaAuthenticationEmail, name:'mfaAuthenticationEmail',
                        ↪  props:true},
101                     { path: '/mfa/authentication/google', component:
                        ↪  mfaAuthenticationGoogle, name:'mfaAuthenticationGoogle',
                        ↪  props:true},
102                     { path: '/mfa/authentication/u2f', component:
                        ↪  mfaAuthenticationU2F, name:'mfaAuthenticationU2F',
                        ↪  props:true},
103                     { path: '/mfa/authentication/sqrl', component:
                        ↪  mfaAuthenticationSQRL, name:'mfaAuthenticationSQRL',
                        ↪  props:true},
104         ]
105     },
106 ];
```

7.3.2 *Desenho de Gráficos*

Para além da *framework* Vue, o *front-end* desta aplicação utiliza ainda uma série de bibliotecas javascript para implementar as funcionalidades definidas aquando do levantamento de requisitos desta. Como tal, destaca-se o uso da biblioteca Echarts, distribuída sob a licença apache 2.0 (*open source*); esta biblioteca é resposável pelo desenho dos gráficos utilizados por esta aplicação, a figura 49 ilustra três tipos de gráficos distintos desenhados através da biblioteca Echarts.

Como se pode comprovar pela figura 49, durante o desenvolvimento da aplicação cliente foi tido o cuidado de utilizar o tipo de gráfico mais indicado para cada situação e contexto. No caso do gráfico de linhas (em cima) é utilizado para exibir os dados diários das leituras efetuadas pelo Mirubee da maneira mais granular possível, sendo possível ver a evolução destes ao longo do tempo em intervalos de meio em meio minuto.

O gráfico de caixa por outro lado, é utilizado no relatório das leituras mensais. Através deste, o utilizador consegue facilmente ter uma ideia da distribuição das leituras para cada dia do mês, uma vez que a partir da visualização deste é possível consultar a mediana, os limites superiores e inferiores e os quartis dos valores registados para cada métrica parametrizada pelo Mirubee.

Finalmente o gráfico de linhas é utilizado para exibir o custo diário (em Euros) para cada tarifa energética definida para o *site* em questão. O facto de ser empregue um gráfico de barras permite ao utilizador comparar visualmente, não só a diferença entre os valores de dia para dia, mas também de tarifa para tarifa dentro do mesmo dia.

Cada tipo de gráfico utilizado requer um formato específico de dados e como tal, após realizar um pedido HTTPS ao servidor e obter as leituras necessárias à construção dos gráficos, a aplicação cliente necessita de processar os dados em bruto de forma a criar as estruturas de dados necessárias para a biblioteca Echarts desenhar os gráficos pretendidos.

O excerto de código 18 é executado durante a criação do gráfico de linhas presente na figura 49. Para desenhar este gráfico, a biblioteca Echarts necessita de três series (uma para cada fase da rede elétrica) em que cada elemento individual da serie assume o valor de *array* composto pelo valor da leitura registada pelo Mirubee bem como o valor da data e hora em formato JSON. Mais, para configurar corretamente o gráfico desenhado torna-se necessário determinar os valores máximos e mínimos

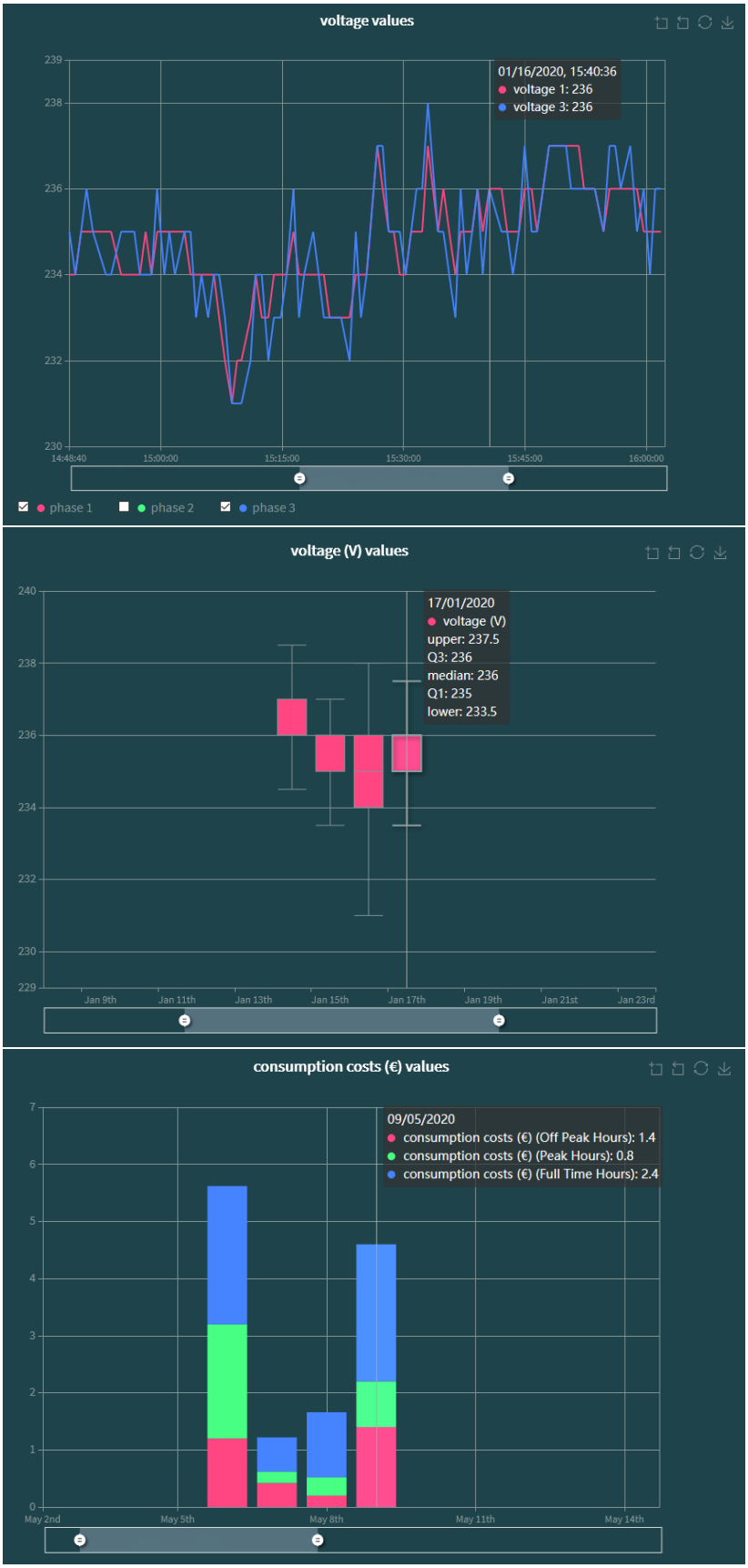


Figura 49: Exemplo dos tipos de gráficos utilizados neste projeto (de cima para baixo: gráfico de linhas, gráfico de caixa e gráfico de barras)

registados para cada fase de forma a definir os maximos e mínimos do eixo vertical do gráfico a desenhar.

Posto isto, após adquirir do servidor os dados em bruto das leituras, o cliente vai iterar os dados em questão para criar o *array* necessário, bem como verificar se o valor da leitura atual é superior ao valor máximo registado previamente bem como verificar se este é inferior ao valor mínimo registado previamente atualizando-os devidamente quando tal-se verifica.

Listagem 18: Processamento dos leituras provenientes do servidor (excerto do código fonte do cliente da plataforma web)

```
345 setupSeriesandDrawGraph(){
346     this.serie1 = [];
347     this.serie2 = [];
348     this.serie3 = [];
349 
350     let unit = this.units[this.selectedUnit].nameDB;
351     let lastTime = 0;
352 
353     this.readings.forEach(reading => {
354         if(lastTime + 30 < reading.time){
355 
356             let auxDate = new Date(reading.time*1000);
357             let Reading1 = reading[unit + '1'];
358             let Reading2 = reading[unit + '2'];
359             let Reading3 = reading[unit + '3'];
360 
361             if (this.yMax < Reading1) this.yMax = Reading1;
362             if (this.yMax2 < Reading2) this.yMax2 = Reading2;
363             if (this.yMax3 < Reading3) this.yMax3 = Reading3;
364 
365             if (this.yMin > Reading1) this.yMin = Reading1;
366             if (this.yMin2 > Reading2) this.yMin2 = Reading2;
367             if (this.yMin3 > Reading3) this.yMin3 = Reading3;
368 
369             this.serie1.push( [auxDate.toJSON(), Reading1] );
370             this.serie2.push( [auxDate.toJSON(), Reading2] );
371             this.serie3.push( [auxDate.toJSON(), Reading3] );
372 
373             lastTime = reading.time;
374         }
375     })
376 
377     this.refreshEchartSeries();
378 },
```

7.4 SÍNTESE

Com base na análise não exaustiva ao processo de desenvolvimento desta plataforma realizada neste Capítulo é possível verificar o cuidado constante que foi tido na correta implementação das funcionalidades desta ao longo deste projeto. Para além

disto foi também tido em conta quais os melhores tipos de tecnologias e soluções empregues nos vários componentes desta.

A conjunção de ambos estes fatores resulta numa plataforma web que, tanto quanto possível, tenta colmatar as principais vulnerabilidades encontradas nas soluções web atuais.

8

TESTES DE SEGURANÇA

De forma a testar a correta implementação dos mecanismos e funcionalidades de segurança desenvolvidos ao longo deste projeto foi realizado um conjunto de testes de segurança a esta plataforma que serão documentados neste Capítulo.

Inicialmente será feito um enquadramento do «*Test Bed*» utilizado no decorrer dos testes abordados, seguido pela análise dos quatro conjuntos de testes realizados. Será explicado o objetivo de cada teste realizado, as ferramentas utilizada para o efeito e ainda será analisado o resultado obtido para cada teste.

8.1 *test bed*

Os testes realizados neste Capítulo foram feitos através das ferramentas de segurança disponibilizadas no Sistema Operativo (SO) Kali Linux. Este SO é utilizado maioritariamente para realizar testes de penetração, uma vez que contem uma série de aplicativos e ferramentas pré-instalados (*e. g.* Metasploit, Hydra, Nmap, Wireshark) considerados fundamentais em qualquer teste de penetração.

O SO Kali Linux foi instanciado numa máquina virtual através do *software* VirtualBox. Esta máquina vitual está correr a *rolling release* mais recente do Kali Linux aquando da realização destes testes, lançada a doze de março de 2020 com o kernel «5.5.0-kali2-amd64» e tem dois Gigabytes de memória Random Access Memory (RAM) alocada.

Por outro lado, o servidor da plataforma web utilizado nestes cenários de teste encontra-se numa máquina virtual com o SO Ubuntu versão 18.04.2 LTS com o kernel Linux «4.15.0-47-generic». Esta máquina encontra-se ainda a correr a versão 7.3.4 da linguagem PHP e tem quatro Gigabytes de RAM alocada.

Ambas as máquinas virtuais encontram-se hospedadas pela mesma máquina Windows 10 versão 1909, com a *build* 18363.900 e com dezasseis Gigabytes de memória RAM dedicada.

Cada uma das máquinas virtuais utilizadas nestes testes possui dois adaptadores de rede, um em modo Network Address Translation (NAT) (para aceder à Internet quando necessário) e o outro em modo Host-only (para permitir a comunicação entre as duas máquinas virtuais bem como com a máquina anfitriã). Foi ainda alterado o ficheiro «`hosts`» do SO Kali e do SO Windows para que o *hostname* «`mirubee.test`» fosse resolvido com o endereço IP associado à maquina virtual com o SO Ubuntu.

Para além disto, foi criado um utilizador para efeitos de testes no *website* com o email «`username@email.com`» e com a *password* «`password`».

8.2 INJEÇÃO SQL

O primeiro conjunto de testes abordados neste Capítulo teve como objetivo testar a susceptibilidade do servidor da plataforma a ataques de injeção SQL. Conforme referido anteriormente, teoricamente, o uso do ORM Eloquent disponibilizado pela *framework* Laravel protege este servidor de ataques deste tipo no entanto, é necessário garantir a eficácia deste mecanismo de segurança em cenários práticos.

Para tal, foi utilizada a ferramenta *open source* «Sqlmap» que automatiza os testes de penetração em aplicações que utilizem bases de dados SQL. Assim, através desta ferramenta foram testados os *endpoints* do servidor desta plataforma para aferir a susceptibilidade de cada um destes a ataques de injeção SQL. Serão de seguida abordados em maior detalhe os testes realizados a três *endpoints*, mais especificamente: o *endpoint* «`https://mirubee.test/api/login`», «`https://mirubee.test/api/users/email/{userEmail}`» e «`https://mirubee.test/api/sites/{siteId}/devices/`»

8.2.1 *Endpoint «https://mirubee.test/api/login»*

O primeiro *endpoint* abordado é utilizado pelos utilizadores para fazer *login* e corresponde ao URI «`https://mirubee.test/api/login`». Este *endpoint* é um dos poucos nesta aplicação que pode ser acedido sem qualquer tipo de autenticação prévia uma vez que é utilizado para autenticar utilizadores e como tal é o *endpoint* mais facilmente acessível por parte de utilizadores externos. Posto isto, foi executado o seguinte comando:

Listagem 19: Comando executado para testar a injeção SQL no *endpoint* «`https://mirubee.test/api/login`»

```
sqlmap -u "https://mirubee.test/api/login" --dbms MySql --force-ssl --data
↪  "email=username@email.com&password=password" -p "email"
```

Na listagem 19 é definido o URI a testar, o Database Management System (DMBS) utilizado pelo servidor (o que em condições normais não seria conhecido), foi especificado que deveria ser utilizado o protocolo HTTPS através do parâmetro «`--force-ssl`», quais os dados a enviar no corpo do pedido através do parâmetro «`--data`» e quais os campos a testar através do parâmetro -p.

O *output* deste comando encontra-se no Apêndice A deste documento na Listagem 24 e é possível observar a partir deste que execução deste comando resulta num erro de autenticação o que é classificado pelo Sqlmap como um erro crítico.

Uma vez que este erro poderia ser causado por uma mal configuração dos cabeçalhos HTTP do pedido feito ao servidor, foi utilizada a ferramenta Burp Suite para interceptar um pedido HTTPS de *login* legítimo efetuado através do *browser* Firefox ilustrado na figura 50.

Este pedido foi então guardado no ficheiro de texto intitulado de «`login`» que foi utilizado pelo Sqlmap para refazer o teste de injeção SQL ao *endpoint* «`https://mirubee.test/api/login`» através do comando em baixo:

Listagem 20: Comando executado para testar a injeção SQL no *endpoint* «`https://mirubee.test/api/login`» com base num pedido HTTPS interceptado

```
sqlmap -r Documents/sql_injection/requests/https/login --dbms MySql --force-ssl
↪  -p "email"
```

É possível observar que na Listagem 20 em vez de ser definido o URI a testar é definido o ficheiro no qual se encontram os detalhes do pedido a efetuar ao servidor, neste caso o ficheiro «`Documents/sql_injection/requests/https/login`». O *output* deste comando encontra-se na listagem 25 no Apêndice A deste documento e resulta no mesmo erro encontrado no comando executado anteriormente.

Assim, chega-se à conclusão que o Sqlmap não detetou qualquer vulnerabilidade neste *endpoint* e que, tal como seria de esperar, o *endpoint* encontra-se devidamente protegido contra ataques de injeção SQL.

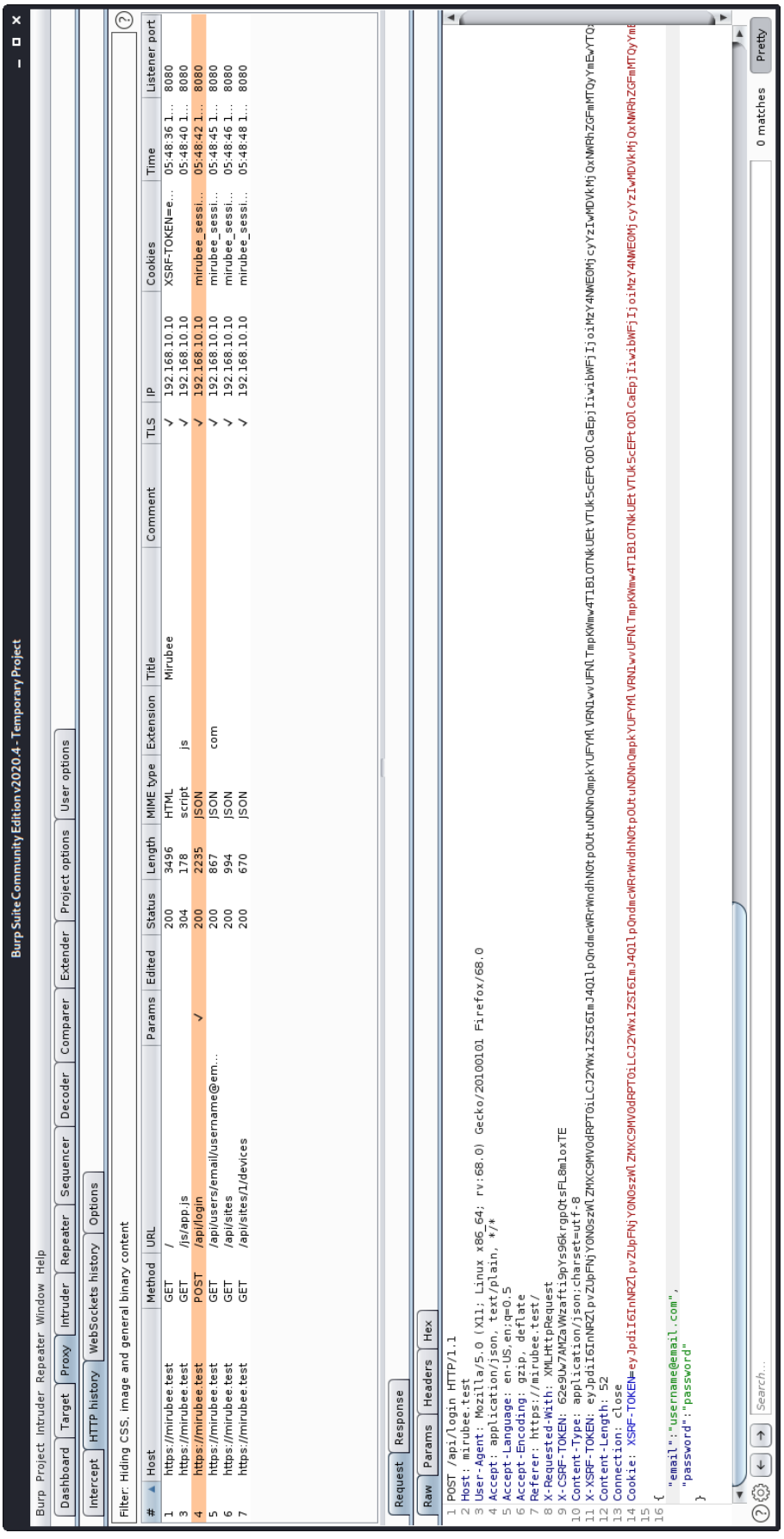


Figura 50: Intercepção do pedido HTTPS de *login*

Listagem 21: Comando executado para testar a injeção SQL no *endpoint* «`https://mirubee.test/api/users/email/username@email.com`»

```
sqlmap -u "https://mirubee.test/api/users/email/username@email.com*" --dbms
↪  MySql --force-ssl --level 5 --risk 3 --headers="Authorization: Bearer
↪  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImNkM2YwMWUyNWRlMzdkYjk2YzgxNDUx
↪  NDE4ZGI3Njg0YWIxZDc5ODYxYjBmNmUwZWQ5YTM4YjMwY2JjMjI4NzA2YTA3MjdlMGFlZTMwZTM2
↪  In0.eyJhdWQiOiIyIiwianRpIjoiY2QzZjAxZTI1ZGUzN2RiOTZjODE0NTE0MThkYjc2ODRhYjFk
↪  Nzk4NjFiMGY2ZTBlZDlhMzhiMzBjYmMyMjg3MDZhMDcyN2UwYWVlMzBlMzYiLCJpYXQiOjE1OTE5
↪  NTQ4MjMsIm5iZiI6MTU5MTk1NDgyMywiZXhwIjoxNjIzNDkwODIzLCJzdWIiOiIxIiwic2NvcGVz
↪  IjpbXX0.bmE0hvke99JnRooFtA3g5SO33YPcrZC0snvjasKjW15QpB3ZR24BnxyEyX7ClNhBqYMR
↪  c6viOk9-6kRyxvI6ERCE1J8QMG4jLGXAOx8gfbFWRRwwvBWlZXHyUfU5daVDW8deuDgXplKvTiuk
↪  6QlOVQv4AlJYnDiMtl6TPF3aVCGZRp7l9Yoy4YIViRRB0GrdOZMawGA6bvKC0jqme14y-C_SsGgm
↪  xHYnpJjgc9ArVt844xKtxFUVPW_Z5SsjwxAEpEjoY6aLk20fEyRIr1odzJHYlk_0XUucpbHm_yFf
↪  jt3awYdFzFZjbEslbcZbjaMXTosZOpHK9h1xNclhx-jPemjsH29iEn_hip2tSk39M7mlB86lN_tW
↪  Gw1ftI6yic8b5GuXPSYvuCp1ORabLkZMSRry9duyY0KBDlkT2m1c-DTRWd7VTVAkL0lxI-lSCeuY
↪  OERILX9n4LYpczEHjXXmxeOcHj_QF6Mvn08xPX09EQ5Di6TN9P3zvTbPebycCGhgJKWjkep2yqrE
↪  -4miebUShCanygL73cT_t9xZMrCTl5Zwce6fODYeEOeWYUmyVDm-vRyWHHb5K_Qj_5zim0SuWjCJT
↪  D9Z0vVBnXUQ32a1BFlUdcPEa2sO2HiPrYE5hCu1eqK4OkzyQhdlZa3BiFT9p_OR-gZKa3ebVD2zB
↪  sLpMLU"
```

8.2.2 *Endpoint «https://mirubee.test/api/users/email/{userEmail}»*

O segundo *endpoint* analisado é utilizado para consultar os dados de um utilizador e requer que o utilizador esteja autenticado na plataforma para aceder a este. Como tal torna-se necessário que os pedidos efetuados a este *endpoint* sejam acompanhados pelo cabeçalho HTTP «`Authorization`» e que este assuma o valor de um *token* de autenticação emitido pelo servidor quando o utilizador se autentica nesta plataforma.

Posto isto, foi executado o comando presente na Listagem 21 no qual é definido o URI do *endpoint* «`https://mirubee.test/api/users/email/username@email.com*`» e é definido o ponto de injeção deste a partir do caractere «`*`». É ainda estipulado o o DMBS utilizado pelo servidor da aplicação e é forçado o uso do protocolo HTTPS tal como aconteceu no comando Sqlmap anterior.

No entanto, este comando define alguns parâmetros novos que como tal serão de seguida explicados em maior detalhe:

- **`--level 5`** : define o nível de escrutínio a utilizar por parte do Sqlmap, pode assumir valores entre 1 a 5 sendo que quanto maior o valor deste parâmetro mais *payloads* serão testados pelo Sqlmap;
- **`--risk 3`**: define o grau do risco a tomar nos pedidos efetuados, pode assumir valores entre 1 a 3 sendo que quanto maior o valor deste parâmetro mais ousados serão os pedidos efetuados, no caso específico do nível 3 são feitos testes com base no operador SQL «`OR`»;

- **`--headers`**: define os cabeçalhos a adicionar aos pedidos realizados, neste caso foi adicionado o cabeçalho «`Authorization`» com o valor de um *token* de autenticação emitido previamente pelo servidor.

O *output* deste comando encontra-se presente no Apêndice A deste documento na Listagem 26 e, a partir deste verifica-se que o parâmetro testado não aparenta ser injetável com base nos 3264 pedidos que o Sqlmap realizou, sendo que 2929 dos quais resultaram em erros internos do servidor (código HTTP 500) e 335 resultaram no código de erro HTTP 429 («`Too Many Requests`») despoletado pelo servidor quando o utilizador ultrapassa o limite de pedidos estabelecido. O Sqlmap detetou ainda que, provavelmente, a aplicação alvo era protegida por uma Web Application Firewall (WAF) ou por um Intrusion Prevention System (IPS).

É conveniente realçar que o Sqlmap no final da sua execução recomendou que o teste fosse refeito desta vez com a inclusão dos seguintes parâmetros:

- **`--tamper=space2comment`**: define um *script* que será utilizado pelo Sqlmap para ofuscar a *string* a injetar e é utilizado principalmente quando é detetada a presença de uma WAF na aplicação alvo;
- **`--random-agent`**: esta *flag* sinaliza ao Sqlmap a necessidade de injetar nos pedidos realizados o cabeçalho HTTP «`User-Agent`» escolhido aleatoriamente com base numa série de valores predefinidos.

Mesmo após a inclusão destes dois parâmetros, o Sqlmap continuou sem detetar quaisquer indícios de injetabilidade por parte do parâmetro testado. Dos 3267 pedidos que o Sqlmap realizou na segunda iteração deste teste, 21 resultaram em erros internos do servidor (código HTTP 500) e 3246 resultaram no código de erro HTTP 404 («`Not Found`»).

8.2.3 *Endpoint «https://mirubee.test/api/sites/{siteId}/devices/»*

O terceiro e último *endpoint* testado é utilizado para consultar os dados dos dispositivo de um dos *sites* do utilizados em específico. Tal como acontece com o *endpoint* anterior, este *endpoint* necessita de um *token* de autenticação e como tal requer a presença do cabeçalho HTTP «`Authorization`».

O comando Sqlmap executado para testar este endpoint, presente na Listagem 22, utiliza os mesmo parâmetros utilizados no teste anterior divergindo deste apenas no URI a testar e, consequentemente, no parâmetro cuja injetabilidade vai ser testada.

Listagem 22: Comando executado para testar a injeção SQL no *endpoint* «`https://mirubee.test/api/sites/1*/devices/`»

```
sqlmap -u "https://mirubee.test/api/sites/1*/devices/" --dbms MySql --force-ssl
↪  --level 5 --risk 3 --headers="Authorization:  Bearer eyJ0eXAiOiJKV1QiLCJhbGc
↪  iOiJSUzI1NiIsImp0aSI6ImNkM2YwMWUyNWRlMzdkYjk2YzgxNDUxNDE4ZGI3Njg0YWIxZDc5ODY
↪  xYjBmNmUwZWQ5YTM4YjMwY2JjMjI4NzA2YTA3MjdlMGFlZTMwZTM2In0.eyJhdWQiOiIyIiwianR
↪  pIjoiY2QzZjAxZTI1ZGUzN2RiOTZjODE0NTE0MThkYjc2ODRhYjFkNzk4NjFiMGY2ZTBlZDlhMzh
↪  iMzBjYmMyMjg3MDZhMDcyN2UwYWVlMzBlMzYiLCJpYXQiOjE1OTE5NTQ4MjMsIm5iZiI6MTU5MTk
↪  1NDgyMywiZXhwIjoxNjIzNDkwODIzLCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.bmE0hvke99JnRoo
↪  FtA3g5SO33YPcrZC0snvjasKjW15QpB3ZR24BnxyEyX7ClNhBqYMRc6viOk9-6kRyxvI6ERCE1J8
↪  QMG4jLGXAOx8gfbFWRRwwvBWlZXHyUfU5daVDW8deuDgXplKvTiuk6QlOVQv4AlJYnDiMtl6TPF3
↪  aVCGZRp7l9Yoy4YIViRRB0GrdOZMawGA6bvKC0jqme14y-C_SsGgmxHYnpJjgc9ArVt844xKtxFU
↪  VPW_Z5SsjwxAEpEjoY6aLk20fEyRIr1odzJHYlk_0XUucpbHm_yFfjt3awYdFzFZjbEslbcZbjaM
↪  XTosZOpHK9h1xNclhx-jPemjsH29iEn_hip2tSk39M7mlB86lN_tWGw1ftI6yic8b5GuXPSYvuCp
↪  1ORabLkZMSRry9duyY0KBDlkT2m1c-DTRWd7VTVAkL0lxI-lSCeuYOERILX9n4LYpczEHjXXmxeO
↪  cHj_QF6Mvn08xPX09EQ5Di6TN9P3zvTbPebycCGhgJKWjkep2yqrE-4miebUShCanygL73cT_t9x
↪  ZMrCTl5Zwce6fODYeEOeWYUmyVDm-vRyWHHb5K_Qj_5zim0SuWjCJTD9Z0vVBnXUQ32a1BFlUdcP
↪  Ea2sO2HiPrYE5hCu1eqK4OkzyQhdlZa3BiFT9p_OR-gZKa3ebVD2zBsLpMLU"
```

neste caso será testada a injetabilidade do ID do *site* cujos dados dos dispositivos vão ser acedidos.

O *output* deste comando encontra-se Listagem 27 presente no Apêndice A deste documento e a partir deste verifica-se que o Sqlmap não encontrou qualquer indício do parâmetro testado ser injetável. Desta vez, foram realizados 2366 pedidos, sendo que num pedido foi devolvido o código de erro 405 («`Not Allowed`»), em 214 pedidos o código de erro 500 e em 1916 pedidos com o código erro 429.

Tal como no caso anterior foi recomendada a inclusão dos parâmetros `--tamper=space2comment` e `--random-agent` o que continuou a não revelar quaisquer indícios de injetabilidade. Desta vez foram feitos 3324 pedidos, 12 dos quais devolveram o código de erro 500 enquanto que 3212 dos quais devolveram o código 404.

É conveniente referir que, durante a execução de cada um dos três testes de injeção realizados, o servidor gerou uma série de *logs* com informação relativa aos vários erros de processamento de pedidos que ocorreram devido aos valores inesperados que foram assumidos pelos parâmetros cuja injetabilidade foi testada. Estes *logs* poderão ser eventualmente processados por um Security Information Event Management (SIEM) e ataques futuros deste género podem ser mais facilmente identificados e devidamente tratados.

8.3 QUEBRA DE AUTENTICAÇÃO

Com o intuito de testar a susceptibilidade desta aplicação a ataques de quebra de autenticação foi simulado um ataque de dicionário direcionado à conta do

utilizador com o email «`username@email.com`». Conveniente será referir que durante este teste a conta deste utilizador tinha a autenticação multi fator ativada (mais especificamente a autenticação por dispositivo U2F).

Para tal foi gerado um ficheiro de texto com uma série de possíveis *passwords* para simular um dicionário de credenciais geradas por um eventual atacante. Para além das *passwords* geradas aleatoriamente foi ainda adicionada a *password* real do utilizador («`password`») de modo a garantir que o ataque de dicionário fosse efetuado com sucesso.

Uma vez que esta aplicação se trata de uma SPA e como tal não usa *forms* HTTP tradicionais, foi utilizada a ferramenta Patator para realizar o ataque de dicionário em prole da alternativa mais popular Hydra. O Patator foi escolhido uma vez que este processa o sucesso da tentativa de login com base no código HTTP da resposta do servidor em vez de fazer o *parsing* do *form* devolvido pelo servidor (como acontece no Hydra) o que não é possível de fazer nesta aplicação devido ao facto desta ser uma SPA.

A figura 51 ilustra o *output* de execução do Patator e podemos ver que este recebe os seguintes argumentos:

- **`http_fuzz`**: define o módulo a usar por parte do Patator, neste caso o módulo «`http_fuzz`» foi desenvolvido para fazer ataques de força bruta HTTP e HTTPS;
- **`url=http://mirubee.test/api/login`**: o URL utilizado pelo Patator para fazer os pedidos HTTP;
- **`method=POST`**: define o método HTTP a utilizar, neste caso o método POST;
- **`body`**: define o corpo do pedido HTTP realizado, neste caso, uma vez que se trata de um pedido de autenticação este é composto pelo email do utilizador e pela *password* deste;
- **`0=passwords.txt`**: estipula o ficheiro que deve ser utilizado como lista de *passwords* no ataque de força bruta,
- **`-x ignore:code=401`**: define a condição que deve ser observada para o Patator definir uma tentativa como uma falha de autenticação, neste caso o Patator define uma tentativa de autenticação como falhada quando esta recebe uma resposta com o código HTTP 401;
- **`-l output_patator`**: define uma diretoria na qual será criado um ficheiro com o formato CSV e outro com o formato XML com os resultados do teste bem

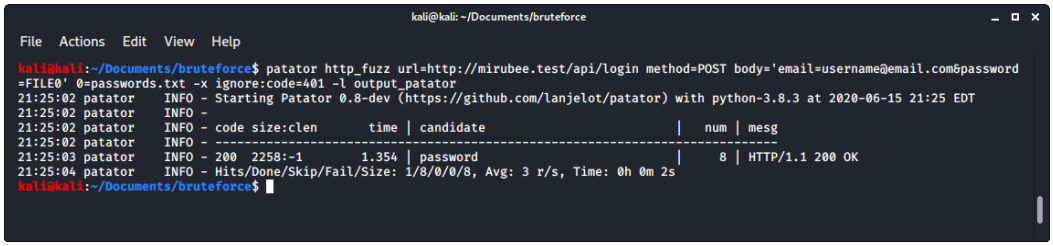


Figura 51: Ataque de dicionário efetuado com a ferramenta Patator

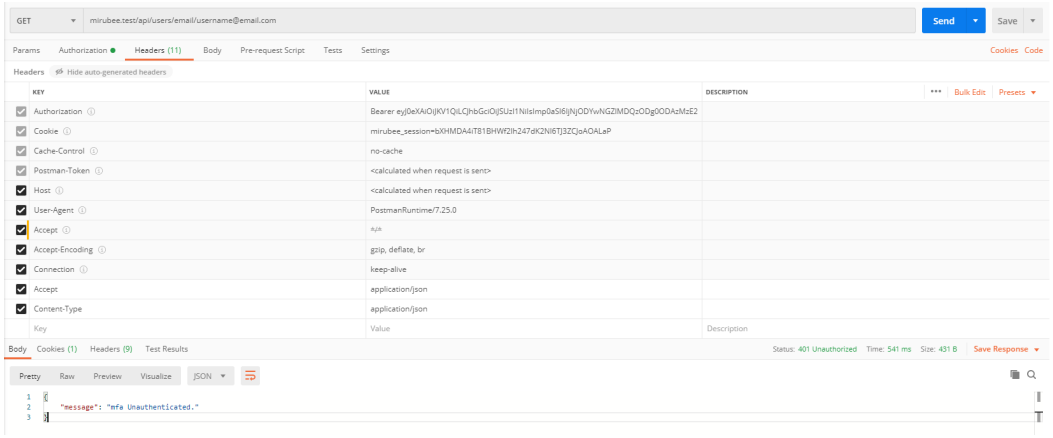


Figura 52: Pedido efetuado através da ferramenta Postman

como um ficheiro com qualquer pedido, e respetiva resposta, que o Patator classfique como um «`Hit`» (caso algum exista).

Pode-se observar com base na figura 51 que apenas um pedido de autenticação foi efetuado com sucesso quando a *password* enviada assumiu o valor «`password`». No Apêndice A deste documento, na Listagem 28 encontra-se o *output* do pedido e resposta classificados como um «`Hit`» pelo Patator no ataque efetuado na figura 51, é possível observar que na resposta do servidor, é devolvido um *token* de autenticação que deve ser incluído nos cabeçalhos dos pedidos efetuados pelo utilizador no futuro para este passar o primeiro passo de autenticação.

Posto isto, foi utilizada a ferramenta Postman para fazer um pedido ao servidor com o *token* de autenticação contido na resposta presente na Listagem 28 para aceder ao *endpoint* GET «`mirubee.test/api/users/email/username@email.com`». A figura 52 ilustra o *output* do Postman e é possível observar que, mesmo com a inclusão do *token* de segurança no cabeçalho «`Authorization`», o pedido é barrado pelo servidor uma vez que este utilizador tem a autenticação multi fator ativada e que este *endpoint* é protegido pelo *middleware* «`multi_factor_authentication`».

Assim, é possível comprovar que, apesar da plataforma desenvolvida não ser completamente imune ataques de quebra de autenticação, esta disponibiliza aos

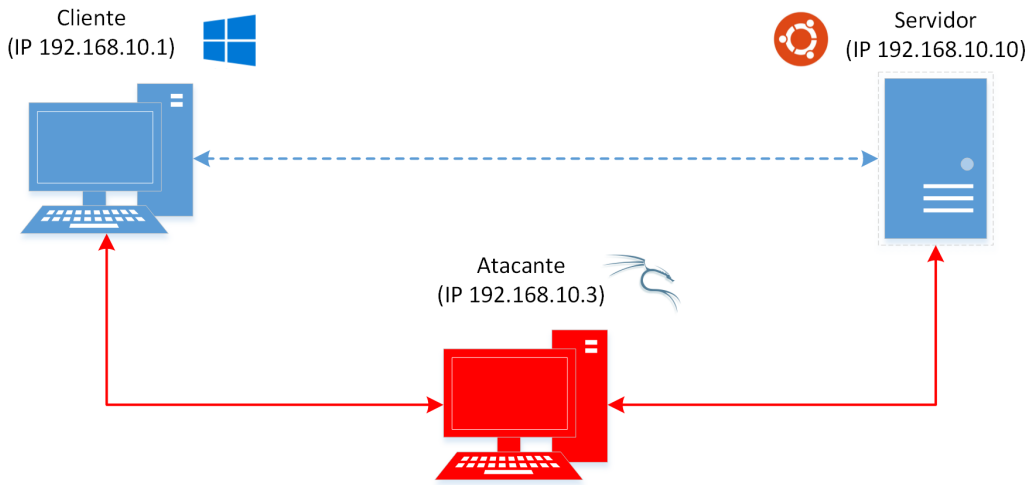


Figura 53: Esquema de rede do cenário de teste MITM

seus utilizadores as ferramentas necessárias para mitigar praticamente qualquer impacto causado por um ataque de força bruta ou de dicionário ao permitir que estes se autentiquem através de multiplos fatores de autenticação.

8.4 ATAQUES DE MAN-IN-THE-MIDDLE

Para testar a vulnerabilidade da plataforma a ataques de MITM foi criado um cenário semelhante ao teste anterior no qual a autenticação multi fator do utilizador «`username@email.com`» se encontrava ativada, mais concretamente a autenticação por dispositivo U2F.

Para este teste em específico foram utilizadas três máquinas distintas hospedadas na sub-rede 192.168.10.0/24. Mais especificamente foi utilizada a máquina Ubuntu (servidor da aplicação) no endereço IP 192.168.10.10, a máquina Kali (que atuará como atacante MITM) no endereço IP 192.168.10.3 e a máquina Windows 10 (que hospeda as duas máquinas virtuais) no endereço IP 192.168.10.1 que será o cliente da aplicação (o alvo do ataque MITM). O cenário criado encontra-se ilustrado resumidamente na figura 53.

Para interceptar os pacotes transmitidos entre o cliente e o servidor deste cenário foi utilizada a *framework* Bettercap. Podemos observar na Figura 54 que, através desta ferramenta foi inicialmente feita a descoberta de *endpoints* na rede 192.168.10.0/24, foi feito o *Spoofing* do Address Resolution Protocol (ARP) da máquina 192.168.10.1 (cliente) e de seguida foi ativado o módulo «`net.sniff`» para capturar os pacotes interceptados.

Figura 54: *Output* da *framework* Bettercap

Após ser iniciado o processo de captura de pacotes, foi feita a autenticação do utilizador «`username@email.com`» na máquina cliente, primeiro por *password* e posteriormente por dispositivo U2F. Os pacotes transmitidos durante este processo foram devidamente interceptados pela máquina atacante e foram guardados num ficheiro com o formato pcapng.

Com recurso à ferramenta Wireshark foi aberto o ficheiro pcapng e foi explorado o seu conteúdo. Pode-se observar na Figura 55 que, uma vez que é empregue o protocolo HTTPS para transmitir conteúdo entre o cliente e o servidor desta aplicação, este encontra-se cifrado e que dificilmente um atacante extrairia informação sensível com base nos pacotes capturados sem antes conseguir decifrá-los.

Posto isto, no âmbito de testar as limitações de segurança desta aplicação, será suposto que o atacante conseguiu contornar o processo de cifragem dos pacotes interceptados (apesar das dificuldades práticas que tal processo acarreta). Para simular a captura dos pacotes em *cleartext* foi utilizada a ferramenta Burp Suite (que instancia um *proxy* de rede que intercepta os pacotes recebidos) e foi alterada a configuração do *browser* do cliente para passar a utilizar este *proxy*. O Burp Suite passa então a interceptar os pacotes enviados pelo *browser* do cliente e passa forçar que este utilize um certificado SSL personalizado, emitido por este, para todos os *websites* HTTPS que o cliente visite. Os detalhes deste certificado podem ser consultados na figura 64, no Apêndice A deste documento.

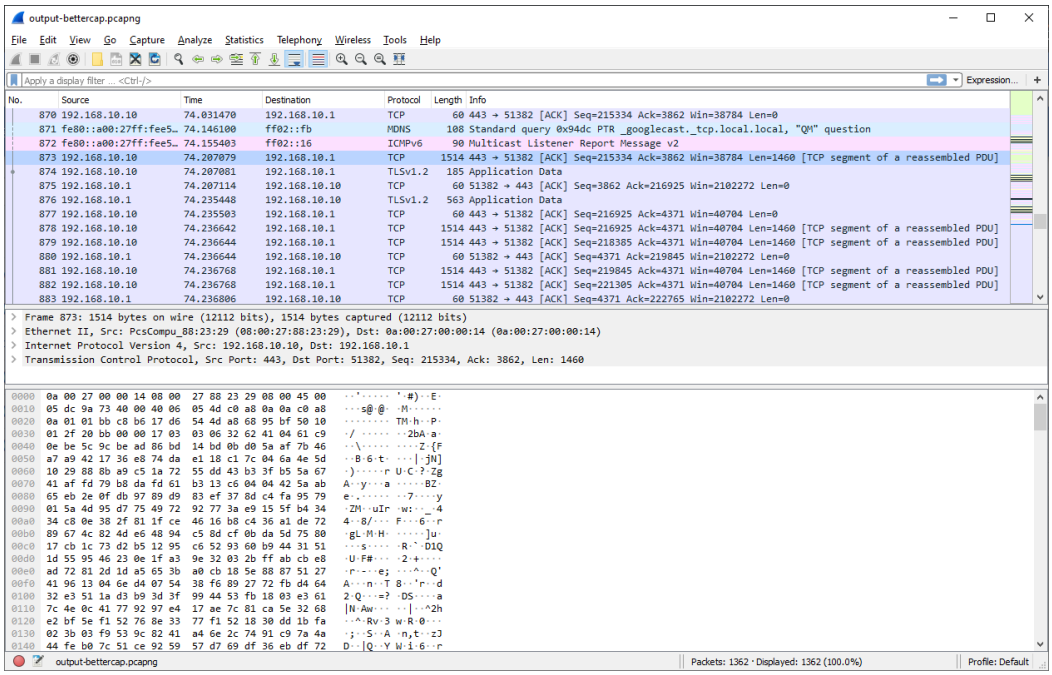


Figura 55: Pacotes transmitidos entre o cliente e o servidor

Uma vez que o certificado SSL utilizado para cifrar os pacotes entre o cliente e o *proxy* foi gerado pelo Burp Suite, este consegue facilmente decifrar os pacotes interceptados (que originalmente seriam direcionados ao servidor da aplicação). De seguida foi refeito o processo autenticação do utilizador «`username@email.com`» e foi capturado o tráfego gerado. Na figura 56 é possível ver os pacotes capturados pelo Burp Suite durante este processo, para além de ser possível ver a ordem dos *endpoints* utilizados durante o processo é ainda possível ver o conteúdo transmitido em *cleartext* tanto nos pedidos do cliente como nas respostas do servidor.

Através da análise dos pacotes capturados é possível chegar à conclusão que o processo de autenticação foi feito em dois passos (por *password* e posteriormente por dispositivo U2F) compostos por três pedidos:

1. **`api/login`**: o único pedido realizado no processo de autenticação por *password*, neste pedido POST o utilizador envia o seu email e *password* e recebe um *token* de autenticação;

2. **`api/u2f/getArgs`**: o primeiro de dois pedidos realizados no processo de autenticação por dispositivo U2F, neste pedido GET o utilizador pede ao servidor um *challenge* que lhe é transmitido na resposta a este pedido. A resposta capturada pelo Burp Suite que contem o *challenge* gerado pode ser consultada na Listagem 29 do Apêndice A deste documento;

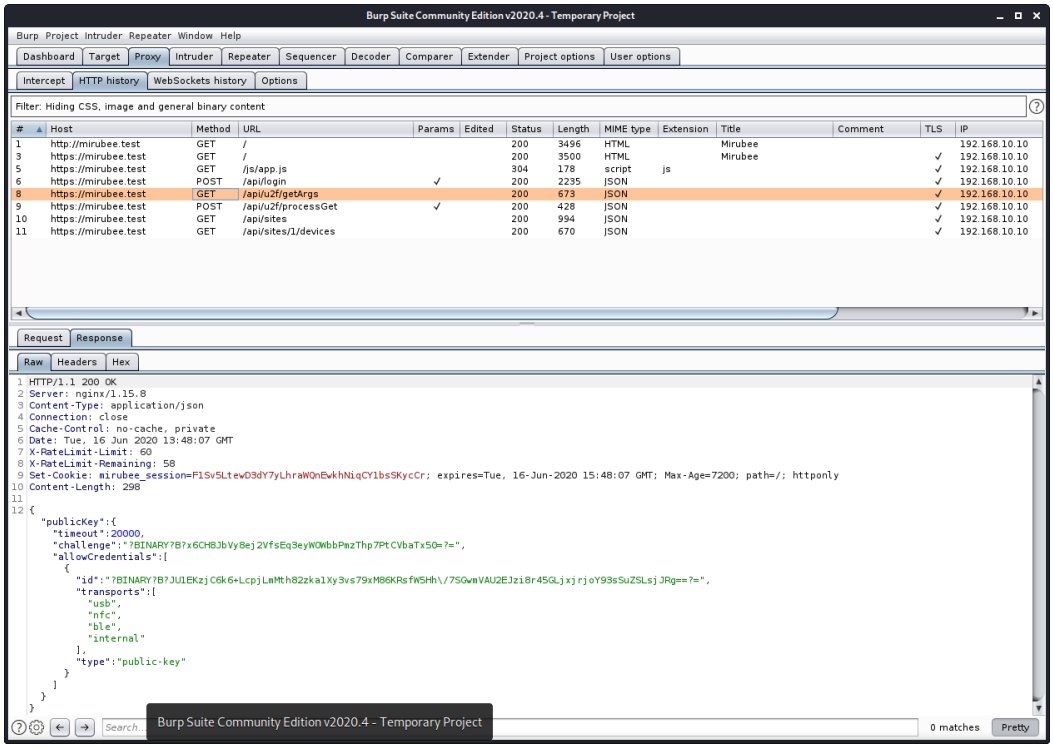


Figura 56: Pacotes capturados pelo Burp Suite

3. **`api/u2f/processGet`**: o segundo e último pedido realizado no processo de autenticação por dispositivo U2F, neste pedido POST o dispositivo U2F processa o *challenge* recebido no pedido anterior e envia a assinatura deste que será utilizada para autenticar o utilizador. Tanto o pedido como a resposta capturada pelo Burp Suite que contêm respetivamente a assinatura gerada pelo dispositivo U2F e a confirmação de autenticação do utilizador podem ser consultados nas listagens 30 e 31 do Apêndice A deste documento.

Finalmente, basta testar se é possível que um atacante se autentique ao reenviar o conteúdo dos pedidos interceptados. Teoricamente o atacante apenas seria capaz de passar no primeiro passo de autenticação (por *password*) uma vez que no segundo passo (autenticação U2F) as assinaturas geradas pelo dispositivo U2F são feitas com bases nos *challenges* gerados pelo servidor e são de uso singular. Para tal foi utilizado o Postman para reenviar as credenciais capturadas no passo anterior.

A Figura 57 capturada após reenviar a assinatura gerada pelo dispositivo U2F (que foi interceptada com o Burp Suite) comprova que, tal como seria expectável, o servidor devolve uma resposta com o código de erro 500 no eventual caso de um atacante reenviar uma assinatura previamente gerada pela chave U2F do cliente.

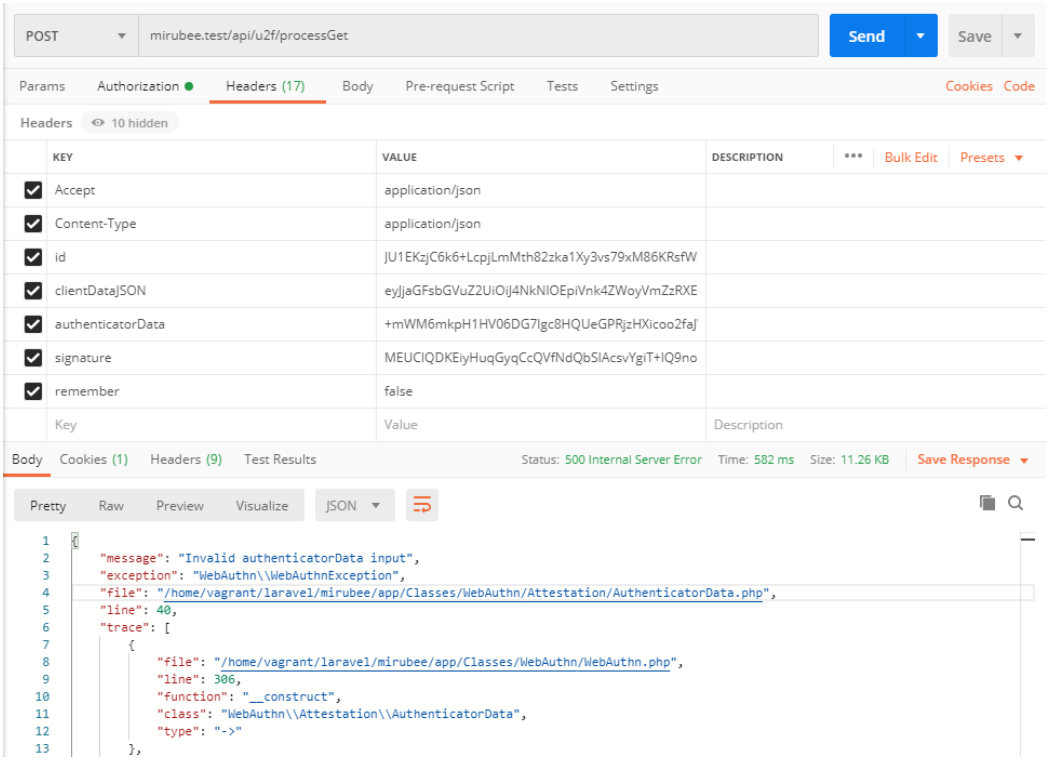


Figura 57: Erro de autenticação da assinatura reenviada

Com isto, é possível afirmar que esta aplicação encontra-se devidamente protegida contra ataques de MITM, uma vez que oferece um grau de proteção base satisfatório permitindo ainda que os seus utilizadores ativem a autenticação por múltiplos fatores de autenticação aumentando ainda mais o seu nível de proteção contra ataques deste género.

8.5 CROSS-SITE SCRIPTING

Para testar a capacidade desta plataforma de combater ataques de XSS, inicialmente foram realizados testes manuais de XSS à plataforma mas não foi encontrada nenhuma vulnerabilidade a partir destes. A figura 58 ilustra o resultado de um destes testes, no qual foi enviada a *string* presente na Listagem 23 num pedido de alteração de um dos dispositivos do utilizador.

Listagem 23: Código injetado na *string* envida ao servidor

```
1 <script>alert("Hello! I am an alert box!!")</script>
```

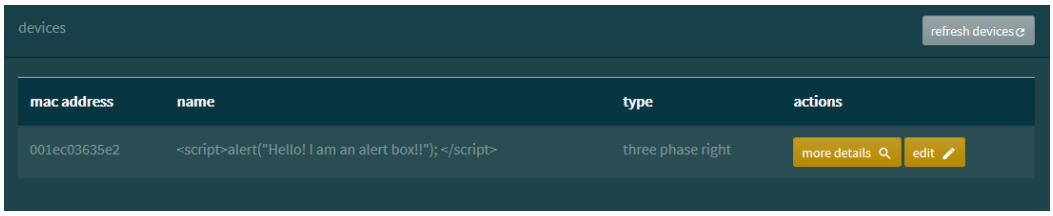


Figura 58: Teste manual de Cross-Site Scripting

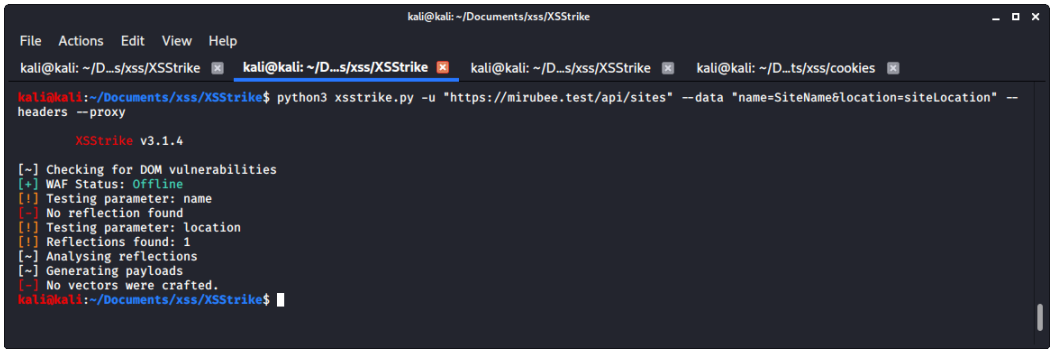


Figura 59: Teste de Cross-Site Scripting no *endpoint* «api/sites»

Caso este componente em específico fosse vulnerável a este tipo de ataque, o utilizador, ao carregar esta página iria receber um alerta do *browser* com a frase «Hello! I am an alert box!!». No entanto, uma vez que tal não se verificou e que em vez disso foi feito o carregamento normal da página em questão, é possível afirmar que, pelo menos para este tipo de ataque XSS, este componente encontra-se protegido.

Uma vez que não foi possível detetar qualquer tipo de vulnerabilidades de XSS com testes manuais, foi utilizada a ferramenta XSStrike para realizar testes automatizados em todos os *endpoints* passíveis de ser vulneráveis a este tipo de ataque. A Figura 59 ilustra o uso desta ferramenta para testar o *endpoint* POST «`mirubee.test/api/sites`».

É possível verificar que a ferramenta XSStrike aceita os seguintes argumentos de entrada:

- **`-u`**: define o URI a testar;
- **`-data`**: define o corpo do pedido e consequentemente quais os parâmetros a testar;
- **`--headers`**: ao receber este argumento, o XSStrike irá solicitar que utilizador defina quais os cabeçalhos HTTP a utilizar;
- **`--proxy`**: este argumento transmite ao XSStrike que este deve utilizar o *proxy* de rede que se encontra definido no ficheiro de configuração deste. Nesta caso

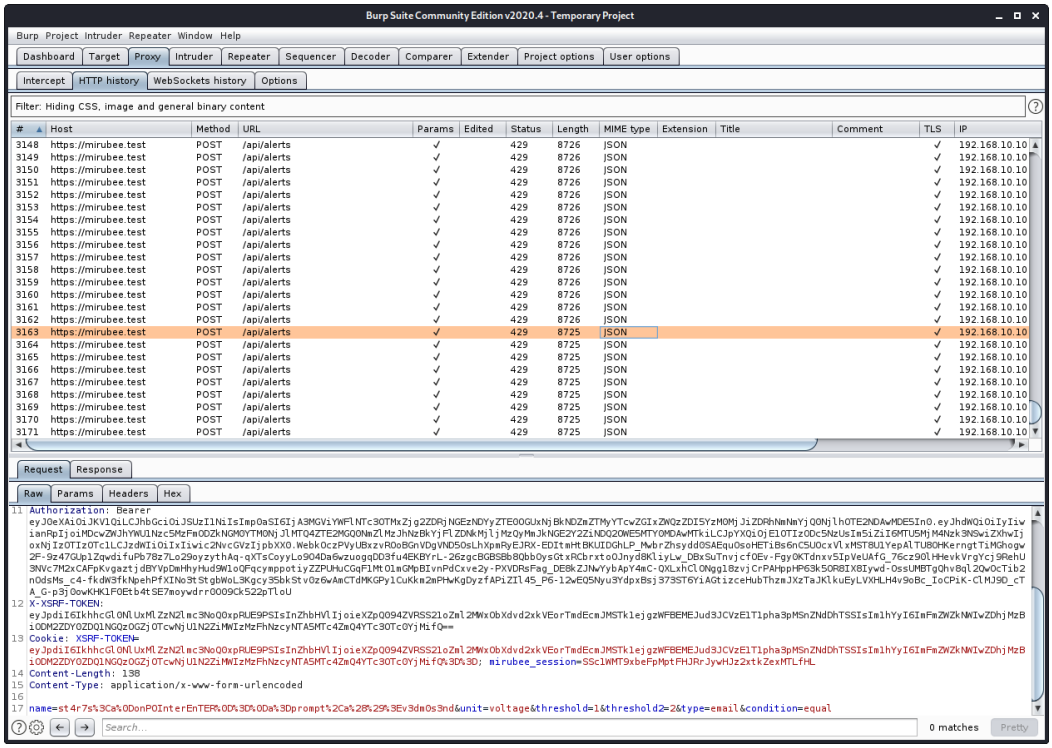


Figura 60: Pedidos gerados pelo XSStrike capturados pelo Burp Suite

em específico foi utilizado o endereço do *proxy* criado pela ferramenta Burp Suite para facilitar a análise dos pedidos enviados.

A análise efetuada pelo XSStrike a este *endpoint* revelou que o parâmetro «`location`» é susceptível a ser refletido, no entanto o XSStrike não foi capaz de criar nenhum *payload* útil e como tal não foi encontrada qualquer vulnerabilidade neste *endpoint*.

Através da Figura 60 é possível verificar todos os pedidos realizados pelo XSStrike, a partir do Burp Suite foi possível consultar não só os *endpoints* mas também os parâmetros testados e ainda os *payloads* gerados caso estes existam.

Na grande maioria dos *endpoints* testados o XSStrike não foi capaz de criar nenhum *payload* capaz de gerar ataques de Cross-Site Scripting (XSS) com a excepção do *endpoint* «`mirubee.test/api/alerts`» no qual, para o parâmetro «`name`», foram gerados 223 *payloads* válidos. A Figura 61 ilustra o começo e o final do teste de XSS realizado a este *endpoint*, é possível verificar que foram feitos 3072 pedidos ao servidor desta aplicação para testar os *payloads* gerados. No entanto, apesar dos 223 *payloads* válidos que foram gerados pelo XSStrike este não conseguiu encontrar nenhuma vulnerabilidade na aplicação testada.

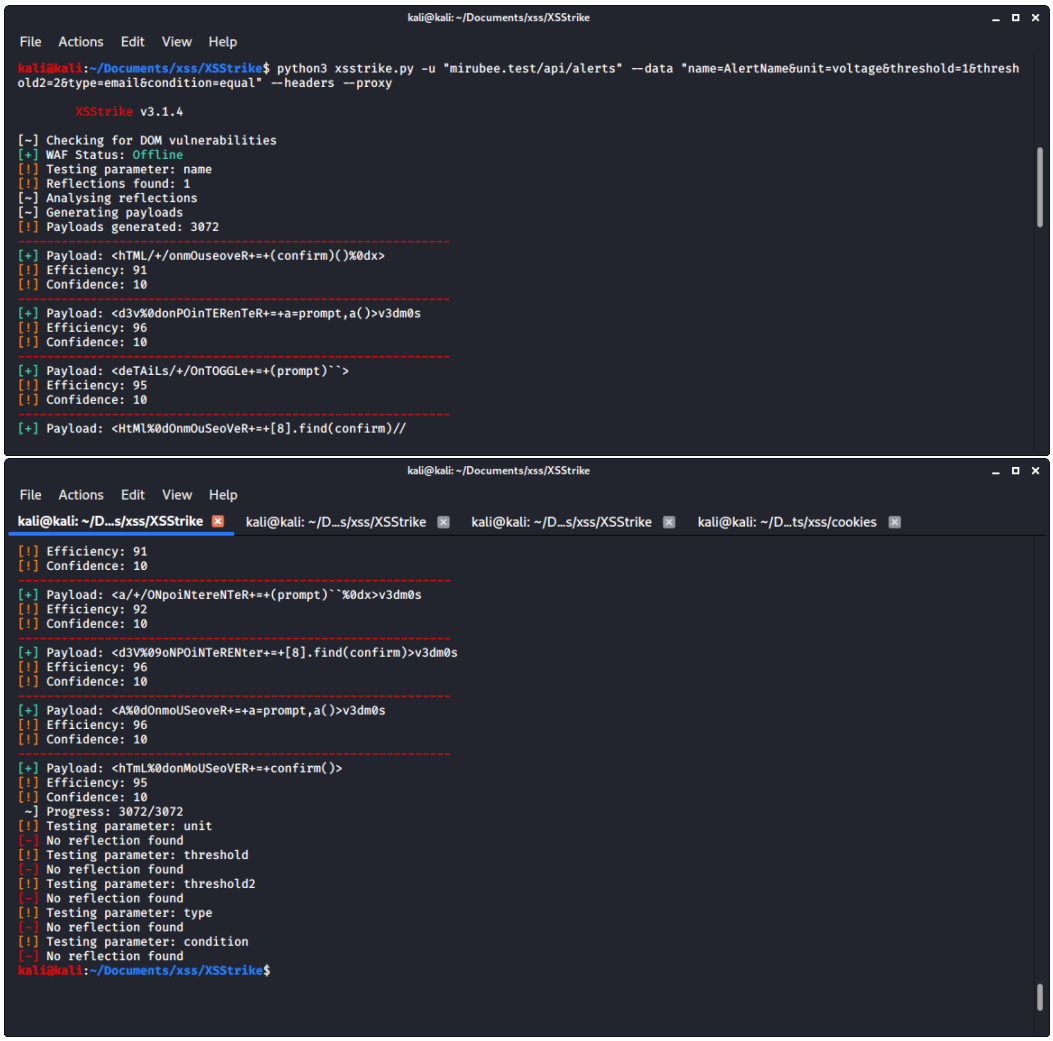


Figura 61: Teste de Cross-Site Scripting no *endpoint* «api/alerts»

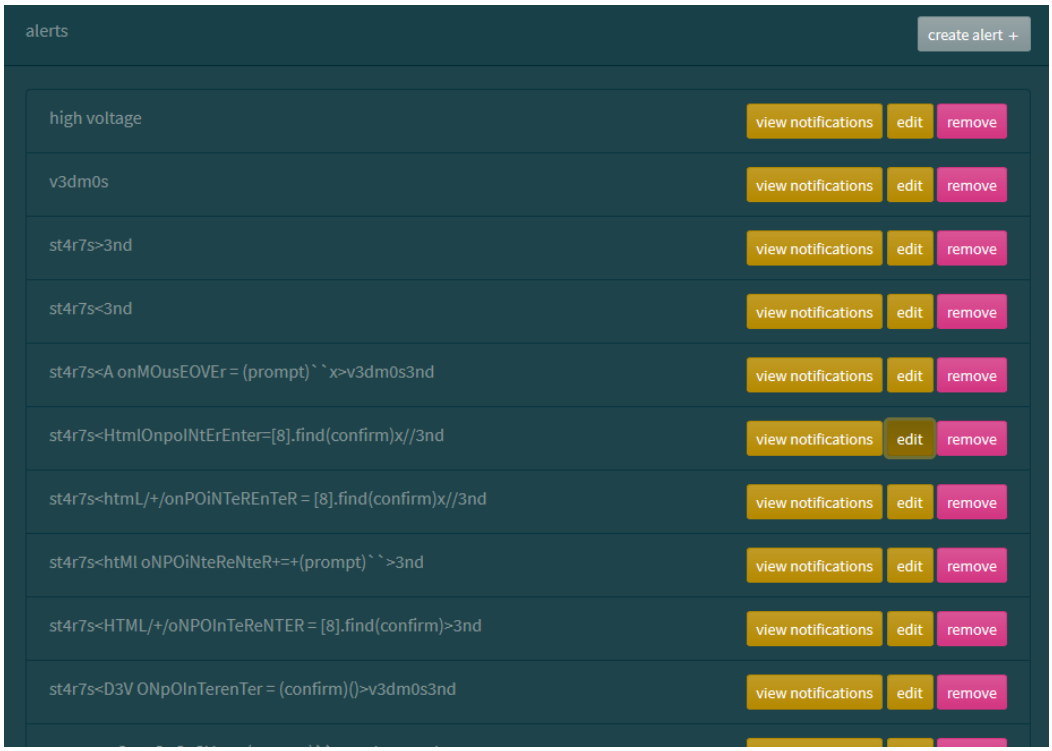


Figura 62: Página de alertas na aplicação web

É conveniente referir que, mesmo se os *payloads* gerados fossem passíveis de ser utilizados em ataques de XSS a *framework* Vue faz o *escape* automático por omissão das variáveis javascript incluídas no DOM das páginas desta aplicação, conseguindo evitar deste modo este tipo de ataques. Na Figura 63 é possível verificar os alertas gerados durante o teste do XSStrike, para além disto é possível observar que o carregamento desta página é feito sem qualquer efeito adverso causado pela execução indevida do código javascript injetado no nome dos alertas exibidos como seria de esperar.

Assim é possível chegar à conclusão que, apesar de em alguns casos, o servidor aceitar e armazenar as *strings* enviadas pelos clientes sem fazer o seu devido processamento, a aplicação cliente, ao receber estas *strings* possivelmente maliciosas, faz o seu *escape* e previne qualquer tipo ataque de XSS. Posto isto, com base nos resultados deste teste foi decidido realizar a «higienização» das *string* recebidas pelo servidor para evitar que este armazene *strings* potencialmente maliciosas na sua Base de Dados. Após terem sido implementadas as mudanças necessárias no servidor para «higienizar» as *strings* recebidas, foi repetido o teste presente na Figura 61 sendo que desta vez não foi gerado um único *payload* válido conforme ilustra a Figura 63.

Figura 63: Segundo teste de Cross-Site Scripting no *endpoint* «api/alerts»

9

CONCLUSÕES

O principal objetivo do trabalho realizado ao longo deste projeto passou pela análise de segurança realizada ao *smart meter* Mirubee e subsequente aplicação dos conhecimentos adquiridos ao longo deste mestrado para especificar e implementar uma solução que não só colmatasse as vulnerabilidades de segurança identificadas mas também fosse facilmente implementada pelo utilizador quer localmente (em sua casa) ou na Internet (*e. g.* utilizando serviços como a AWS ou o Digital Ocean).

Posto isto, tornou-se necessário realizar um trabalho de pesquisa com o intuito de melhor entender não só as principais alternativas ao Mirubee (e plataformas web associadas) mas também as principais evoluções assistidas ao longo dos últimos anos no ramo da IoT, mais especificamente nos dispositivos de monitorização inteligente (intitulados em inglês de *smart meters*) nos quais se enquadra o Mirubee.

Com base na análise de segurança realizada ao Mirubee, chegou-se à conclusão que não só o Mirubee mas também as plataformas web associadas a este (a Smilics e Circutor) apresentam comportamentos, funcionalidades e mecanismos de segurança que podem comprometer não só a privacidade dos dados tratados e transmitidos por estes mas também a integridade das leituras energéticas armazenadas pelas plataformas web. A maioria dos problemas de segurança encontrados durante esta análise podem ser atribuídos ao uso de mecanismo e funcionalidades de segurança considerados hoje em dia obsoletos e poderiam facilmente ser colmatados.

De seguida, foi fundamentado o ênfase dado à confidencialidade das leituras energéticas transmitidas por *smart meters* uma vez que com base nestas é possível derivar comportamentos e hábitos dos utilizadores destes dispositivos. Assim, tornar-se-ia possível que um agente externo mal intencionado, ao ter acesso a estes dados, deduzisse os hábitos de um utilizador inclusivamente inferir quando é que este se encontra ou não em casa.

Visto que não é viável ou mesmo possível alterar o comportamento original (não seguro) do Mirubee sem manipular o *firmware* deste, tornou-se necessário criar uma solução que contorna as vulnerabilidades deste. Mais especificamente, para contornar a transmissão não confidencial de dados do Mirubee foi traçada uma arquitetura que utiliza VLANs para isolar este processo independentemente da

localização do Mirubee em relação ao servidor no qual o utilizador decidiu hospedar a plataforma. De seguida foi realizado o levantamento das principais funcionalidades de segurança e de monitorização de consumos energéticos da solução desenvolvida.

Foram também documentadas e fundamentadas as escolhas tomadas durante o processo de desenvolvimento de cada um dos três principais componentes da plataforma de monitorização desenvolvida (O *proxy* Mirubee, o *back-end* e o *front-end* da plataforma).

Finalmente, foi considerado pertinente testar os mecanismos e funcionalidades de segurança implementados nesta plataforma num contexto prático e para tal foi realizada uma série de testes de penetração à plataforma. Foram testados os quatro tipos de vulnerabilidades encontradas em aplicações web consideradas mais relevantes no contexto deste trabalho:

1. Injeção SQL;
2. Quebra de autenticação;
3. Ataques de MITM;
4. Ataques de XSS.

No geral, como seria expectável, os testes realizados ajudaram a comprovar a resiliência desta plataforma a este tipo de ataque. Porém, através da ferramenta de automatização de testes de XSS (XSStrike) foi detetado que um dos *endpoints* desta plataforma aceitava pelo menos 223 *payloads* passíveis de causar este tipo de ataques. Deste modo, para averiguar o potencial impacto desta vulnerabilidade foi constatado que, apesar do *back-end* aceitar e armazenar *payloads* potencialmente maliciosos, o *front-end* ao receber do servidor estas *strings* realizava automaticamente o *escape* destas mitigando assim o potencial risco de segurança associado a esta vulnerabilidade. Apesar disto, com base nos resultados deste teste, foi alterado o comportamento do servidor para que este passasse a realizar a «higienização» dos *payloads* recebidos pelo utilizador; após esta alteração foi repetido o teste em questão e, com efeito, a ferramenta XSStrike passou a não ser capaz de gerar nenhum *payload* passível de ser utilizado como vetor de ataque num ataque XSS.

Com isto é possível concluir que, mesmo não sendo possível alterar diretamente os protocolos de comunicação utilizados pelo Mirubee foi criada uma solução que não só corrige as vulnerabilidades de segurança detetadas nas plataformas da Smilics e da Circutor mas também proporciona ao utilizador um maior nível de flexibilidade na recolha e processamento dos seus consumos energéticos.

9.1 TRABALHO FUTURO

Como trabalho futuro, seria interessante continuar o trabalho desenvolvido no *website*, possivelmente ao implementar novas ferramentas de visualização e análise de consumos. Também seria considerado oportuno a realização de mais testes de segurança à plataforma desenvolvida, não só em amplitude, testando mais tipos de vulnerabilidades, mas também ao testar em maior profundidade as vulnerabilidades já testadas. Seria também proveitoso fazer a engenharia reversa do *firmware* do Mirubee de forma a corrigir as vulnerabilidades de segurança identificadas.

Por último, é sugerido o desenvolvimento de uma aplicação móvel que interagisse com a plataforma desenvolvida neste projeto para permitir ao utilizador consultar os seus dados em qualquer altura e em qualquer lugar.

9.2 NOTIFICAÇÃO RESPONSÁVEL

Foi considerado oportuno notificar a Smilics das vulnerabilidades de segurança identificadas na sua plataforma e no *smart meter* Mirubee. Para tal, foi enviado um email para a Smilics com as principais vulnerabilidades de segurança encontradas, mais concretamente o uso de algoritmos de *hashing* de *passwords* obsoletos e o facto do Mirubee não utilizar o protocolo HTTPS.

Na resposta a este email foi referido que a equipa de desenvolvimento da Smilics está ciente destas falhas de segurança e que está atualmente a desenvolver uma nova aplicação com o objetivo de resolver o problema de segurança que resulta do processo atual de *hashing* das *passwords* introduzidas. Para além disto é referido que está a ser projetada uma nova arquitetura de *hardware* para o Mirubee que permitirá cifrar os dados enviados, sendo referido ainda que o *chipset* atual aparentemente não tem poder suficiente para realizar a cifragem de dados.

Tanto o email enviado inicialmente como a resposta da Smilics encontram-se na integra no Apêndice B deste documento.

BIBLIOGRAFIA

Asghar, Muhammad Rizwan et al. (2017). *Smart Meter Data Privacy: A Survey.*

Ashton, Kevin (2009). *That "Internet of Things" thing.*

Bernstein, Daniel (2006). *A state-of-the-art Diffie-Hellman function.* Website. https://cr.yp.to/ecdh.html.

Borgohain, Tuhin, Uday Kumar e Sugata Sanyal (2015). *Survey of Security and Privacy Issues of Internet of Things.*

Boylestad, R. L. (2015). *Introductory Circuit Analysis (13th Edition).* Pearson.

CENELEC (2001). *NP EN 50160: Características da tensão fornecida pelas redes de distribuição publica de energia elétrica.*

CEVE (2017). *Leitura de Contadores.* Website. https://www.ceve.pt/?id=19.

Circutor (2015a). *Sobre a Circutor.* Website. http://circutor.com/pt/empresa/informacao/sobre-a-circutor.

— (2015b). *WiBeee: Os consumos na palma da sua mão.* Website. http://circutor.com/pt/documentacao/artigos/3027-wibeee-os-consumos-na-palma-da-sua-mao.

Cuijpers, Colette e Bert-Jaap Koops (2012). *Smart Metering and Privacy in Europe: Lessons from the Dutch Case.*

EDP (2016). *EDP RE:DY.* Website. https://www.edp.pt/particulares/servicos/redy/.

— (2018). *Smart Meters EDP.* Website. https://www.edpdistribuicao.pt/en/networks-future/intelligent-networks/smart-meters.

— (2020a). *EDP Horarios Energéticos.* Website. https://www.edp.pt/particulares/apoio-cliente/opcao-horaria.aspx.

— (2020b). *EDP Tarifários.* Website. https://www.edp.pt/particulares/energia/tarifarios/.

Efergy (2018a). *Efergy E2 Classic.* Website. https://efergy.com/e2-classic/.

— (2018b). *Efergy E2 Classic.* Website. https://efergy.com/elite-classic/.

— (2018c). *Efergy Engage.* Website. https://efergy.com/engage/.

— (2018d). *Efergy Pro.* Website. https://efergy.com/efergypro/.

Efimarket (2019). *Efimarket - Produtos OWL.* Website. https://www.efimarket.pt/owl-eficiencia-energetica.

Erdemir, Ecenaz, Deniz Gündüz e Pier Dragotti (2020). *Privacy in Dynamical Systems*.

ERSE (2017). *Regulamento da Qualidade de Serviço*.

Gibson Research Corporation (out. de 2019). *SQRL Explained*. Website. URL: https://www.grc.com/sqrl/SQRL_Explained.pdf.

Grácio, Ângelo e Eduardo Faria (2018). *Análise da Aplicação Android Wibeee*.

Hardt, D. (out. de 2012). *The OAuth 2.0 Authorization Framework*. RFC 6749. Microsoft, pp. 1–76. URL: https://tools.ietf.org/html/rfc6749.

IoTAnalytics (2018). *State of the IoT 2018*. Website. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

Jain, Priyanka e Apoorv Gupta (2018). *Review of IoT Market and Open Source Technologies in IoT*.

Khan, Minhaj e Khaled Salah (2017). *IoT security: Review, blockchain solutions, and open challenges*.

Koponen, Pekka et al. (2017). *Definition of Smart Metering and Definition of Smart Metering and Applications andIdentification of Benefits*.

M'Raihi, D. et al. (dez. de 2005). *HOTP: An HMAC-Based One-Time Password Algorithm*. RFC 4226, pp. 1–37. URL: https://tools.ietf.org/html/rfc4226.

Madakam, Somayya e R. Ramaswamy anf Siddharth Tripathi (2015). *Internet of Things (IoT): A Literature Review*.

Muntjir, Mohd, Mohd Rahul e Hesham Alhumiany (2017). *An Analysis of Internet of Things(IoT): Novel Architectures, Modern Applications,Security Aspects and Future Scope with Latest Case Studies*.

Oborkhale, Lawrence e Oluwagbemiga Shoewu (2018). *Power Line Communication Technology*.

OpenEnergyMonitor (2017a). *Emoncms API docs*. Website. https://guide.openenergymonitor.org/technical/api/.

— (2017b). *OpenEnergyMonitor - About*. Website. https://openenergymonitor.org/?q=about.

— (2017c). *OpenEnergyMonitor - Shop*. Website. https://guide.openenergymonitor.org/technical/api/.

OWASP (2017). *The Ten Most Critical Web Application Security Risks*.

OWL (2015a). *OWL - About us*. Website. http://www.theowl.com/index.php/about-us/owl/.

— (2015b). *OWL Energy Monitors*. Website. http://www.theowl.com/index.php/products/energy-monitors/.

— (2015c). *OWL Smart Electricity Monitors*. Website. http://www.theowl.com/index.php/products/smart-electricity-monitors/.

Ribeiro, Gonçalo e Miguel Frade (set. de 2020). *goncalo-ribeiro/mirubee-mcif: v0.2*. Versão v0.2. DOI: 10.5281/zenodo.4015669. URL: https://doi.org/10.5281/zenodo.4015669.

SATEC (2014). *Electricity metering accuracy explained*. Website. https://www.ecdonline.com.au/content/electrical-distribution/article/electricity-metering-accuracy-explained-372339275.

Shelby, Z., K. Hartke e C. Bormann (jun. de 2014). *The Constrained Application Protocol (CoAP)*. RFC 7252. Universitaet Bremen TZI, pp. 1–112. URL: https://tools.ietf.org/html/rfc7252.

Smilics (2016). *Smilics - About us*. Website. https://wibeee.com/compania/.

Srinivas, Sampath et al. (abr. de 2017). *Universal 2nd Factor (U2F) Overview*. Website. URL: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html.

Xiaocong, Qian e Zhang Jidong (2010). *Study on the structure of "Internet of Things(IOT)" business operation support platform*.

APÊNDICES

A

OUTPUT DOS TESTES DE SEGURANÇA REALIZADOS

A.1 TESTES DE INJEÇÃO

Listagem 24: Teste de injeção SQL ao *endpoint* «`https://mirubee.test/api/login`»

```
kali@kali:~$ sqlmap -u "https://mirubee.test/api/login" --dbms MySql --force-ssl --data "email=username@email.com&password=password" -p "email"
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.4.6#stable}
|_ -| . [(]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 06:48:16 /2020-06-13/

[06:48:16] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('mirubee_session=Fl8O92ykDHr...HyUAajk2hb'). Do you want to use those [Y/n]
[06:48:17] [INFO] testing if the target URL content is stable
[06:48:18] [WARNING] target URL content is not stable (i.e. content differs). sqlmap will base the page comparison on a sequence matcher. If no dynamic nor injectable parameters are detected, or in case of junk results, refer to user's manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit]
[06:48:19] [INFO] searching for dynamic content
[06:48:19] [INFO] dynamic content marked for removal (1 region)
[06:48:20] [CRITICAL] not authorized, try to provide right HTTP authentication type and valid credentials (401)
[06:48:20] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times

[*] ending @ 06:48:20 /2020-06-13/
```

Listagem 25: Teste de injeção SQL ao *endpoint* «`https://mirubee.test/api/login`» com base num pedido HTTP interceptado

```
kali@kali:~$ sqlmap -r Documents/sql_injection/requests/https/login --dbms MySql
↪  --force-ssl -p "email"
        ___
       __H__
↪
 ___ ___["]_____ ___ ___  {1.4.6#stable}
↪
|_ -| . [.]     | .'| . |
↪
|___|_  [,]_|_|_|__,|  _|
↪
      |_|V...       |_|   http://sqlmap.org
↪

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
↪  consent is illegal. It is the end user's responsibility to obey all
↪  applicable local, state and federal laws. Developers assume no liability and
↪  are not responsible for any misuse or damage caused by this program

[*] starting @ 06:53:44 /2020-06-13/

[06:53:44] [INFO] parsing HTTP request from
↪  'Documents/sql_injection/requests/https/login'
JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[06:53:47] [INFO] testing connection to the target URL
[06:53:47] [INFO] testing if the target URL content is stable
you provided a HTTP Cookie header value, while target URL provides its own
↪  cookies within HTTP Set-Cookie header which intersect with yours. Do you
↪  want to merge them in further requests? [Y/n]
[06:53:51] [WARNING] target URL content is not stable (i.e. content differs).
↪  sqlmap will base the page comparison on a sequence matcher. If no dynamic
↪  nor injectable parameters are detected, or in case of junk results, refer to
↪  user's manual paragraph 'Page comparison'
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit]
[06:53:51] [INFO] searching for dynamic content
[06:53:51] [INFO] dynamic content marked for removal (1 region)
[06:53:52] [CRITICAL] target URL content appears to be heavily dynamic. sqlmap
↪  is going to retry the request(s)
[06:53:53] [INFO] testing for SQL injection on (custom) POST parameter 'JSON
↪  email'
[06:53:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:53:54] [CRITICAL] not authorized, try to provide right HTTP authentication
↪  type and valid credentials (401)
[06:53:54] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times

[*] ending @ 06:53:54 /2020-06-13/
```

Listagem 26: Teste de injeção SQL ao *endpoint* «`https://mirubee.test/api/users/email/username@email.com`»

```
kali@kali:~$ sqlmap -u "https://mirubee.test/api/users/email/username@email.com*" --dbms MySql --force-ssl --tamper=space2comment --random-agent --level 5 --risk 3 --headers="Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImNkM2YwMWUyNWRlMzdkYjk2YzgxNDUxNDE4ZGI3Njg0YWIxZDc5ODYxYjBmNmUwZWQ5YTM4YjMwY2JjMjI4NzA2YTA3MjdlMGFlZTMwZTM2In0.eyJhdWQiOiIyIiwianRpIjoiY2QzZjAxZTI1ZGUzN2RiOTZjODE0NTE0MThkYjc2ODRhYjFkNzk4NjFiMGY2ZTBlZDlhMzhiMzBjYmMyMjg3MDZhMDcyN2UwYWVlMzBlMzYiLCJpYXQiOjE1OTE5NTQ4MjMsIm5iZiI6MTU5MTk1NDgyMywiZXhwIjoxNjIzNDkwODIzLCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.bmE0hvke99JnRooFtA3g5SO33YPcrZC0snvjasKjW15QpB3ZR24BnxyEyX7ClNhBqYMRc6viOk9-6kRyxvI6ERCE1J8QMG4jLGXAOx8gfbFWRRwwvBWlZXHyUfU5daVDW8deuDgXplKvTiuk6QlOVQv4AlJYnDiMtl6TPF3aVCGZRp7l9Yoy4YIViRRB0GrdOZMawGA6bvKC0jqme14y-C_SsGgmxHYnpJjgc9ArVt844xKtxFUVPW_Z5SsjwxAEpEjoY6aLk20fEyRIr1odzJHYlk_0XUucpbHm_yFfjt3awYdFzFZjbEslbcZbjaMXTosZOpHK9h1xNclhx-jPemjsH29iEn_hip2tSk39M7mlB86lN_tWGw1ftI6yic8b5GuXPSYvuCp1ORabLkZMSRry9duyY0KBDlkT2m1c-DTRWd7VTVAkL0lxI-lSCeuYOERILX9n4LYpczEHjXXmxeOcHj_QF6Mvn08xPX09EQ5Di6TN9P3zvTbPebycCGhgJKWjkep2yqrE-4miebUShCanygL73cT_t9xZMrCTl5Zwce6fODYeEOeWYUmyVDm-vRyWHHb5K_Qj_5zim0SuWjCJTD9Z0vVBnXUQ32a1BFlUdcPEa2sO2HiPrYE5hCu1eqK4OkzyQhdlZa3BiFT9p_OR-gZKa3ebVD2zBsLpMLU"
        ___
       __H__
 ___ ___['']_____ ___ ___  {1.4.6#stable}
|_ -| . [,]     | .'| . |
|___|_  [()]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 06:28:27 /2020-06-15/

[06:28:27] [INFO] loading tamper module 'space2comment'
[06:28:27] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 5.1; pt-BR) AppleWebKit/523.15 (KHTML, like Gecko) Version/3.0 Safari/523.15' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
custom injection marker ('*') found in option '-u'. Do you want to process it? [Y/n/q] Y
[06:28:34] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('mirubee_session=LLemcV9XZhI...4bAyWwgCSY'). Do you want to use those [Y/n]
[06:28:37] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
[06:28:37] [INFO] testing if the target URL content is stable
[06:28:38] [INFO] target URL content is stable
[06:28:38] [INFO] testing if URI parameter '#1*' is dynamic
[06:28:44] [WARNING] URI parameter '#1*' does not appear to be dynamic
[06:28:48] [INFO] heuristic (basic) test shows that URI parameter '#1*' might be injectable (possible DBMS: 'IBM DB2')
```

```
[06:28:53] [INFO] testing for SQL injection on URI parameter '#1*'
[06:28:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:29:23] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[06:29:24] [WARNING] reflective value(s) found and filtering out
[06:29:46] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (NOT)'
[06:30:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (subquery - comment)'
[06:30:45] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (subquery - comment)'
[06:30:58] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (comment)'
[06:31:02] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (comment)'
[06:31:06] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT
↪  - comment)'
[06:31:11] [INFO] testing 'Boolean-based blind - Parameter replace (original
↪  value)'
[06:31:11] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL)'
[06:31:12] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL -
↪  original value)'
[06:31:12] [INFO] testing 'Boolean-based blind - Parameter replace (CASE)'
[06:31:13] [INFO] testing 'Boolean-based blind - Parameter replace (CASE -
↪  original value)'
[06:31:13] [INFO] testing 'HAVING boolean-based blind - WHERE, GROUP BY clause'
[06:31:32] [INFO] testing 'Generic inline queries'
[06:31:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (MySQL comment)'
[06:31:43] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (MySQL comment)'
[06:31:53] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT
↪  - MySQL comment)'
[06:32:03] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING,
↪  ORDER BY or GROUP BY clause'
[06:32:21] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (MAKE_SET)'
[06:32:41] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (MAKE_SET)'
[06:33:00] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (ELT)'
[06:33:22] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (ELT)'
[06:33:42] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (bool*int)'
[06:34:02] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (bool*int)'
[06:34:22] [INFO] testing 'MySQL boolean-based blind - Parameter replace
↪  (MAKE_SET)'
[06:34:24] [INFO] testing 'MySQL boolean-based blind - Parameter replace
↪  (MAKE_SET - original value)'
[06:34:26] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT)'
[06:34:28] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT -
↪  original value)'
```

```
55 [06:34:30] [INFO] testing 'MySQL boolean-based blind - Parameter replace
   ↪  (bool*int)'
56 [06:34:32] [INFO] testing 'MySQL boolean-based blind - Parameter replace
   ↪  (bool*int - original value)'
57 [06:34:34] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪  clause'
58 [06:34:35] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪  clause (original value)'
59 [06:34:36] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪  clause'
60 [06:34:36] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪  clause (original value)'
61 [06:34:36] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Stacked queries'
62 [06:34:49] [INFO] testing 'MySQL < 5.0 boolean-based blind - Stacked queries'
63 [06:34:49] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (BIGINT UNSIGNED)'
64 [06:35:01] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
   ↪  (BIGINT UNSIGNED)'
65 [06:35:13] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (EXP)'
66 [06:35:25] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
   ↪  (EXP)'
67 [06:35:37] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (JSON_KEYS)'
68 [06:35:49] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING
   ↪  clause (JSON_KEYS)'
69 [06:36:01] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (FLOOR)'
70 [06:36:13] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY
   ↪  or GROUP BY clause (FLOOR)'
71 [06:36:24] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (EXTRACTVALUE)'
72 [06:36:38] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
   ↪  or GROUP BY clause (EXTRACTVALUE)'
73 [06:36:50] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (UPDATEXML)'
74 [06:37:02] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
   ↪  or GROUP BY clause (UPDATEXML)'
75 [06:37:14] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER
   ↪  BY or GROUP BY clause (FLOOR)'
76 [06:37:27] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause
   ↪  (FLOOR)'
77 [06:37:40] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
78 [06:37:46] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE
   ↪  (EXTRACTVALUE)'
79 [06:37:54] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT
   ↪  UNSIGNED)'
80 [06:37:55] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
81 [06:37:55] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace
   ↪  (JSON_KEYS)'
82 [06:37:55] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
83 [06:37:55] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
   ↪  (UPDATEXML)'
```

```
84  [06:37:55] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
    ↪  (EXTRACTVALUE)'
85  [06:37:56] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause
    ↪  (BIGINT UNSIGNED)'
86  [06:37:56] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause
    ↪  (EXP)'
87  [06:37:57] [INFO] testing 'MySQL >= 5.7.8 error-based - ORDER BY, GROUP BY
    ↪  clause (JSON_KEYS)'
88  [06:37:57] [INFO] testing 'MySQL >= 5.0 error-based - ORDER BY, GROUP BY clause
    ↪  (FLOOR)'
89  [06:37:57] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause
    ↪  (EXTRACTVALUE)'
90  [06:37:58] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause
    ↪  (UPDATEXML)'
91  [06:37:59] [INFO] testing 'MySQL >= 4.1 error-based - ORDER BY, GROUP BY clause
    ↪  (FLOOR)'
92  [06:37:59] [INFO] testing 'MySQL inline queries'
93  [06:37:59] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
94  [06:38:05] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
95  [06:38:14] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP -
    ↪  comment)'
96  [06:38:19] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
97  [06:38:28] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query -
    ↪  comment)'
98  [06:38:34] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
99  [06:38:43] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
100 [06:38:56] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
101 [06:39:07] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
102 [06:39:19] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP)'
103 [06:39:30] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP -
    ↪  comment)'
104 [06:39:38] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP - comment)'
105 [06:39:46] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP -
    ↪  comment)'
106 [06:39:53] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP -
    ↪  comment)'
107 [06:40:00] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'
108 [06:40:12] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query)'
109 [06:40:23] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query -
    ↪  comment)'
110 [06:40:32] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query -
    ↪  comment)'
111 [06:40:40] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
112 [06:40:51] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (comment)'
113 [06:40:59] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
114 [06:41:11] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP -
    ↪  comment)'
115 [06:41:18] [INFO] testing 'MySQL AND time-based blind (ELT)'
116 [06:41:30] [INFO] testing 'MySQL OR time-based blind (ELT)'
117 [06:41:41] [INFO] testing 'MySQL AND time-based blind (ELT - comment)'
118 [06:41:49] [INFO] testing 'MySQL OR time-based blind (ELT - comment)'
119 [06:41:56] [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query) -
    ↪  PROCEDURE ANALYSE (EXTRACTVALUE)'
```

```
120 [06:42:04] [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query - comment)
    ↪  - PROCEDURE ANALYSE (EXTRACTVALUE)'
121 [06:42:08] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace'
122 [06:42:09] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace
    ↪  (substraction)'
123 [06:42:09] [INFO] testing 'MySQL < 5.0.12 time-based blind - Parameter replace
    ↪  (heavy queries)'
124 [06:42:09] [INFO] testing 'MySQL time-based blind - Parameter replace (bool)'
125 [06:42:09] [INFO] testing 'MySQL time-based blind - Parameter replace (ELT)'
126 [06:42:11] [INFO] testing 'MySQL time-based blind - Parameter replace (MAKE_SET)'
127 [06:42:13] [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY
    ↪  clause'
128 [06:42:13] [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY
    ↪  clause (heavy query)'
129 it is recommended to perform only basic UNION tests if there is not at least one
    ↪  other (potential) technique found. Do you want to reduce the number of
    ↪  requests? [Y/n] Y
130 [06:45:10] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
131 [06:45:20] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
132 [06:45:30] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
133 [06:45:41] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
134 [06:45:51] [WARNING] URI parameter '#1*' does not seem to be injectable
135 [06:45:51] [CRITICAL] all tested parameters do not appear to be injectable. As
    ↪  heuristic test turned out positive you are strongly advised to continue on
    ↪  with the tests
136 [06:45:51] [WARNING] HTTP error codes detected during run:
137 500 (Internal Server Error) - 21 times, 404 (Not Found) - 3246 times
138
139 [*] ending @ 06:45:51 /2020-06-15/
```

Listagem 27: Teste de injeção SQL ao *endpoint* «`https://mirubee.test/api/sites/1/devices`»

```
kali@kali:~/Documents/sql_injection/requests$ sqlmap -u
↪  "https://mirubee.test/api/sites/1*/devices/" --dbms MySql --force-ssl
↪  --headers="Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI
↪  6ImNkM2YwMWUyNWRlMzdkYjk2YzgxNDUxNDE4ZGI3Njg0YWIxZDc5ODYxYjBmNmUwZWQ5YTM4YjM
↪  wY2JjMjI4NzA2YTA3MjdlMGFlZTMwZTM2In0.eyJhdWQiOiIyIiwianRpIjoiY2QzZjAxZTI1ZGU
↪  zN2RiOTZjODE0NTE0MThkYjc2ODRhYjFkNzk4NjFiMGY2ZTBlZDlhMzhiMzBjYmMyMjg3MDZhMDc
↪  yN2UwYWVlMzBlMzYiLCJpYXQiOjE1OTE5NTQ4MjMsIm5iZiI6MTU5MTk1NDgyMywiZXhwIjoxNjI
↪  zNDkwODIzLCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.bmE0hvke99JnRooFtA3g5SO33YPcrZC0snv
↪  jasKjW15QpB3ZR24BnxyEyX7ClNhBqYMRc6viOk9-6kRyxvI6ERCE1J8QMG4jLGXAOx8gfbFWRRw
↪  wvBWlZXHyUfU5daVDW8deuDgXplKvTiuk6QlOVQv4AlJYnDiMtl6TPF3aVCGZRp7l9Yoy4YIViRR
↪  B0GrdOZMawGA6bvKC0jqme14y-C_SsGgmxHYnpJjgc9ArVt844xKtxFUVPW_Z5SsjwxAEpEjoY6a
↪  Lk20fEyRIr1odzJHYlk_0XUucpbHm_yFfjt3awYdFzFZjbEslbcZbjaMXTosZOpHK9h1xNclhx-j
↪  PemjsH29iEn_hip2tSk39M7mlB86lN_tWGw1ftI6yic8b5GuXPSYvuCp1ORabLkZMSRry9duyY0K
↪  BDlkT2m1c-DTRWd7VTVAkL0lxI-lSCeuYOERILX9n4LYpczEHjXXmxeOcHj_QF6Mvn08xPX09EQ5
↪  Di6TN9P3zvTbPebycCGhgJKWjkep2yqrE-4miebUShCanygL73cT_t9xZMrCTl5Zwce6fODYeEOe
↪  WYUmyVDm-vRyWHHb5K_Qj_5zim0SuWjCJTD9Z0vVBnXUQ32a1BFlUdcPEa2sO2HiPrYE5hCu1eqK
↪  4OkzyQhdlZa3BiFT9p_OR-gZKa3ebVD2zBsLpMLU" --level 5 --risk
↪  3
        ___
       __H__
↪
 ___ ___["]_____ ___ ___  {1.4.6#stable}
↪
|_ -| . ['']     | .'| . |
↪
|___|_  [)]_|_|_|__,|  _|
↪
      |_|V...       |_|   http://sqlmap.org
↪

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
↪  consent is illegal. It is the end user's responsibility to obey all
↪  applicable local, state and federal laws. Developers assume no liability and
↪  are not responsible for any misuse or damage caused by this program

[*] starting @ 06:28:44 /2020-06-13/

custom injection marker ('*') found in option '-u'. Do you want to process it?
↪  [Y/n/q] Y
[06:28:46] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own
↪  ('mirubee_session=ugqB3sUatT2...LrkrPL4VtA'). Do you want to use those [Y/n]
[06:28:55] [INFO] checking if the target is protected by some kind of WAF/IPS
[06:29:00] [CRITICAL] heuristics detected that the target is protected by some
↪  kind of WAF/IPS
are you sure that you want to continue with further target testing? [Y/n] Y
[06:29:05] [WARNING] please consider usage of tamper scripts (option '--tamper')
[06:29:05] [INFO] testing if the target URL content is stable
[06:29:05] [INFO] target URL content is stable
[06:29:05] [INFO] testing if URI parameter '#1*' is dynamic
```

```
[06:29:10] [WARNING] URI parameter '#1*' does not appear to be dynamic
[06:29:10] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might
↪  not be injectable
[06:29:10] [INFO] testing for SQL injection on URI parameter '#1*'
[06:29:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:29:26] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[06:29:26] [WARNING] reflective value(s) found and filtering out
[06:30:14] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[06:30:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (subquery - comment)'
[06:30:51] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (subquery - comment)'
[06:31:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (comment)'
[06:31:54] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (comment)'
[06:32:35] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT
↪  - comment)'
[06:32:47] [INFO] testing 'Boolean-based blind - Parameter replace (original
↪  value)'
[06:32:49] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL)'
[06:32:51] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL -
↪  original value)'
[06:32:51] [INFO] testing 'Boolean-based blind - Parameter replace (CASE)'
[06:32:54] [INFO] testing 'Boolean-based blind - Parameter replace (CASE -
↪  original value)'
[06:32:54] [INFO] testing 'HAVING boolean-based blind - WHERE, GROUP BY clause'
[06:33:04] [INFO] testing 'Generic inline queries'
[06:33:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause
↪  (MySQL comment)'
[06:33:13] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause
↪  (MySQL comment)'
[06:33:24] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT
↪  - MySQL comment)'
[06:33:35] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING,
↪  ORDER BY or GROUP BY clause'
[06:33:52] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (MAKE_SET)'
[06:34:03] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (MAKE_SET)'
[06:34:29] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (ELT)'
[06:34:49] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (ELT)'
[06:36:06] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (bool*int)'
[06:36:17] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER
↪  BY or GROUP BY clause (bool*int)'
[06:36:43] [INFO] testing 'MySQL boolean-based blind - Parameter replace
↪  (MAKE_SET)'
[06:36:44] [INFO] testing 'MySQL boolean-based blind - Parameter replace
↪  (MAKE_SET - original value)'
[06:36:44] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT)'
```

```
55 [06:36:44] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT -
   ↪ original value)'
56 [06:36:44] [INFO] testing 'MySQL boolean-based blind - Parameter replace
   ↪ (bool*int)'
57 [06:36:45] [INFO] testing 'MySQL boolean-based blind - Parameter replace
   ↪ (bool*int - original value)'
58 [06:36:45] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪ clause'
59 [06:36:46] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪ clause (original value)'
60 [06:36:46] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪ clause'
61 [06:36:46] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY
   ↪ clause (original value)'
62 [06:36:46] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Stacked queries'
63 [06:36:54] [INFO] testing 'MySQL < 5.0 boolean-based blind - Stacked queries'
64 [06:36:54] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (BIGINT UNSIGNED)'
65 [06:37:10] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
   ↪ (BIGINT UNSIGNED)'
66 [06:37:28] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (EXP)'
67 [06:37:45] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause
   ↪ (EXP)'
68 [06:38:03] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (JSON_KEYS)'
69 [06:38:21] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING
   ↪ clause (JSON_KEYS)'
70 [06:38:39] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (FLOOR)'
71 [06:38:56] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY
   ↪ or GROUP BY clause (FLOOR)'
72 [06:39:13] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (EXTRACTVALUE)'
73 [06:39:29] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
   ↪ or GROUP BY clause (EXTRACTVALUE)'
74 [06:39:44] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (UPDATEXML)'
75 [06:40:01] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY
   ↪ or GROUP BY clause (UPDATEXML)'
76 [06:40:16] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER
   ↪ BY or GROUP BY clause (FLOOR)'
77 [06:40:31] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause
   ↪ (FLOOR)'
78 [06:40:47] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
79 [06:41:12] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE
   ↪ (EXTRACTVALUE)'
80 [06:41:23] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT
   ↪ UNSIGNED)'
81 [06:41:23] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
82 [06:41:24] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace
   ↪ (JSON_KEYS)'
83 [06:41:24] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
```

```
[06:41:24] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
↪ (UPDATEXML)'
[06:41:25] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace
↪ (EXTRACTVALUE)'
[06:41:25] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause
↪ (BIGINT UNSIGNED)'
[06:41:25] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause
↪ (EXP)'
[06:41:26] [INFO] testing 'MySQL >= 5.7.8 error-based - ORDER BY, GROUP BY
↪ clause (JSON_KEYS)'
[06:41:27] [INFO] testing 'MySQL >= 5.0 error-based - ORDER BY, GROUP BY clause
↪ (FLOOR)'
[06:41:27] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause
↪ (EXTRACTVALUE)'
[06:41:28] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause
↪ (UPDATEXML)'
[06:41:28] [INFO] testing 'MySQL >= 4.1 error-based - ORDER BY, GROUP BY clause
↪ (FLOOR)'
[06:41:29] [INFO] testing 'MySQL inline queries'
[06:41:29] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[06:41:36] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[06:41:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP -
↪ comment)'
[06:41:55] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[06:42:05] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query -
↪ comment)'
[06:42:13] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[06:42:24] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[06:42:39] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP)'
[06:42:54] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
[06:43:09] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP)'
[06:43:23] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP -
↪ comment)'
[06:43:32] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP - comment)'
[06:43:41] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP -
↪ comment)'
[06:43:51] [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP -
↪ comment)'
[06:44:00] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'
[06:44:16] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query)'
[06:44:30] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query -
↪ comment)'
[06:44:39] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query -
↪ comment)'
[06:44:50] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
[06:45:04] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (comment)'
[06:45:13] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
[06:45:27] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP -
↪ comment)'
[06:45:36] [INFO] testing 'MySQL AND time-based blind (ELT)'
[06:45:51] [INFO] testing 'MySQL OR time-based blind (ELT)'
[06:46:05] [INFO] testing 'MySQL AND time-based blind (ELT - comment)'
[06:46:15] [INFO] testing 'MySQL OR time-based blind (ELT - comment)'
```

```
120 [06:46:25] [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query) -
    ↪  PROCEDURE ANALYSE (EXTRACTVALUE)'
121 [06:46:35] [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query - comment)
    ↪  - PROCEDURE ANALYSE (EXTRACTVALUE)'
122 [06:46:41] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace'
123 [06:46:42] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace
    ↪  (substraction)'
124 [06:46:42] [INFO] testing 'MySQL < 5.0.12 time-based blind - Parameter replace
    ↪  (heavy queries)'
125 [06:46:42] [INFO] testing 'MySQL time-based blind - Parameter replace (bool)'
126 [06:46:43] [INFO] testing 'MySQL time-based blind - Parameter replace (ELT)'
127 [06:46:43] [INFO] testing 'MySQL time-based blind - Parameter replace (MAKE_SET)'
128 [06:46:43] [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY
    ↪  clause'
129 [06:46:44] [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY
    ↪  clause (heavy query)'
130 it is recommended to perform only basic UNION tests if there is not at least one
    ↪  other (potential) technique found. Do you want to reduce the number of
    ↪  requests? [Y/n]
131 [06:46:51] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
132 [06:47:08] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
133 [06:47:17] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
134 [06:47:26] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
135 [06:47:35] [WARNING] URI parameter '#1*' does not seem to be injectable
136 [06:47:35] [CRITICAL] all tested parameters do not appear to be injectable. If
    ↪  you suspect that there is some kind of protection mechanism involved (e.g.
    ↪  WAF) maybe you could try to use option '--tamper' (e.g.
    ↪  '--tamper=space2comment') and/or switch '--random-agent'
137 [06:47:35] [WARNING] HTTP error codes detected during run:
138 405 (Method Not Allowed) - 1 times, 500 (Internal Server Error) - 214 times, 429
    ↪  (Too Many Requests) - 1916 times
139
140 [*] ending @ 06:47:35 /2020-06-13/
```

A.2 QUEBRA DE AUTENTICAÇÃO

Listagem 28: *Output* do pedido, e respetiva resposta, classificados como «`Hit`» pelo Patator

```
1  POST /api/login HTTP/1.1
2  Host: mirubee.test
3  User-Agent: Mozilla/5.0
4  Accept: */*
5  Content-Length: 44
6  Content-Type: application/x-www-form-urlencoded
7
8  email=username%40email.com&password=password
9
10 HTTP/1.1 200 OK
11 Server: nginx/1.15.8
12 Content-Type: application/json
13 Transfer-Encoding: chunked
14 Connection: keep-alive
15 Cache-Control: no-cache, private
16 Date: Tue, 16 Jun 2020 01:25:08 GMT
17 X-RateLimit-Limit: 60
18 X-RateLimit-Remaining: 44
19 Set-Cookie: mirubee_session=QHB6GjtKfmKVwaVbjTx8lyxax50xtuO5e2hltgXG;
   ↪ expires=Tue, 16-Jun-2020 03:25:08 GMT; Max-Age=7200; path=/; httponly
20
21 743
22 {"token_type":"Bearer","expires_in":31536000,"access_token":"eyJ0eXAiOiJKV1QiLCJ⌋
   ↪ hbGciOiJSUzI1NiIsImp0aSI6IjNjODYwNGZlMDQzODg0ODAzMzE2YmM4Y2I5MGJkZWZjNDk5Yzl⌋
   ↪ hODcwZWNlOGM0YjIxNzliZTJlMDE3ZmZhOGE3NWRlZmE1NTc2ODMwMjQ1In0.eyJhdWQiOiIyIiw⌋
   ↪ ianRpIjoiM2M4NjA0ZmUwNDM4ODQ4MDMzMTZiYzhjYjkwYmRlZmM0OTljOWE4NzBlY2U4YzRiMjE⌋
   ↪ 3OWJlMmUwMTdmZmE4YTc1ZGVmYTU1NzY4MzAyNDUiLCJpYXQiOjE1OTIyNzA3MDgsIm5iZiI6MTU⌋
   ↪ 5MjI3MDcwOCwiZXhwIjoxNjIzODA2NzA4LCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.GHy7mqbhI0t⌋
   ↪ _815hkUH_6zdtFHnVP25bxgVoARkVplFNmjrfRFIAiltNaX-8CbF5G8R5z_t2nE_sr2Jut6r5NCc⌋
   ↪ NYHLHOk9A5_qTkSn-v82SkUD46IQ6IA5eaU4iS9D85P1cOu_saP0SrYlacnbeQE4Wy8-3YlgCHXP⌋
   ↪ rbfiIbO3a7LrJNiekC4XmVEZnpNRF-UdxTE7wKdspMCkv7e66y_q4UWKRDD7P383m0Uw8CmAK5xC⌋
   ↪ RTBDbbJ-5G86pLRGXlfoomaH1OhF2BVQp9DrC4aWKNydvKDoP8ulyWfXB7tlSqE4enWuefh2Hz0w⌋
   ↪ KSPMLHXYpIINYNWZegYna46G-CAYnzGkrWUtufxtHDDSQk46W7HL4cGLH1sKHu08QIATzf-0LbFS⌋
   ↪ zY7Dw-nIyZdKJJO5cEs3ZubJr5xZwuHZ_MuWtbX4t0g5dUtEK6p4LjfQdbFAYc508iGV5ceXRjNG⌋
   ↪ pO6S_3-Sp4ySnVNZXOxTDaXILyY00i3b-BOAdzjE6_svZ1fKoJ2cuROHY_cjlyoqBx13JHKNiYr_⌋
   ↪ UuFueWD8ILhM7LdP7ArNI5vPKV9oZ90bojf5NsZUa10Mu3e-h__woeDdzoCP7aT-D0vnV_Xkfgjf⌋
   ↪ URqBWjZJXC0K4J7bMAQnsSwEH75RRGs_Vb_gs2Qzi9nkbrDPbac86fa79O60xpT0","refresh_t⌋
   ↪ oken":"def50200aac4b5e765a5f0e5edc6c54254397f2cf0e910f9069fca2b53db1fc42d448⌋
   ↪ 685dddaa052ab43343d03de485d40f419bfe1aa2a3f0a2fb09844795fd6ab6aed293f1a14109⌋
   ↪ 8171da27d91c5c3d9d1519460853c1b297886e0cb606ea1302c9fc732900cca243c755ef38cd⌋
   ↪ 5c324730a6a95690a58ed752285f4f4f837922cf67fa76a2158b9943ed4b8c9f32d1c2cfaa6c⌋
   ↪ 7377155977a8634b20248b56f0ecb341723d987b9eb420424043bf50fdf92c1ebbd84b74f4a7⌋
   ↪ 09cdf7d11324d70f17a8555ed482e524a9770ad60660ec1de9edf071e2dbc31f6df8bbf89f6f⌋
   ↪ 4fdcdb7bd2184c499123a8234c4ba336c87692ee366062581f6affdfc07a4b9a0fee0149fde1⌋
   ↪ 37612a6c3f1218d8526f25196955463a616b60203353c95710b38008b5c68ff5bf5a0c354d39⌋
   ↪ e34a683c788ab85983134257a185db927924d2680f3c13de79b17e5368e2cbc44bcbc61c4386⌋
   ↪ 89d8dbf11e1247f47650a39a1718c"}
23 0
```

A.3 ATAQUES DE MAN-IN-THE-MIDDLE

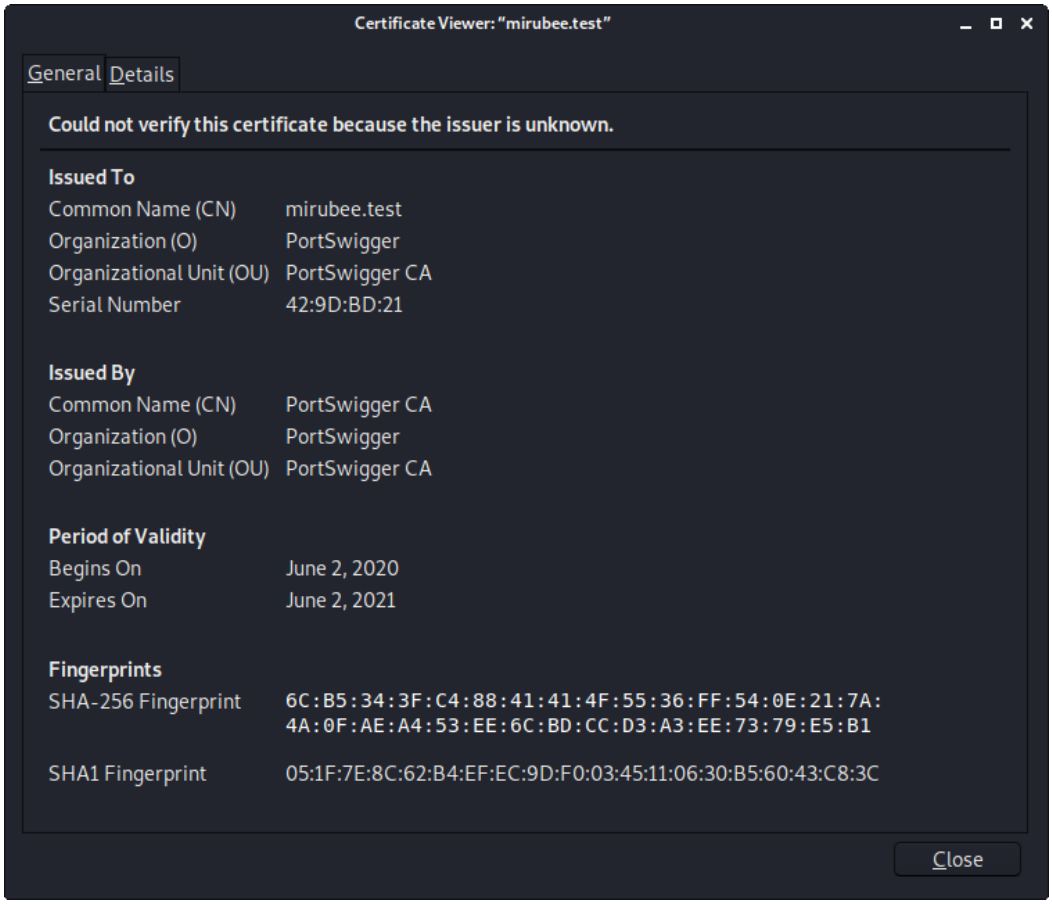


Figura 64: Certificado empregue pelo Burp Suite

Listagem 29: Resposta do servidor ao pedido de emissão do *challenge* de autenticação

```
HITP/1.1 200 OK
Server: nginx/1.15.8
Content-Type: application/json
Connection: close
Cache-Control: no-cache, private
Date: Tue, 16 Jun 2020 13:48:07 GMT
X-RateLimit-Limit: 60
X-RateLimit -Remaining: 58
Set-Cookie: mirubee_session=FlSv5LtewD3dY7yLhraWQnEwkhNiqCY1bsSKycCr;
↪  expires=Tue, 16-Jun-2020 15:48:07 GMT; Max-Age=7200; pathe/; httponly
Content-Length: 298

{
    "publickey":{
        "timeout ": 20000,
        "challenge" : "?BINARY?B?x6CH8JbVy8ej2VfsEq3eyWOWbbPmzThp7PtCVbaTx50=?=",
        "allowCredentials":[
            {
                "id" :"?BINARY?B?JU1EKzjC6k6+LcpjLmMth82zkalXy3vs79xM86KRsfW5Hh\⌋
↪  /7SGwnVAU2EJzi8r45GLjxjrjoY93sSuZSLsjJRg==?=",
                "transports":[
                    "usb",
                    "nfct,
                    "ble",
                    "internal"
                ],
                "type": "public-key"
            }
        ]
    }
}
```

Listagem 30: Pedido de autenticação do cliente com a assinatura gerada pelo dispositivo U2F

```
POST /api/u2f/processGet HTTP/1.1
Host: mirubee.test
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://mirubee.test/
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6IjYxN2U3YmVhZW
↪ RlMzE1YjNlNDcyMzE3MmY0MzcxMzA1OWNmN2NiM2Q0NWZkOThlMWVkOTkyMjk3ODY4N2IwN2NhZD
↪ c4ZDU3MjUyYWNmOWI3In0.eyJhdWQiOiIyIiwianRpIjoiNjE3ZTdiZWFlZGUzMTViM2U0NzIzMT
↪ cyZjQzNzEzMDU5Y2Y3Y2IzZDQ1ZmQ5OGUxZWQ5OTIyOTc4Njg3YjA3Y2FkNzhkNTcyNTJhY2Y5Yj
↪ ciLCJpYXQiOjE1OTIzMTUyODIsIm5iZiI6MTU5MjMxNTI4MiwiZXhwIjoxNjIzODUxMjgyLCJzdW
↪ IiOiIxIiwic2NvcGVzIjpbXX0.pOP1KDemxKbaR09J8L15y8Rjo8a_l339z6LiQOO8CRTBq2aX4O
↪ APtiLRNUzATsDeAjTY2dX8Ca_T9UiXwX2_DsAg_keS1uWZGN7TU3CUjje5hKRWLdqkpFRXMpbvnZ
↪ hLIaV5kFA20GvclItthbbUZ3W9MkwSW6Kt9GkPPFqPSVuZygdawKAPeXHqzpTDAZXtZhZosN1HVV
↪ cNMz3yaa5uLoYlXFGwbLlC7YNu1jWYDqQy41XkZJesA6fe2Ke0WJuOnVqyGBYVwW2TZc3hr7mRb2
↪ ttdnU3_iluP7l825H3qxQ5mcE_ZoX7bDdLdS5q4YKdTEUyAdc8N8rj9vaBMQN0w-mI0IfdLOVk_L
↪ DYM4ZfhP_8VVbPHQ31BlogoZl4wJPFUXZTEuq9akwbE7mplvSIWumoOWaFZB1rb8zQcBYwvZJzN0
↪ ArvPs4NMDpmIgbOKv9XPtfRIwucdAWOBAFB4AEP8GIvgwrtG-0vvr7W6kfR3a7FCzkBTi1kxcNY2
↪ xEmnAd_kVXyo0DdS5yW9UPPQrerCFFmmLrwH7oPOZ-jBQ3eQnyQriiRL2xbvos-De6bGMzKzAucN
↪ ErgkQpBubZoHD7WmIaOziyRpON59SY6XeH7J7B7RbYAWK2n4fbfnGuxL0WI4gHg0FYFvpclieyil
↪ pfzmcxIpnc8_UuXIAVzXRR-WU
Origin: https://mirubee.test
Content-Length: 536
Connection: close
Cookie: XSRF-TOKEN=eyJpdiI6IjMwdngzNWtYb2xXRHkxanBYaVNOdXc9PSIsInZhbHVlIjoidk5SM
↪ kxLK3JKWWJPbml0SXZUdU45UnJodUZUcmc4eFwvV09iazhmZFdNdlV2TVhoTzdZRjFURnJrQmRqR
↪ DBJT1MiLCJtYWMiOiI5ZjgyMjQ0OGFhZjI5MjI4ZjU0ZGZhMzg5NDU4ZTA0NWEzZWRhMGQ1OTllZ
↪ TVlM2E2MGY4NjNiY2NiOWFhODFjIn0%3D;
↪ mirubee_session=F1Sv5LtewD3dY7yLhraWQnEwkhNiqCY1bsSKycCr

{"id":"JU1EKzjC6k6+LcpjLmMth82zka1Xy3vs79xM86KRsfW5Hh/7SGwmVAU2EJzi8r45GLjxjrjoY
↪ 93sSuZSLsjJRg==","clientDataJSON":"eyJjaGFsbGVuZ2UiOiJ4NkNIOEpiVnk4ZWoyVmZzR
↪ XEzZXlXT1diYlBtelRocDdQdENWYmFUeDUwIiwiY2xpZW50RXh0ZW5zaW9ucyI6e30sImhhc2hBb
↪ Gdvcml0aG0iOiJTSEEtMjU2Iiwib3JpZ2luIjoiaHR0cHM6Ly9taXJ1YmVlLnRlc3QiLCJ0eXBlI
↪ joid2ViYXV0aG4uZ2V0In0=","authenticatorData":"+mWM6mkpH1HV06DG7lgc8HQUeGPRjz
↪ HXicoo2faJWWsBAAAAvA==","signature":"MEUCIQDKEiyHuqGyqCcQVfNdQbSIAcsvYgiT+lQ
↪ 9nouWil7qcQIgFFNMgyl1IxIqVCHx++4hr32tJKwvABdpBiQsnjmdr/Q=","remember":false}
```

Listagem 31: Resposta do servidor

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Content-Type: application/json
Connection: close
Cache-Control: no-cache, private
Date: Tue, 16 Jun 2020 13:48:27 GMT
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 57
Set-Cookie: mirubee_session=F1Sv5LtewD3dY7yLhraWQnEwkhNiqCY1bsSKycCr;
↪ expires=Tue, 16-Jun-2020 15:48:27 GMT; Max-Age=7200; path=/; httponly
Content-Length: 54

{"success":true,"message":"authenticated through u2f"}
```

B

NOTIFICAÇÃO RESPONSÁVEL À SMILICS

Listagem 32: Notificação à Smilics dos erros identificados

```
Hi, my name is Gonçalo Ribeiro and I am a student of cybersecurity, currently undergoing a master's degree of cybersecurity and computer forensics.

Under the scope of my master's thesis I have done a security analysis of your platform and I have found some pressing security vulnerabilities which I would like to share with you, on the basis of good faith and responsable disclosure. I've been studing the Wibeee device and its platform from a cybersecurity point of view and found several vulnerabilities that I think should be addressed as soon as possible.
The first set of issues has to do with your mobile application and they were identified by analysing the application:

    1. Before sending the user password to your servers, the application hashes the password twice, firstly by using the MD5 algorithm and secondly using the SHA1 algorithm. This approach means that the final hash value never changes and therefore if captured can be replayed to authenticate on your site without knowing the password the user entered. Nowadays there are better alternatives like using the argon2 or bcrypt algorithms;
    2. This digested hash is then stored and kept even after the user has logged out which could lead to it being found by someone with access to the device's storage.

The second set of issues has to do with the web platform (https://smilics.com/wibeee)

    1. It seems like some endpoints of this platform do not use the HTTPS protocol (which was very recently implemented on said platform). Most notably the energy readings sent by your smart meters to your server are sent through the HTTP protocol which means they are sent without any kind of encryption on the transport layer;
    2. The energy readings are sent in cleartext. Since the readings are sent through HTTP and the smart meters do not encrypt the data sent (on the application layer) the readings are effectively being sent on clear text, meaning they could easily be intercepted by a third party;
    3. The energy readings are sent through a GET request without any kind of authentication done by the smart meter. It seems like your server identifies the user account which sent the reading through the MAC address of the smart meter (which is sent by field of the GET request). As such it is possible to falsify the MAC address field of the GET request to that of another smart meter of another user and effectively send false data to the account of another user.

Please feel free to contact me if you need more details about these vulnerabilities.

Kind regards,
Gonçalo Ribeiro.
```

Listagem 33: Resposta da Smilics

```
1  Hello Gonçalo, thank you very much for your kind and supportive email.
2
3  I commented it with the tech guys and they said they were aware of these weak
   ↪  points. In fact, we are preparing a new APP that will address this issue of
   ↪  the hash. Regarding the energy measures, we are changing the HW architecture
   ↪  to allow sending data in an encrypted way (the current chipset has not
   ↪  enough power to do encryption). It's a long work but we are already working
   ↪  on it.
4
5  Thank you for your offer anyway!
6
7  Best regards,
```

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado neste projeto, com o título *"Monitorização e Análise Segura de Consumos Energéticos"*, é original e foi realizado por Gonçalo Graça Ribeiro (2180076) sob orientação de Professor Doutor Miguel Monteiro de Sousa Frade (miguel.frade@ipleiria.pt).

Leiria, setembro de 2020

Gonçalo Graça Ribeiro

Gonçalo Graça Ribeiro