Security in SmartEmotion Platform

Mestrado de Engenharia Informática – Computação Móvel

Carla Mendes, 2220663
Carlos Costa, 2220662
Rafael Pereira, 2220659

A report presented for the degree of
Cybersecurity

Computer Engeneering
Leiria
23-01-2023

Security in SmartEmotion Platform[⋆,⋆⋆]

Carla Mendes[a,*,1] (Researcher), Rafael Mendes Pereira[a,*,1] (Researcher) and Carlos Eduardo Pereira Martinho da Costa[a,*,1] (Researcher)

[a] *Polytechnic Institute of Leiria, R. Gen. Norton de Matos Apartado 4133, 2411-901 Leiria, Portugal*

ARTICLE INFO

Keywords:
security, privacy, Personally Identifiable Information (PII), cyberattacks, log files, Security Information and Event Management (SIEM), Elastic

ABSTRACT

Nowadays, information is essential to any company or business process. Its compromise leads to consequences ranging from multiple domains, namely, financial, legal, reputation, etc [20]. Malicious clients can launch numerous diverse attacks (e.g Brute Force (BF), Distributed Denial of Service (DDoS), Denial of Service (DoS), etc) to damage the system's performance or to acquire important data illegally. Therefore, protecting the systems and their data is imperative in any business. SIEM systems allow the collection, filtering, analysis and presentation of real-time data regarding the network, devices, databases, servers, operative systems and even system and application logs [30, 27].

After a meticulous analysis of the existing SIEM systems, we propose the usage of the SIEM ELK Stack to monitor both the SmartEmotion platform and the Ubuntu virtual machine which is responsible for hosting our platform. The SmartEmotion platform aims to accompany continuously the older adult's emotional state and alert its caregiver of any alarming situation, any time and anywhere.

Furthermore, we researched the likelihood of vulnerabilities in our virtual machine and platform where we can conclude that the Ubuntu machine is vulnerable to BF attacks, whereas our platform can be a target of DoS or DDoS but is well protected against SQL injection attacks. Both the SmartEmotion platform and the Ubuntu virtual machine produce numerous log files, some of which are analyzed by our SIEM, where we created personalized dashboards and notification mechanisms with integrations with Microsoft Teams, Email and Discord. However, we also developed a customized log with a Python script to continuously evaluate the status of our most three important services in the platform: PHP, MySQL and Websockets. Both the Ubuntu virtual machine and the platform were test subjects of a BF and DoS attack respectively.

With the present work, we became aware of the high relevance and advantages of monitoring systems and devices with a SIEM system to assure both system's performance and security, continuously. Furthermore, acquiring knowledge regarding the existing SIEMs, log files and methods to simulate cyberattacks in machines or systems.

Contents

1 Introdução **1**

2 Background **2**
2.1 PII 2
2.2 Ciberataques 2
2.3 Ficheiro *log* 3
2.4 Sistemas SIEM 3

3 Plataforma **4**

4 Vulnerabilidades do sistema **5**

5 Proposta de Solução **6**
5.1 Ficheiros de *logs* analisados 6
5.2 SIEM 7
5.2.1 Instalação 7
5.2.2 Monitorização de ficheiros logs . . 7
5.2.3 Dashboard 8
5.2.4 Regras e Alertas de logs monitorizados 9

6 Testes às vulnerabilidades do sistema **10**
6.1 Ataque DoS 11
6.2 Ataque de força bruta à autenticação por SSH 11

7 Conclusão **12**

[⋆] This document is the results of the research project funded by the National Science Foundation.

[⋆⋆] The second title footnote which is a longer text matter to fill through the whole text width and overflow into another line in the footnotes area of the first page.

This note has no numbers. In this work we demonstrate a_b the formation Y_1 of a new type of polariton on the interface between a cuprous oxide slab and a polystyrene micro-sphere placed on the slab.

[*] Corresponding author

[**] Principal corresponding author

carla.c.mendes@ipleiria.pt (C. Mendes); rafael.m.pereira@ipleiria.pt (R.M. Pereira); carlos.m.costa@ipleiria.pt (C.E.P.M.d. Costa)

ORCID(s): 0000-0001-7138-7124 (C. Mendes); 0000-0001-8313-7253 (R.M. Pereira)

https://www.linkedin.com/profile/view?id='carla-mendes-5b3586233' (C. Mendes), https://www.linkedin.com/profile/view?id='rafaelmendespereira' (R.M. Pereira), https://www.linkedin.com/profile/view?id='carlos-costa-99baa5176' (C.E.P.M.d. Costa)

[1] This is the first author footnote. but is common to third author as well.

[2] Another author footnote, this is a very long footnote and it should be a really long footnote. But this footnote is not yet sufficiently long enough to make two lines of footnote text.

1. Introdução

Nos dias de hoje os dados e informação de um sistema ou empresa são recursos muito importantes para qualquer negócio, onde o comprometimento destes dados valiosos leva a diversos prejuízos quer a nível financeiro, legal, privacidade

e de reputação, quer dos clientes ou empresa, ou até a nível do funcionamento de serviços associados (onde a sua disponibilidade poderá estar comprometida) [20]. Os dados poderão ser comprometidos de inúmeros modos, quer seja através de divulgação indevida, modificações inapropriadas, ou até mesmo destruição dos mesmos [20].

Um ciberataque consiste numa tentativa intencional levada a cabo pelos atacantes de modo a ganhar acesso a sistemas e às respetivas informações. Os atacantes visam a tentar descobrir vulnerabilidades nos sistemas através da realização de diversos tipos de ataques, nomeadamente ataques de *malware*, *SQL injection*, *pishing*, *botnets*, BF, ou até mesmo *DoS* ou *DDoS* [40, 26].

O restante documento está estruturado conforme descrito em seguida. A secção 2 apresenta a descrição de conceitos envolventes à solução proposta, nomeadamente PII, ciberataques existentes, análise aos tipos de ficheiros *logs* existentes e ainda comparação entre diversos tipos de sistemas SIEM disponíveis no mercado. A secção 3 tem como objetivo introduzir a plataforma SmartEmotion, contexto de utilização, funcionalidades, componentes e arquitetura, e ainda alguns mecanismos de segurança implementados. A secção 4 visa a descrever algumas das vulnerabilidades encontradas na plataforma ou na máquina virtual onde está alojada. A secção 5 visa a descrever a solução desenvolvida, nomeadamente detalhando cada ficheiro de *log* utilizado e o sistema SIEM escolhido, desde a sua instalação, monitorização dos *logs*, *dashboards* criadas, regras e alertas para monitorização do estado do sistema onde a plataforma está alojada, e da própria plataforma. A secção **??** irá abordar as conclusões obtidas com a resolução do presente trabalho. As próximas secções consistem nos apêndices, onde a secção 7 apresenta algum código Python desenvolvido e a secção 7 apresenta algumas imagens relativamente às *dashboards* criadas.

2. Background

Esta secção tem como objetivo introduzir alguns dos conceitos que serão relevantes ao longo do presente documento, nomeadamente PII, descrição de ciberataques existentes, tipos de ficheiros *logs* e respetivos formatos, e por último a análise a diversos sistemas SIEM existentes no mercado.

2.1. PII

PII constitui todas as informações que poderão ser utilizadas para identificar um indíviduo, quer seja o seu nome completo, data de nascimento, número de segurança social, entre outros [21]. *Protected Health Information (PHI)* constitui toda a informação de saúde que permite identificar um indíviduo, e.g informações sobre prescrições, medicamentos, doenças, data de nascimento/morte, exames médicos, sendo um ramo de *PII* [38]. Personal Security Information (PSI) corresponde a informações que os utilizadores utilizam para aceder a serviços ou a informações na Internet, nomeadamente *usernames* e *passwords* [29]. A plataforma descrita anteriormente lida com dados pessoais que permitem identificar os utilizadores, nome, email, password, idade, fotos dos rostos, estado emocional, etc. Deste modo é necessário a toma de medidas para proteger estes dados dos utilizadores.

2.2. Ciberataques

Existem diversos tipos de ciberataques que podem causar graves danos aos sistemas, organizações e pessoas envolvidas, quer seja em termos de *malware*, base de dados, rede, entre outros. Deste modo ao longo desta secção irão ser introduzidos os respetivos ataques: *malware*, *SQL injection*, *pishing*, *botnets*, BF, ou até mesmo *DoS* ou *DDoS* [40, 26].

Nm ataque através de *malware* o atacante tem como objetivo instalar software malicioso, *malware*, nos sistemas operativos ou outros dispositivos das vítimas. O principal objetivo com este tipo de ataque é que o atacante ganhe acesso ao equipamento da vítima sem o seu conhecimento onde o atacante poderá aceder a PII dos utilizadores e rastrear as suas atividades. Os ataques de *malware* são classificados como *spyware*, vírus ou *ransomware* [40].

O ataque de *SQL injection* implica a utilização de *queries SQL* nos campos de texto de Websites ou aplicações de modo a atacar as bases de dados dos sistemas. Deste modo, com este tipo de ataque o objetivo consiste em obter acesso a dados restritos, que posteriormente poderão ser apagados ou falsificados, e garantir ao atacante a posse da base de dados [40].

Pishing consiste num ataque onde o objetivo é enganar os utilizadores a abrir emails, *pop-ups* ou outras mensagens enganosas tentando fazer com que o utilizador acredite na veracidade das mesmas. Deste modo, estas mensagens enganosas visam a roubar, ou enganar o utilizador de modo a que este forneça, algumas credenciais ou até informações financeiras. Os ataques *pishing* podem ser dividos em três categorias, nomeadamente *spear phishing*, *clone phishing* e *whaling* [40].

Num ataque com *Botnets*, um atacante consegue obter controlo sobre os computadores das vítimas através da instalação de *malware*. Após a obtenção do controlo sobre as máquinas, o atacante poderá iniciar um conjunto de atividades maliciosas de modo a interromper processos de negócio, obter recompensas financeiras ou até obter privilégios de administração [40].

Força bruta, ou *BF*, consiste num ataque onde o atacante efetua inúmeros pedidos para tentar obter as *passwords* do utilizador. Neste ataque, categorizado como "tentativa e erro", o atacante observa a vítima com o objetivo de desvendar algumas informações sobre a mesma que possam constituir alguma pista sobre as suas *passwords* [26].

Nos ataques de *DoS* e *DDoS*, o objetivo do atacante consiste em desacelerar ou bloquear serviços ou recursos por tempo indeterminado. Estes ataques implicam o envio de pedidos falsos em quantidades maiores do que o suportado pelo sistema ou serviço alvo, deste modo impossibilitando o sistema ou serviço de responder a pedidos de clientes reais. Num ataque de *DoS*, os pedidos falsos são provenientes de um único ponto, enquanto que num ataque de *DDoS* os pedidos são oriundos de diversas fontes distintas [40].

2.3. Ficheiro *log*

Os ficheiros *log* são essenciais para qualquer sistema, quer seja para efetuar a monitorização da performance, detetar alguns *bugs*, acompanhar o comportamento dos utilizadores, investigar problemas de segurança, entre outros [1]. Estes ficheiros, com a extensão .log, contêm texto informativo acompanhado de *timestamp* de modo a compreender quando os eventos ocorrem [41].

Existem diversos tipos de ficheiros *logs* [41], nomeadamente:

- *Logs* de eventos - registam eventos e atividades do sistema de modo a rastrear atividades e problemas do mesmo;
- *Logs* de transação - registam todas as transações que afetaram a base de dados, o que se torna muito útil para recuperação de falhas ou erros e para manter os dados consistentes;
- *Logs* de mensagens - estes são *logs* mais utilizados para guardar conversas em *chats* ou programas de mensagens instantâneas, normalmente são escritos ou em *plain text* ou JavaScript Object Notation (JSON);
- *Logs* de servidor - registam a lista de atividades executadas, criadas ou mantidas automaticamente pelo próprio servidor.

Existem diversos formatos para of ficheiros de *logs*, sendo que normalmente cada um dos formatos permite compreender se o conteúdo do ficheiro é estruturado ou semi-estruturado ou não estruturado; se os dados estão em binário ou texto simples, o mecanismo de codificação e a separação entre cada registo do *log*. Deste modo podemos destacar alguns dos formatos existentes: JSON, Event logs do Windows, Common Event Format (CEF), Common Log Format (CLF), Extended Log Format (ELF) e W3C [1].

O formato em JSON é o formato mais comum, onde os dados são semi-estruturados organizados entre pares chave/-valor, fáceis de ler, e com representação hierárquica. Para além disto JSON permite diversos tipos de dados distintos desde *strings*, valores booleanos e numéricos até objetos, vetores [1]. JSON normalmente utiliza codificação UTF-8 e não coloca restrições relativamente à quantidade, organização ou tipo de pares chave/valor que o ficheiro *log* em JSON pode incluir [1].

Os *event logs* do Windows armazenam dados relativos a eventos que ocorrem neste sistema operativo, como por exemplo dados sobre a segurança, aplicações, hardware, rede, DNS, entre outros. Este tipo de *logs* acabam por ser bastante detalhados, contendo *timestamp*, identificador do evento, nome do utilizador, *hostname*, a mensagem e a categoria da tarefa [1].

CEF é um outro formato de *logs* baseado em texto utilizado para aplicações e dispositivos mais relacionados com segurança. Este formato utiliza também UTF-8, apresentando ainda um cabeçalho CEF e ainda uma lista de tamanho variável de pares chave/valor. Este formato é muitas vezes utilizado pelos sistemas SIEM durante o processo de recolha e agregação dos dados [1].

CLF é um formato de *logs* mais antigo, baseado em texto e padronizado sendo muito utilizado por servidores web. Este formato é menos flexível com a personalização dos campos, onde cada linha contém sempre o endereço remoto do *host*, o nome do *log* remoto, o nome de utilizador, *timestamp*, o pedido HTTP realizado e a versão do protocolo, o *status-code* do pedido HTTP e o número de *Bytes* enviados [1].

ELF é muito utilizado por aplicações web, este formato é similar ao anterior no entanto permite que sejam indicados quais campos a utilizar ou não, deste modo este formato acaba por ser mais flexível. Este formato contém informações acerca de um pedido HTTP, e contém ainda dados sobre data, hora, *software*, etc [1].

W3C ELF é um formato utilizado pelos servidores IIS do Windows. A grande vantagem deste ficheiro consiste na sua personalização, onde podem ser indicados quais os campos a incluir, de entre os seguintes: *timestamp*, endereços IP do cliente e servidor, *URI-Stem*, o *status-code* do pedido HTTP, o número de *Bytes* enviados e recebidos, a duração em tempo e a versão [1].

2.4. Sistemas SIEM

Um dos mecanismos para garantir a seguranças dos dados implica a utilização de sistemas SIEM e Security Orchestration Automation and Response (SOAR). Um SIEM é definido como uma tecnologia que permite coletar, filtrar, analisar e apresentar em tempo real informações de redes e dispositivos de segurança, bases de dados, servidores, sistema operativos e ainda *logs* de aplicações e sistemas [30, 27]. Deste modo, permite agregar todos os dados a monitorizar numa única plataforma de maneira a permitir detetar, investigar e classificar algumas ameaças ou ataques, compreender a correlação entre os mesmos, e criação de *reports* [27].

Qualquer sistema SIEM permite visualizar informações acerca da origem de determinada atividade nomeadamente:

- **Utilizador** - nome de utilizador, estado da autenticação, localização atual;
- **Dispositivos** - Fabricante, modelo, versão do sistema operativo, endereço MAC, método de conexão à Internet, localização;
- **Postura** - conformidade do dispositivo com políticas de segurança, ficheiros antivírus e *patches* do sistema operativo.

Diversos são os sistemas SIEM que estão disponíveis no mercado, contudo devido ao âmbito académico do projeto, decidimos restringir-nos apenas a sistemas SIEM *open-source*. Deste modo, os seguintes sistemas SIEM foram analisados: OSSIM, ELK Stack, OSSEC, Wazuh, Metron e MozDef.

OSSIM [13] foi desenvolvido pela empresa AT&T Cybersecurity e permite recolha, normalização e correlação de eventos. Para além disto, apresenta as seguintes capacidades essenciais de segurança: descoberta de recursos, avaliação do nível de vulnerabilidade, deteção de intrusões, monitorização de comportamentos e correlação entre eventos. O OSSMi permite a leitura e análise de *logs* e criação de *reports* utilizando informações dos *logs* nas aplicações ou dispositivos analisados e utiliza ainda inteligência artificial de modo a ajudar os administradores na identificação de ameaças. Algumas das desvantagens deste SIEM consistem na dificul-

dade para personalizar o sistema e configurar o mesmo para sistemas Windows [43, 23].

Elasticsearch [15] consiste num motor de pesquisa, indexação e análise de dados (na sua maioria documentos) que integra o Elastic Stack. O Elasticsearch permite pesquisar e analisar, quase em tempo real, todos os tipos de dados quer sejam estes estruturados ou não, dados geoespaciais, numéricos. Deste modo para além de permite recolher e agregar dados este motor analítico permite ainda encontrar padrões nos dados [15, 43, 23].

O Logstash [8] é um motor de recolha de dados que utiliza um pipeline de modo a processar os dados do servidor em tempo-real. O Logstash permite assim agregar e normalizar os dados provenientes de inúmeras fontes simultaneamente e envia-os para um destino à escolha, que no nosso caso, consiste no Elasticsearch [8].

O Kibana [5] é ma ferramenta de visualização constituida por diversos *dashboards* de pesquisa e análise que podem ser visualizados nos *browsers*. Deste modo, o Kibana permite pesquisar, observar e proteger os dados através da análise de *logs* e outros documentos. O Kibana permite a criação de *dashboards* personalizados através da criação de gráficos, mapas, tabelas, entre outros, permitindo assim ua análise extensiva e visual dos dados [5].

Filebeat [4] permite o envio e centralização de *logs* sendo assim definido como expedidor [4]. O Filebeat é instalado nos servidores de modo a monitorizar os ficheiros de *logs* ou qualquer outra diretoria especificada pelo utilizador e envia os mesmos quer seja para o Elasticsearch ou para o Logstash [4].

Deste modo o ELK Stash apresenta algumas vantagens nomeadamente o facto de ser constituído por módulos, boa capacidade de gestão e agregação de ficheiros e dados, um *frontend* adaptável/personalizado e por permitir aos utilizadores criarem as próprias regras de deteção de ameaças, trás ainda vantagens como pré-configurações, mecanismos inteligentes de apoio à análise e integrações pré-definidas e configuráveis para sistemas mais complexos [43, 23].

OSSEC [12] é um software da empresa Atomicorp mais conhecido para a deteção de intrusões que apresenta mecanismos de configuração de acordo com as necessidades de cada sistema, de análise e correlação entre diversos de *logs* de variados dispositivos, capacidade de deteção de ameaças, integração com o sistema ELK mencionado anteriormente. Este sistema está dividido entre dois grandes componentes: servidor e agentes [12, 43, 23].

O servidor é responsável por coletar os *logs* das várias diversas fontes de informação sendo que os agentes permite o processamento e a análise do conteúdo desses *logs*, deteção de intrusão, validação de integridade entre outros. A sua principal desvantagem consiste em precisar de ferramentas de terceiros para análises mais extensivas (exemplo: o Kibana da Elastic) [12, 43, 23].

Wazuh [42] consiste num sistema que resultou de uma ramificação do sistema anterior, OSSEC. Wazuh permite detetar ameaças, vulnerabilidades e intrusões, responder a incidentes, monitorizar integridade e conformidade dos dados e sistema, segurança da *cloud*, *containers*, análise dos dados de *logs*, análise de segurança. Deste modo, este sistema é muito versátil e permite a integração com o Elastic Stack [42, 43, 23].

Apache Metron [2] é uma *framework* que permite detetar e responder a ciberataques visto que consegue receber, processar e guardar dados de várias fontes diferentes inclusive de protocolos de rede. Para além disto, Apache Metro é constituída por quatro grandes funcionalidades: *data lake* seguro, *framework* conectável, aplicação de segurança que permite realizar diversas funcionalidades de um SIEM, sendo que por último ainda incorpora algoritmos de *Machine Learning (ML)* para deteção de anomalias e técnicas de defesa contra ataques [2, 43, 23].

MozDef [14] consiste num sistema constituído microserviços criado pela Mozilla Foundation que está integrado também com o ELK Stack, nomeadamente utiliza o Elasticsearch para registar, indexar e armazenar dados e o Kibana para criar os dashboards e pesquisa de dados de eventos. Umadas vantagens do MozDef consiste em este ser um microserviço leve que aproveita muitas funcionalidades do ELK Stack, contudo isto leva também à sua desvantagem de ser mais um complemento para outro SIEM existente do que um produto independente e de precisar de imenso tempo para configuração e aprendizagem [14, 43, 23].

Em suma, com base na análise de diversos sistemas SIEM diferentes, o sistema SIEM escolhido foi o ELK Stack tendo em consideração as suas inúmeras funcionalidades e ferramentas disponibilizadas e capacidades de personalização. Comparativamente com outros sistemas SIEM mencionados, que não são da autoria da empresa Elastic, que utilizam ferramentas do ELK Stack, acreditamos ser uma melhor decisão utilizar o SIEM nativo da Elastic, ELK Stack.

3. Plataforma

Nos dias de hoje familiares idosos passam muito tempo nas suas habitações sozinhos, este é um problema que veio a agravar-se com a pandemia COVID-19 onde os mesmos foram impossibilitados de receber visitas de familiares ou amigos. Desta forma a sua solidão e isolamento cresceram ainda mais, o que levanta questões relativamente à sua segurança e bem-estar físico e mental. Consequentemente causando ainda enorme preocupação aos cuidadores destes idosos.

Deste modo, foi implementada uma solução, SmartEmotion, que permita aos cuidadores acompanharem remotamente e em tempo-real o estado emocional das pessoas idosas ao seu encargo e simultaneamente alertar os cuidadores quando ocorrer uma alteração do estado emocional do idoso quando o mesmo se encontra sozinho em casa. Desta forma procura-se garantir a saúde e bem-estar dos idosos, providenciando ainda um sentimento de segurança aos cuidadores.

O protótipo SmartEmotion é composto por um serviço, Elder Home, que acompanha o estado emocional do idoso (fora do *scope* deste projeto) e por uma plataforma que permite ao cuidador receber alertas da sua alteração. A plataforma permite analisar o ambiente envolvente para questões de segurança física da pessoa idosa, configurar o sistema tendo

em conta as preferências dos cuidadores (quais emoções a notificar, tempo entre as mesmas) e ainda realizar a classificação das imagens capturadas com emoções personalizadas. Paralelamente a isto, a plataforma permite ainda aos administradores do sistema gerirem o sistema.

A plataforma, cuja arquitetura se pode observar na Figura 1, é constituída por: um *website*; uma interface de programação de aplicações (API); uma base de dados; e um protocolo de comunicação assíncrona.

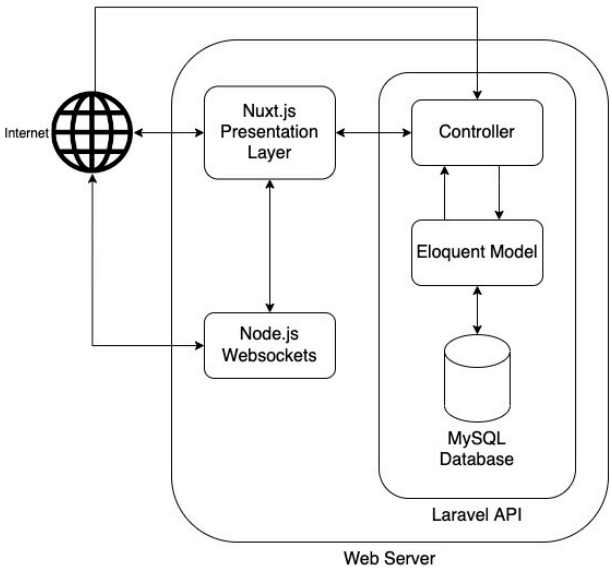


Figure 1: Arquitetura da plataforma SmartEmotion.

O *website*, implementado utilizando Nuxt.js e Vuetify, disponibiliza uma interface gráfica de modo a permitir que os cuidadores e administradores acedam à informação pretendida a qualquer hora e em qualquer lugar desde que esteja garantido acesso à *internet*. A API, concebida em Laravel, é responsável pelo processamento, validação, persistência e divulgação dos dados de todo o sistema, sendo que esta recebe e processa todos os pedidos provenientes tanto do sistema Elder Home como da plataforma Web. Após o recebimento e validação dos dados, a API persiste os mesmos numa base de dados desenvolvida com MySQL. Para além disto, foi ainda concebido em Node.js um protocolo de comunicação assíncrono, Websockets, de modo a garantir a partilha de notificações em tempo real entre o sistema Elder Home e o *Website*.

A acessibilidade e a disponibilidade que a plataforma SmartEmotion deve ter é uma preocupação, isto quer dizer que deve estar disponível a partir de qualquer lugar para qualquer cuidador, administrador ou sistema Elder Home em qualquer momento. Assim, toda a plataforma foi publicada num domínio interno do Politécnico de Leiria usando uma máquina virtual Ubuntu 20.04. Está disponível com o seguinte endereço: `http://aalemotion.dei.estg.ipleiria.pt`.

Durante a fase de desenvolvimento do sistema foram implementados alguns mecanismos para assegurar a segurança a vários níveis, desde mecanismo de conexão à máquina virtual, autenticação na plataforma, entre outros. O protótipo desenvolvido conta ainda com a utilização do protocolo SFTP para transferir os ficheiros para a máquina virtual Ubuntu, de modo a atualizar todo o serviço quando são realizadas atualizações. No entanto, sem ser para o envio de ficheiros, as comunicações com a máquina virtual Ubuntu utilizam o protocolo SSH de modo a assegurar um acesso remoto à máquina seguro. A plataforma SmartEmotion utiliza ainda o OAuth2 como serviço de autenticação o que implica que o acesso aos dados do cliente são limitados através do uso de *tokens*. Além de mais está configurado ainda o *reverse proxy*, que fica atrás da *firewall* e que tem como objetivo direcionar os pedidos dos clientes para o servidor *backend* desenvolvido providenciando um nível adicional de abstração e controlo maior. A comunicação com a plataforma atualmente é feita pelo protocolo Hypertext Transfer Protocol (HTTP), porém foi iniciada a aplicação do certificado seguro – Hypertext Transfer Protocol Secure (HTTPS), ao qual a plataforma está atualmente à espera da resposta do administrador de sistema para aplicação da mesma.

Em suma, assume-se que a plataforma tem capacidade para gerar registos de *logs* suficientes o que torna viável a implementação de um SIEM. Devido à plataforma estar alojada num domínio privado e monitorizado pelos administradores de sistema, pertencente à escola Instituto Politécnico de Leiria, que já implementam barreiras de segurança pré-definidas.

4. Vulnerabilidades do sistema

Esta secção tem como objetivo introduzir e detalhar as vulnerabilidades encontradas na plataforma SmartEmotion e na máquina virtual Ubuntu responsável por alojar a plataforma.

O *login* por SHH permite ao utilizador conectar-se a máquina virtual Ubuntu onde a plataforma está alojada. De modo a efetuar o *login* o utilizador poderá introduzir a *password* um total de três vezes, caso a autenticação seja mal sucedida após todas as tentativas, torna-se necessário a introdução do comando para a autenticação por SSH novamente. No entanto, nenhum mecanismo impede um utilizador malicioso de realizar diversas tentativas de *login* por SSH à máquina virtual, quer seja à mão ou com um *script*. Deste modo, algum atacante poderá tentar realizar um ataque de força bruta, descritos na secção 2, de modo a descobrir a palavra-passe de um utilizador do sistema, o que constitui uma das vulnerabilidades do sistema Ubuntu.

A *framework* Laravel, utilizada para o desenvolvimento da plataforma SmartEmotion, define o Eloquent Object-Relational Mapping (ORM) como a implementação do padrão arquitetural de ORM ActiveRecord [6, 33]. Deste modo, cada tabela da base de dados está associada a um modelo e o padrão Eloquent ORM permite converter de forma fácil os modelos e as suas relações em *arrays* ou JSON, e vice-versa. Para além disto o Eloquent ORM fornece ainda métodos para aceder a cada propriedade de um modelo de dados, deste modo excluindo a necessidade de utilizar *raw* SQL *queries* na maioria dos casos, o que evita os ataques de SQL *injection* descritos na Secção 2 [33]. Para além disto, foram ainda

definidas classes do tipo FormRequest, que consiste num tipo de classes que encapsula a lógica de validação e autorização, que permitem efetuar validações de dados complexas [6]. Deste modo o seguimento de boas práticas e utilização correta dos padrões definidos pelo Laravel tornam inviável a ocorrência de ataques de SQL *injection*.

A plataforma SmartEmotion poderá ser ainda alvo de ataques DoS e DDoS, descritos na Secção 2, visto que utilizadores maliciosos poderão tentar enviar inúmeros pedidos HTTP falsos à plataforma (numa quantidade superior à suportada pelo servidor), e que não existe nenhum limite definido na plataforma relacionado com o número máximo de pedidos que um utilizador pode realizar num determinado espaço de tempo.

5. Proposta de Solução

Ao longo desta secção será introduzido o processo da aquisição dos ficheiros *logs* situados na máquina virtual Ubuntu, instalação e descrição das ferramentas utilizadas para obter um SIEM, nomeadamente alguns dos produtos da Elastic: Elasticsearch, Logstash, Kibana e Filebeat.

5.1. Ficheiros de *logs* analisados

A presente secção visa descrever a análise manual realizada aos ficheiros *logs* para fundamentar o seu uso posteriormente no SIEM.

Tal como foi previamente mencionado na secção 2.3 os ficheiros *logs* permitem monitorizar sistemas e equipamentos, acompanhar o comportamento dos utilizadores, investigar problemas de segurança entre outros. Deste modo nos ficheiros *logs* registados qualquer tipo de ações, desde eventos do *kernel* até ações dos utilizadores. O sistema operativo Ubuntu, que constitui uma distribuição de Linux, contém uma pasta específica que armazena os ficheiros *log*, localizada na diretoria "/var/log". Esta pasta contém ficheiros *logs* do próprio sistema operativo, serviços e as diversas aplicações em execução no sistema.

Um dos ficheiros *log* mais relevantes desta diretoria consiste no ficheiro "auth.log", que regista todas as atividades dos processos do sistema relacionados com autenticação e autorização. Deste modo, este ficheiro regista todos os *logins*, tentativas de *login*, *logouts* realizados ao sistema e todas as ocorrências da utilização do comando *sudo*. No caso de qualquer autenticação ou *logout* ter sido mal sucedido, este ficheiro indica também a causa (sessão terminada, *password* incorreta, entre outros). Este ficheiro anota também todos as utilizações do comando *sudo*, visto que é um comando que fornece privilégios de administrador ao utilizador autenticado para a realização de um comando específico. Cada registo deste ficheiro indica sempre a data e hora, *hostname*, nome do processo e PID (ou nome do comando) e a mensagem respetiva. No caso do registo ser sobre a utilização de um comando,a mensagem irá contém o nome do utilizador que executou o comando, o comando e a diretoria. Caso contrário, esta mensagem é relativa ao processo em causa [25].

O ficheiro *log* "syslog" armazena todas as informações relativas ao estado global do sistema Ubuntu e das suas aplicações, sendo que cada registo neste ficheiro contém: data e hora, *hostname*, nome do processo conjuntamente com o PID e por último a mensagem fornecida pelo processo [25].

Para além dos ficheiros *logs* mais focados na monitorização do sistema Ubuntu mencionados anteriormente, podemos ainda destacar os ficheiros *log* associados ao serviço do nginx (disponíveis na diretoria "var/log/nginx"): "access.log" e "error.log". O ficheiro "access.log" é responsável por armazenar todos os pedidos realizados pelos clientes ao Website SmartEmotion [11]. Cada registo deste ficheiro conta com o endereço IP remoto para onde foi feito o pedido, a data e a hora, o pedido, o seu *status code*, número de *Bytes* enviados no *Body* e informações do cliente que realizou o pedido (nome, sistema operativo, versão, Browser utilizado, entre outros) [9, 11, 45]. O ficheiro "error.log", por outro lado contém informações sobre quaisquer erros, avisos ou *bugs* que ocorreram, quer seja erros de permissão, ou de configuração do próprio nginx [11, 45]. Cada registo no "error.log" contém data e hora, nivel de registo (no total existem oito: *emerg*, *alert*, *crit*, *error*, *warn*, *notice*, *info* e *debug*), PID e TID do processo e a mensagem [45, 3].

De entre os *logs* disponíveis, podemos destacar ainda os *logs* relativos ao serviço MySQL e à respetiva base de dados, que estão localizados na diretoria "var/log/mysql". No entanto estes *logs* não são criados por omissão, deste modo tornou-se necessário trocar os valores de algumas variáveis do sistema de modo a ativar a utilização destes *logs*, nomeadamente as variáveis "general_log" e "slow_query_log" que passaram a ter o valor de 1, e reinicializar o serviço MySQL.

O serviço MySQL apresenta três ficheiros de *logs* distintos: "mysql-slow.log", "error.log" e "query.log". Uma *query* cujo tempo de execução ultrapasse o valor especificado pela variável de sistema *long-query-time* será adicionada ao ficheiro "mysql-slow.log". Cada registo nos ficheiros "mysql-slow.log" e "query.log" contém a *query* executada, o id e a data e hora da conclusão da mesma. Para além disto, o ficheiro "mysql-slow.log" apresenta ainda mais informações, nomeadamente: informações do utilizador (nome e *host*), o tempo de processamento, tempo de espera (na eventualidade de ter ocorrido um *lock* na base de dados), número de linhas resultantes, número de linhas examinadas e ainda o número de *Bytes* enviados [10, 7]. O ficheiro "error.log" consta de mensagens, avisos e outras informações que foram geradas durante as operações do servidor, e até durante as fases de *startup* e *shutdown*. Este ficheiro apresenta os seguintes campos: data e hora, ID da *thread*, o tipo de erro (que pode ser categorizado como "system", "note", "warning" ou "error"), o código do erro, o subsistema MySQL e por último a mensagem [10].

A fim de explorar ainda mais *logs*, nomeadamente averiguar o funcionamento da monitorização de *containers* mantidos pelo software de virtualização Docker e algumas métricas relevantes, decidimos criar um *container* contendo uma cópia da base de dados da plataforma SmartEmotion, é de ressalvar que esta cópia da base de dados não apresenta qualquer utilidade. Os *logs* mantidos pelo Docker poderão ser observados através de um comando indicando as *keywords* "docker

logs" seguindo-se do nome ou ID do *container*, que pode ser obtido com o comando "docker ps". Deste modo, os *logs* do Docker permitem acompanhar o estado dos *containers* e respetivos serviços, a sua estrutura é similar à do ficheiro "error.log" do serviço MySQL mencionado anteriormente [17].

Existem três serviços muito críticos para o funcionamento da plataforma, nomeadamente PHP, MySQL e Websockets. O serviço MySQL não gera *logs* em tempo real sobre o estado do serviço. Adicionalmente, os ficheiros *logs* do PHP também não contêm informações sobre o estado do próprio PHP. Por último, os *logs* criados pelo gestor de processos pm2 (responsável por gerir o serviço dos Websockets) não estão num formato compreensível por um SIEM. Tendo em conta estas três situações e a elevada importância destes serviços, foi desenvolvido um *script* em Python para gerar *logs* contendo o estado dos três serviços em tempo real. Deste modo este *script*, que está constantemente em execução, observa, de trinta em trinta segundos, o resultado dos seguintes comandos:

- sudo systemctl status php7.4-fpm
- pm2 status index
- systemctl status mysql

Após execução destes comandos, o *script* adiciona ao ficheiro *log* criado, "logger.log", o estado dos três serviços mencionados acima. O código deste *script* encontra-se disponível na secção 7.

5.2. SIEM

A presente secção visa descrever o processo de escolha e instalação do SIEM, monitorização de ficheiros logs explicando como é realizado o processo de extração e transformação dos *logs* referidos na Secção 5.1, a visualização em *dashboards* que espelham a análise realizada aos *logs* e por último.

O sistema SIEM escolhido foi o ELK Stack tendo em consideração as suas inúmeras funcionalidades, ferramentas disponibilizadas, e capacidades de personalização. Este, comparado com outros mencionados na Secção 2.4 oferece as ferramentas ideais para este desenvolvimento.

5.2.1. Instalação

De modo a realizar a monitorização de eventos nesta máquina e posterior integração com o SIEM escolhido: ELK Stack, tornou-se necessário a instalação dos seguintes serviços na máquina virtual que constituem o ELK Stack: Elasticsearch, Logstash, Kibana e Filebeat ao qual a instalação realizada foi feita seguindo o tutorial do Sr. Shweta Mamidwar em [34]. Todos os serviços referidos são inicializados no momento em que a máquina liga.

As especificações da máquina virtual:

- **OS**: Ubuntu 20.04.4 LTS x86_64
- **Host**: VMware Virtual Platform None
- **Kernel**: 5.4.0-137-generic
- **Shell**: bash 5.0.17
- **Terminal**: /dev/pts/2
- **CPU**: Intel Xeon Gold 6226 (1) @ 2.693GHz
- **GPU**: 00:0f.0 VMware SVGA II Adapter
- **Memory**: 5804MiB / 7957MiB

5.2.2. Monitorização de ficheiros logs

A monitorização de logs é uma das fases cruciais para o bom funcionamento de um SIEM [39, 16], é com base nesta monitorização que é exposto o estado, processos e recursos executados que têm como finalidade trazer a visualização dos mesmos por meio de dashboards como já descrito na Secção 5.2.3 ou aplicação de regras para desencadear alertas. A presente secção visa descrever como foi realizado o processo de monitorização de logs tendo como base os ficheiros analisados na Secção 5.1.

Neste processo de monitorização, é necessário identificar que logs já são gerados pelo sistema ou serviços, e quais devem ser gerados, assim para posteriormente usar o o *Filebeat* e proceder à extração (ExtractTL) destes, *Logstash* para a transformação (ETransformL) e o *Elasticsearch* para o carregamento (ETLoad) na interface visual *Kibana* [19].

Com base nos ficheiros logs referidos na Secção 5.1, segue-se a lista de quais os logs que será descrito o processo de monitorização:

- **/var/log/auth.log** – Logs de autenticação no sistema.
- **/var/log/syslog** – Logs de gerais do sistema.
- **/var/log/nginx/access.log** – Logs de acesso à plataforma que passam no *Nginx*.
- **/var/log/mysql/mysql-slow.log** – Logs de acesso ao serviço e base de dados *MySQL*.
- **/var/run/docker.sock** – Socket que permite a comunicação com os *containers* alojados no *Docker* para recolha de logs.
- **$HOME/logger.log** – Logs customizados e gerados que descrevem o histórico do estado dos serviços relevantes para o funcionamento da plataforma.

De seguida será brevemente descrito como é realizada a monitorização de cada ficheiro log seguindo o padrão de integração de dados Extract Transform Load (ETL) [18].

Os logs de autenticação no sistema descrevem as tentativas bem e mal sucedidas de acesso ao sistema, estas tentativas e estão descritas no ficheiro /var/log/auth.log. Os logs deste ficheiro são observados por dois módulos do *Filebeat*, um em contexto de análise de sistema, e outro em contexto de controlo de autenticação. A configuração do módulo de análise de sistema é pre-desenvolvido pelo ELK Stack e não exige qualquer configuração adicional. A configuração do módulo de controlo de autenticação foi configurado foi configurado manualmente, o que requere a localização do ficheiro para que este possa ser extraído, a definição de um *namespace/dataset* para o qual se pretende monitorizar os logs. Relativamente ao processo de transformação neste módulo, é de referir que não existe configuração adicional para além da configuração por defeito do *Logstash*, que já recolhe e estrutura a informação providenciada pelo observador do *Filebeat* como por exemplo – a data e hora do log, informação do host e mensagem recolhida do ficheiro. O processo de carregamento é gerida pelo *Elasticsearch* que recolhe métricas,

analisa os logs como um todo, e providência toda a estrutura para visualização na interface de utilizador *Kibana*.

Relativamente aos logs gerados pelo sistema, os logs de acesso à plataforma, acesso ao serviço e base de dados *MySQL*, e aos logs gerados pelos *containers* alojados no *Docker* a monitorização destes foi praticamente automática, toda a configuração é pré-desenvolvida pelo *Logstash*, e não foi adicionada qualquer informação adicional. Na Secção 5.2.3 foram descritas as dashboards provenientes das métricas recolhidas por estes módulos, já na Secção 5.2.4 será descrito o processo de como foram desenvolvidas as regras para desencadear os alertas.

A monitorização dos logs customizados que acompanham e registam o estado dos serviços relevantes para o funcionamento da plataforma é realizada de maneira diferente aos demais. Isto acontece devido a pretender-se que a informação recolhida seja estruturada com os seguintes campos – nível de importância do log, qual o nome do processo de onde foi recolhida a informação e o seu estado. O processo de extração de dados provenientes da execução do *script* de python referido na Secção 5.1 é realizado da mesma maneira que os módulos referidos a cima nesta secção. O processo de transformação de dados é realizada de maneira diferente, com recurso às pipelines de processamento do SIEM. Uma pipeline permite analisar cada linha de log que é monitorizada, permitindo assim criar campos calculados, dividir a mensagem, retirar campos desnecessários, e entre muitas outras coisas. Sendo o objetivo destes logs monitorizar o estado dos serviços é necessário desagregar a informação da mensagem original como demonstrado na Figura 2. Para estruturar a informação é utilizado padrões Grok [36**?**], que têm grandes semelhanças às expressões REGEX [24]. Porém com algumas palavras-chave incluídas e retornam dados estruturados com base nos dados recebidos que serão usados para a visualização, controlo e gestão dos logs.

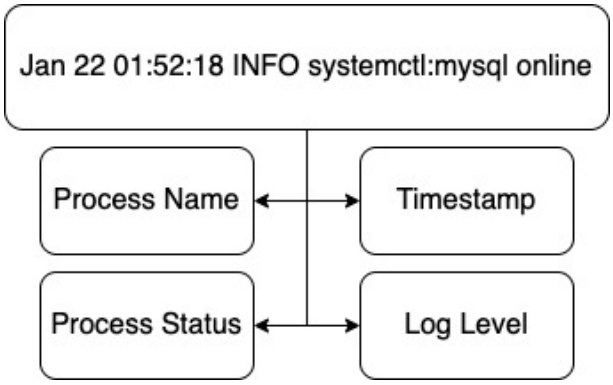


Figure 2: Transformação de logs estruturados

Em suma, com base no que foi anteriormente descrito pode-se referir que a monitorização de logs é um fator crucial para a visualização de anomalias e fluxo do funcionamento de qualquer sistema. A monitorização de logs é bastante facilitada pelo SIEM ELK Stack, trazendo bastantes configurações pré-definidas e provando que este é uma boa escolha para análise de logs. As configurações adicionais realizadas são apenas estruturais e encaixam-se em contextos mais específicos onde é necessária uma transformação mais customizada.

5.2.3. Dashboard

Para mostrar algumas informações e métricas relevantes dos *logs*, foram criadas *dashboards* no *Kibana* para demonstrar mais facilmente para o utilizador estas informações. Na presente secção são descritas as *dashboards* desenvolvidas com base nos *logs* extraídos e transformados referidos na Secção 5.2.2, as *dashboards* a seguir referidas estão disponíveis em figuras nos apêndices na Secção 7.

Para espelhar as métricas do sistema recolhidas após a análise dos *logs* respetivos foi desenvolvida uma *dashboard* que disponibiliza informação do número de *hosts*, percentagem de CPU a ser usado, memória e disco em uso, tráfico de entrada e saída, histórico de registo de *logs* de sistema, número total, aceites e não aceites de tentativas de conexão por SSH. De modo a visualizar o histórico e estado atual dos serviços críticos para o funcionamento da plataforma (Websockets, PHP, MySQL) definimos uma *dashboard* (representada parcialmente na Figura 3), que para cada serviço crítico, apresenta informações sobre o estado do serviço durante a última hora, uma tabela com o histórico de recolha de informação estruturada, e um gráfico que mapeia o estado do serviço.

Figure 3: Métricas do serviço Websocket presente na dashboard - Métricas do Sistema

Com o objetivo de visualizar as entradas e métricas da informação relativa a conexões por SSH foi criada uma *dashboard* que apresenta quantas tentativas de conexão bem ou mal sucedidas ocorreram, e o fluxo de conexões por SSH em formato de tabela.

Para visualizar as métricas da análise realizada aos *logs* de comunicação com a plataforma foi estruturado e desenvolvida uma *dashboard* que contém o total e histórico dos pedidos integrais, dos que usam métodos *http POST* e *GET*, e dos os pedidos falhados. A *dashboard* contém também o fluxo de comunicação em formato tabela, um mapa que usa a região geográfica onde foram realizados pedidos e métricas quantitativas e descritivas sobre o tráfego realizado ao *end-point* de autenticação à plataforma web.

De modo a monitorizar os *containers* do Docker foi criada uma *dashboard*, cuja apresenta métricas como: a utilização do CPU, memória percentual e quantitativa, número

de operações *input/output* realizadas no disco, e o número de *containers* envolvidos. Apresenta quantos *containers* estão ativos pausados ou parados. Contém também dois gráficos circulares verdes que indicam quantos *containers* cada *host* contém e as imagens associadas a cada *container*. Os restantes dois painéis indicam com base no fluxo temporal a percentagem de utilização do CPU, a quantidade de memória utilizada e a quantidade de Bytes que entraram e saíram de cada *container*.

De modo a ter uma visão geral do serviço MySQL foi estruturada uma *dashboard* que trás visualizações como gráficos que mostram num contexto temporal a quantidade de *logs* lentos e de erro do serviço MySQL, quais as *queries* mais executadas num gráfico circular, e quais os níveis de erro que mais ocorrem também num gráfico circular. Em formato tabela é possível ter acesso a quais as *queries* que demoraram mais tempo a ser executadas, o tempo que demoraram e o utilizador da base de dados que as lançou. No último painel é visível quais os erros que ocorreram no serviço com a data que ocorreu e o nível do erro respetivo.

Foi desenvolvida uma *dashboard* que permite visualizar as métricas dos *logs* gerados pela execução das *queries* à base de dados MySQL que está a ser usada pela plataforma SmartEmotion, e trás visualizações como: o número de *queries* do tipo SELECT, INSERT, UPDATE e DELETE. Permite avaliar quantas conexões foram realizadas à base de dados e quantas destas conexões foram abortadas. Ainda poderão ser observadas o número de *threads* que estão ativas relativas à base de dados, o número de páginas utilizadas e livres do *buffer pool*, percentagem de utilização e eficiência do mesmo.

Em suma, podemos afirmar que as *dashboards* descritas trazem grandes vantagens para a visualização da informação recolhida após a transformação de *logs*. O ELK Stack oferece ferramentas que auxiliam e simplificam o processo de criação dos paineis de visualização tornando-se uma mais valia da sua utilização.

5.2.4. Regras e Alertas de logs monitorizados

Uma regra permite especificar uma tarefa que corre em *background* no servidor de modo a verificar determinadas condições, a regra divide-se em três partes, condições. cronograma e ações como consta a Figura 4. A condição da regra indica o que deverá ser detetado, o cronograma define o intervalo de tempo (nº de vezes) para verificar se a condição foi detetada, e por último, a ação indica o que deverá acontecer quando a condição é detetada, onde normalmente as ações implicam conexão a serviços internos e de terceiros. Um alerta consiste numa instância em que a condição foi detetada pela tarefa que está a correr em *background* servidor. Deste modo, uma ação tem um conector (constitui um ponto central para guardar as informações da conexões a serviços) associado para o qual a informação sobre o alerta será enviado [5].

Deste modo, foram definidas um total de sete regras referenciadas na lista baixo. Na eventualidade de algum dos alertas serem desencadeados os conectores realizam o envio de mensagens personalizadas para o Microsoft Teams, Email ou Discord, que serão detalhados posteriormente à explicação das regras definidas.

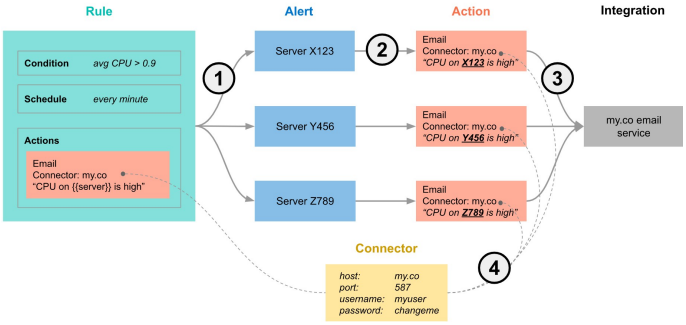


Figure 4: Diagrama com todos os componentes para a deteção de eventos. Retirado de [5]

- Falha de autenticação ao sistema por SSH.
- Falha de autenticação à plataforma web.
- Limite de número de logs.
- Serviço PHP desligado.
- Serviço de Websockets desligado.
- Serviço de Websockets a reiniciar.
- Serviço de MySQL desligado.
- Deteção de grande fluxo de acessos à plataforma para prevenir ataques DoS.

A primeira regra, falha de autenticação ao sistema por SSH, visa a monitorizar os acessos falhados por SSH à máquina virtual onde todo o sistema está alojado, sendo que devido à importância do acesso à máquina decidimos que estes acessos devem ser verificados de minuto a minuto e desencadear. A regra aqui consiste em verificar se durante os últimos cinco minutos ocorreram duas ou mais autenticações falhadas por SSH e na eventualidade de o alerta ser desencadeado é enviada uma mensagem para o Microsoft Teams, Email e Discord utilizando os conectores. A segunda regra, visa a averiguar se um mesmo utilizador falhou o processo de *login* na plataforma Web (resultando em pedidos com o *status code* de 403 que indica falta de autorização) mais do que cinco vezes durante um período de cinco minutos, de modo a tentar evitar ataques de força bruta através de diversas tentativas erro para descobrir as passwords dos utilizadores e entrar nas suas contas. Esta regra à semelhança da anterior é verificada de um a um minuto e eventuais alertas que ocorrem são enviados também para o Microsoft Teams, Email e Discord.

A terceira regra, desenvolvida tem como objetivo monitorizar, de minuto a minuto, se foram encontrados quinhentos ou mais novos registos de *logs* no *syslog* durante os últimos cinco minutos, enviando quaisquer alertas para o Microsoft Teams. A quarta regra, verifica de minuto a minuto se o PHP esteve *offline* pelo menos uma vez durante os últimos cinco minutos, e envia o alerta desencadeado para Microsoft Teams, Email e Discord em intervalos de cinco minutos. Esta regra foi criada devido ao facto que o serviço

do PHP corresponde ao *Backend* da plataforma SmartEmotion, e qualquer interrupção no seu funcionamento implica consequências para os utilizadores do sistema.

A quinta regra tem como objetivo verificar, de minuto a minuto, se o serviço do MySQL esteve *offline* pelo menos uma vez durante os últimos cinco minutos. Caso o alerta seja desencadeado é enviado uma mensagem através dos conectores para o Microsoft Teams, Email e Discord.

As duas regras seguintes são referentes aos WebSockets, outro dos componentes muito relevantes para o sistema devido ao facto de os WebSockets serem responsáveis pelo envio assíncrono de informação e notificações, onde mais uma vez qualquer interrupção trará consequências para os utilizadores. Tendo isto em conta, as duas regras verificam se de um em um minuto o serviço dos Websockets esteve *offline* pelo menos uma vez durante os últimos cinco minutos, respetivamente. Em qualquer uma destas duas regras caso o alerta aconteça é enviado uma mensagem para as três plataformas, Microsoft Teams, Email e Discord.

A última regra é referente à deteção de ataques DoS e tem como objetivo detetar grandes quantidades de acessos à plataforma pelo *nginx*. Esta regra é ativada quando ocorrem mais de quinhentos novos registos nos últimos cinco minutos e ao ser ativada desencadeia o envio de mensagens para as plataformas Microsoft Teams, Email e Discord.

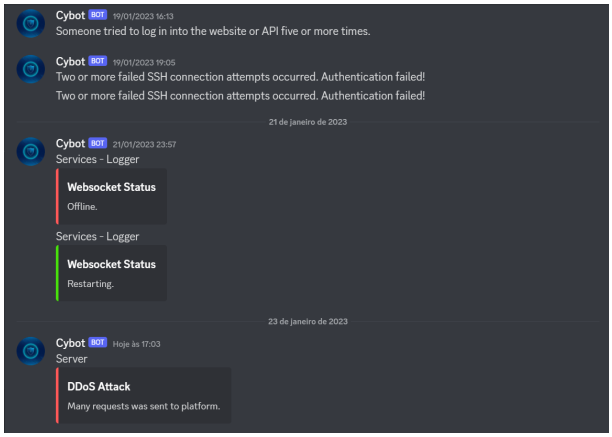
Tal como foi referido anteriormente, foram definidos três conectores, Microsoft Teams, Email e Discord. As Figuras 5c, 5b e 5a demonstram as mensagens enviadas pelos conectores para os respetivas plataformas Microsoft Teams, Email e Discord. O Discord foi o único conector do tipo *webhook*, e neste conector (denominado de "Cyb Discord") foram definidas algumas configurações como o nome do conector, o URL e o pedido HTTP correspondente, e os cabeçalhos para o pedido. Para a criação deste conector foi criado um servidor no Discord para onde irão ser enviados os alertas.

O conector para o Email, de nome "Email Sender", poderia ser configurado com diversos serviços de email diferentes, nomeadamente Gmail, Outlook, Amazon SES, MS exchange server, entre outros. O serviço escolhido foi o Outlook o que definiu automaticamente campos como o *host* do serviço e o porto. Deste modo foi criado o email responsável pelo envio dos alertas.

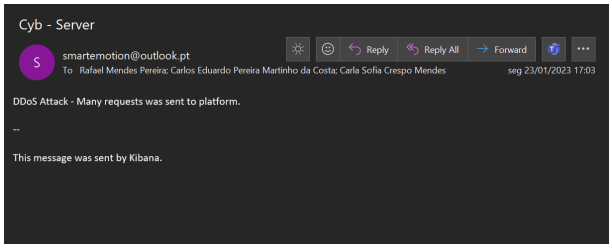
O conector para o Microsoft Teams implicou a criação de uma equipa no Microsoft Teams, e a criação do respetivo conector de entrada Cyb SIEM, ao qual está associado um URL que é então utilizado na configuração do conector no lado do Elastic.

6. Testes às vulnerabilidades do sistema

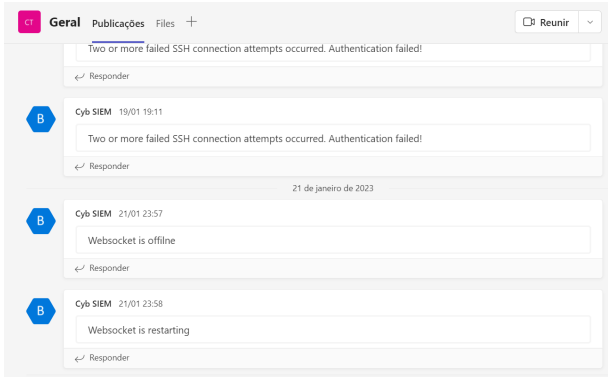
As vulnerabilidades são fraquezas que, quando exploradas por atacantes, podem comprometer o bom funcionamento do sistema. São realizados grande parte das vezes para proveito do atacante [28]. A presente secção visa descrever os testes realizados ao sistema apresentado neste documento. Os testes têm como base as vulnerabilidades descritas na Secção 4 que pretendem ser exploradas com ataques DoS, e de força bruta.



(a) Regra: Conector para o Discord



(b) Regra: Conector para o Email



(c) Regra: Conector para o Microsoft Teams

Figure 5: Conectores aplicados no SIEM para interação com aplicativos terceiros.

Hoje em dia os sistemas operativos estão a ter propósitos de uso, cada um tem as suas vantagens e isso faz com que seja mais usado para umas coisas do que para outras [35], assim, alguns são projetos para criar ataques cibernéticos ou apenas para procurar vulnerabilidades como é o caso do sistema operativo *Kali*. Por tanto para a concretização destes ataques foi criada uma máquina virtual com as seguintes características:

- **OS**: Kali GNU/Linux Rolling aarch64
- **Host**: VMware20,1 1
- **Kernel**: 6.0.0-kali3-arm64
- **Shell**: zsh 5.9
- **CPU**: (2)
- **GPU**: 00:0f.0 VMware Device 0406

- **Memory**: 977MiB / 1977MiB

6.1. Ataque DoS

A presente secção visa descrever como foram realizados os testes de negação de serviço (DoS) ao sistema apresentado neste documento. Este tipo de ataques tem se tornado cada vez mais um problema, ameaçando servidores web em particular [44]. Apesar de ser um mecanismo de ataque bastante conhecido é bastante utilizado devido à sua eficiência [22], que como anteriormente descrito na Secção 2.2 consiste em realizar uma grande quantidade de pedidos falsos a um destino ao qual este não está preparado para receber, assim, baixando a capacidade ou desativando os serviços por tempo indeterminado.

Como software para realizar o ataque foi usado *Slowloris*, este é gratuito e *open source*. É desenvolvido em python e permite fazer ataques de uma máquina para a outra usando o tráfico HTTP [37]. Pretende-se assim, com esta ferramenta gerar um grande volume de tráfico e tentar perceber como o servidor e os seus serviços se comportam.

Como resultado deste teste espera-se que o SIEM detete algum tipo de anomalia usando os seus mecanismos inteligentes, que seja visível na *dashboard* que monitoriza o tráfico da plataforma o aumento de acessos à mesma, espera-se também receber alertas com base na regra definida.

Após o teste, onde foram realizados três mil e seiscentos pedidos falhados à plataforma como é visível na Figura 6 as ações tomadas pelo servidor foram o desencadeamento de um alerta, a deteção de uma anomalia pelos mecanismos inteligentes de nível dezassete, cujo intervalo se encaixa entre zero e cem. Todos os serviços continuaram a funcionar em perfeitas condições o que significa que não ouve qualquer impacto após ou durante o ataque.

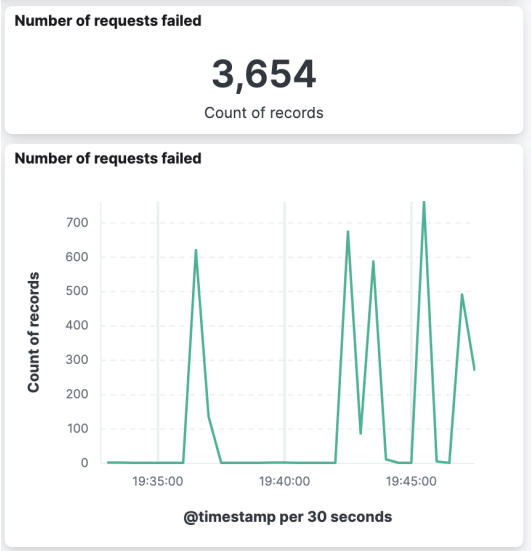
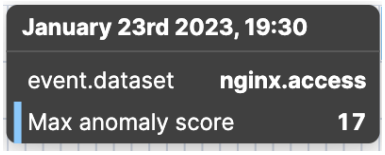
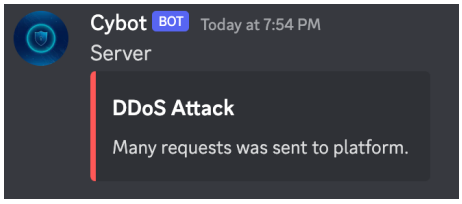


Figure 6: Visualização do fluxo de pedidos após ataque DoS em métrica e gráfico.

Em suma, com base nos resultados obtidos após o ataque DoS podemos assumir que este não teve qualquer efeito e que só com um número muito maior de pedidos é que poderá meter em causa o bom funcionamento da plataforma. Para contra medida poderia ser implementado um mecanismo que bloqueasse os pedidos da fonte quando um ataque destes ocorresse, outra medida poderia ser enviar uma resposta HTTP com o estado "Too Many Requests - 429" e controlar estes ataques a partir da API da plataforma.



(a) Anomalia detetada após ataque DoS



(b) Alerta enviado para o Discord após ataque DoS

Figure 7: Ações tomadas pelo SIEM após ataque DoS.

6.2. Ataque de força bruta à autenticação por SSH

A presente secção visa descrever como foi realizado o ataque de força bruta na autenticação por SSH ao servidor. Este tipo de ataques efetua consecutivamente tentativas de realizar autenticação por password em dado serviço ou recurso com o objetivo de identificar qual o utilizador e a *password* correspondestes para se apoderarem de dados confidenciais. O método de autenticação por *password* atualmente é o mais utilizado pelos serviços comuns [31], sendo considerado um dos mais seguros se a *password* corresponder tiver uma estrutura forte [32].

O software usado para realizar este ataque é o *Hydra*, este é uma ferramenta *open source* que permite executar vários tipos de ataques de força bruta usando listas de palavras. A distribuição de *Linux Kali* já vem com este software e com as listas de palavras para executar os ataques e encontrar *passwords* e *usernames* corretos.

Como resultado espera-se que o ataque de força bruta não seja bem sucedido, e que o sistema não seja comprometido. Espera-se ainda que o SIEM mostre as métricas correspondentes ao número de tentativas de autenticação por SSH, que detete anomalias com recurso a mecanismos inteligentes, e que desencadeie alertas para os conectores.

O ataque realizado foi feito apenas a um nome de utilizador, focando o ataque apenas na *password*. Como lista de *passwords* possíveis foi utilizada a lista que o sistema operativo oferecia que continha mais de quatorze milhões de *passwords* possíveis. Este tipo de ataque exige bastante esforço da máquina virtual o que torna impossível completar o teste em contexto académico. Ainda assim, em uma hora que o teste decorreu foram realizadas cinco mil e setecentas tentativas de autenticação.

Como resultado foi obtido os alertas desencadeados pela quantidade excessiva de tentativas de autenticação. Foi registada uma anomalia de um aumento de quatrocentas vezes mais o número de *logs* como demonstrado na Figura 8. Na *dashboard* correspondente é visível o aumento e número total de registos de autenticação.

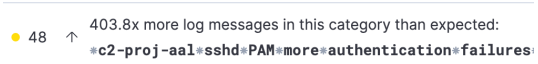


Figure 8: Anomalia detetada após ataque de força bruta de autenticação por SHH.

Em suma, pode-se assumir que o ataque de força bruta não teve sucesso em encontrar a *password* correspondente ao nome de utilizador. Porém, o servidor não tomou qualquer medida para que este ataque abrandasse ou parasse. Com isto, uma contra medida que podia ser implementada era por exemplo bloquear a fonte destes pedidos por um instante temporal variável consoante o número de pedidos falhados realizados até ao momento.

7. Conclusão

A solidão e isolamento dos idosos constitui um crescente problema na sociedade, o que pode causar preocupações em relação à sua segurança e bem-estar físico e mental. Deste modo, a plataforma SmartEmotion permite acompanhar o estado emocional de pessoas idosas de forma continua e em tempo-real, alertando os seus cuidadores sobre eventuais alterações negativas no seu estado emocional. A plataforma apresentada lida com PII e questões de saúde emocional bastante importantes, sendo assim muitíssimo relevante a garantir a segurança e privacidade dos dados, disponibilidade constante do sistema e proteção relativamente aos acessos ao sistema, nomeadamente à máquina virtual Ubuntu responsável por alojar a plataforma.

Um SIEM permite recolher, analisar e no geral monitorizar os diversos aspetos importantes mencionados anteriormente. Posto isto e face à análise realizada sobre alguns dos sistemas SIEM existentes (OSSIM, ELK Stack, OSSEC, Wazuh, Metron e MozDef), foram implementados mecanismos de monitorização, que analisam *logs* provenientes do sistema Ubuntu e até *logs* customizados, cujos dados resultantes podem ser observados nas respetivas *dashboards* criadas no SIEM escolhido: ELK Stack. Paralelamente às monitorizações contínuas foram definidos mecanismos de regras e alertas que efetuam o envio de mensagens personalizadas para três plataformas distintas: Microsoft Teams, Discord e Email.

Paralelamente à monitorização e posteriormente à análise das possíveis vulnerabilidades que a plataforma e o sistema Ubuntu podem sofrer, realizamos dois ataques distintos: BF com o objetivo de descobrir a *password* de acesso à máquina virtual Ubuntu por autenticação com SSH e DoS de modo a impedir o funcionamento da plataforma SmartEmotion. Apesar de não estarem implementados mecanismos de contramedidas para os ataques realizados, é de ressalvar a deteção das anomalias resultante e o envio de alertas através do SIEM mencionado a cima.

Em suma, com o presente trabalho tornou-se possível realizar a monitorização de todo um sistema através da utilização de um SIEM muito completo a nível de funcionalidades e recursos disponíveis. Deste modo, passando por diversas fases desde a recolha e transformação de diversos tipos de informações, análise a possíveis vulnerabilidades, monitorização e visualização do estado atual e histórico de serviços e recursos, e simulação de ataques às vulnerabilidades mais comuns.

References

[1] 6 Common Log File Formats | CrowdStrike.
[2] Apache Metron Big Data Security.
[3] Configuring Logging | NGINX Plus.
[4] Filebeat overview | Filebeat Reference [8.6] | Elastic.
[5] Kibana—your window into Elastic | Kibana Guide [8.6] | Elastic.
[6] Laravel - The PHP Framework For Web Artisans.
[7] Logging and analyzing slow queries in MySQL.
[8] Logstash Introduction | Logstash Reference [8.6] | Elastic.
[9] Module ngx_http_log_module.
[10] MySQL Log File Location.
[11] NGINX Access Logs and Error Logs | DigitalOcean.
[12] OSSEC - World's Most Widely Used Host Intrusion Detection System - HIDS.
[13] OSSIM: The Open Source SIEM | AlienVault.
[14] Overview — Mozilla Enterprise Defense Platform documentation.
[15] What is Elasticsearch? | Elasticsearch Guide [8.6] | Elastic.
[16] An intelligent algorithmically generated domain detection system. *Computers and Electrical Engineering*, 92, 6 2021.
[17] docker logs, January 2023.
[18] Srividya K. Bansal. Towards a semantic extract-transform-load (etl) framework for big data integration. *Proceedings - 2014 IEEE International Congress on Big Data, BigData Congress 2014*, pages 522–529, 9 2014.
[19] Daniel Berman. Filebeat vs. logstash - the evolution of a log shipper | logz.io.
[20] Bob Blakley, Tivoli Systems, Ellen McDermott, J P MorganChase, and Dan Geer. Information Security is Information Risk Management.
[21] Kelly C. Bourne. The Government Gets Involved. In *Application Administrators Handbook*, pages 377–386. Elsevier, 2014.
[22] Rodrigo Braga, Edjard Mota, and Alexandre Passito. Lightweight ddos flooding attack detection using nox/openflow. *Proceedings - Conference on Local Computer Networks, LCN*, pages 408–415, 2010.
[23] Staff Contributor. 10 Best Free and Open-Source SIEM Tools - DNSstuff, November 2019.
[24] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14:1007–1018, 2003.
[25] Jean-Christophe Dansereau. How to Check and Configure Ubuntu Logs, November 2022.
[26] Konark Truptiben Dave. Brute-force Attack "Seeking but Distressing". *International Journal of Innovations in Engineering and Technology*, 2(3), 2013.
[27] Department of Electrical Engineering, Udayana University, Jimbaran, Badung Regency 80361, Indonesia and I Putu Elba Duta Nugraha. A Review on the Role of Modern SOC in Cybersecurity Operations. *International Journal of Current Science Research and Review*, 04(05), May 2021.
[28] Derui Ding, Qing Long Han, Yang Xiang, Xiaohua Ge, and Xian Ming Zhang. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275:1674–1683, 1 2018.

[29] Dave Humphrey. How to meet privacy requirements with your PII.
[30] Almantas Kakareka. Chapter 1 - Detecting System Intrusions. In John R. Vacca, editor, *Network and System Security (Second Edition)*, pages 1–27. Syngress, Boston, January 2014.
[31] Tejaswi Kakarla, Aakif Mairaj, and Ahmad Y. Javaid. A real-world password cracking demonstration using open source tools for instructional use. *IEEE International Conference on Electro Information Technology*, 2018-May:387–391, 10 2018.
[32] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio López. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. *Proceedings - IEEE Symposium on Security and Privacy*, pages 523–537, 2012.
[33] Martin Fowler. *Patterns Of Enterprise Application Architecture By Martin Fowler*. 2012.
[34] Akash Padwal. How to Install Elastic Stack on Ubuntu 20.04 LTS, April 2021. Section: Elastic search.
[35] Paul. Everything you need to know about kali linux | edureka, 2023.
[36] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K. Rajamani, Janice Tsai, and Jeannette M. Wing. Bootstrapping privacy compliance in big data systems. *Proceedings - IEEE Symposium on Security and Privacy*, pages 327–342, 11 2014.
[37] Khundrakpam Johnson Singh and Tanmay De. Mlp-ga based algorithm to detect application layer ddos attack. *Journal of Information Security and Applications*, 36:145–153, 10 2017.
[38] Sue Ann Sisto and Kim Ratner. CHAPTER SIX - Evaluation. In Sue Ann Sisto, Erica Druin, and Martha Macht Sliwinski, editors, *Spinal Cord Injuries*, pages 121–152. Mosby, Saint Louis, January 2009.
[39] K. Sornalakshmi. Detection of dos attack and zero day threat with siem. *Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems, ICICCS 2017*, 2018-January:1–7, 7 2017.
[40] K.Muthamil Sudar, P. Deepalakshmi, P. Nagaraj, and V. Muneeswaran. Analysis of Cyberattacks and its Detection Mechanisms. In *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 12–16, November 2020.
[41] Muhammad Umar. LOG - Log File Format, June 2021.
[42] Wazuh. Wazuh · The Open Source Security Platform.
[43] Amakiri Welekwe. 7 best open source SIEM tools, February 2021.
[44] Yi Xie and Shun Zheng Yu. Monitoring the application-layer ddos sttacks for popular websites. *IEEE/ACM Transactions on Networking*, 17:15–25, 2009.
[45] Gaurav Yadav. NGINX Access & Error Logs Configuration: Logging Explained, May 2022.

Apêndice A. Código Python

Identifica qual o estado dos serviços em intervalos de trinta segundos e gera logs com base nesses estados.

```
#importing the module
import logging
import subprocess
import time

#now we will Create and configure logger
logging.basicConfig(filename="logger.log",
                                    format='%(asctime)s %(levelname)s %(message)s',
                datefmt='%b %d %H:%M:%S',
                                    filemode='w')
#Let us Create an object
logger=logging.getLogger()
while True:
    # pm2 websocket
    output = subprocess.getoutput("pm2 ls | grep online")
    if output == "":
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.INFO)
        #some messages to test
        logger.info("PM2:Websocket offline")
        subprocess.getoutput("pm2 start /var/www/project/websocket/index.js --watch --time")
        logger.info("PM2:Websocket restarting")
    else:
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.INFO)
        #some messages to test
        logger.info("PM2:Websocket online")
    # service php
    output = subprocess.getoutput("systemctl status php7.4-fpm | grep active")
    if output == "":
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.CRITICAL)
        #some messages to test
        logger.critical("systemctl:Php7.4-fpm offline")
    else:
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.INFO)
        #some messages to test
        logger.info("systemctl:Php7.4-fpm online")
    # service php
    output = subprocess.getoutput("systemctl status mysql | grep active")
    if output == "":
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.CRITICAL)
        #some messages to test
        logger.critical("systemctl:mysql offline")
    else:
        #Now we are going to Set the threshold of logger to DEBUG
        logger.setLevel(logging.INFO)
        #some messages to test
        logger.info("systemctl:mysql online")
    time.sleep(30)
```

Apêndice B. Dashboards

O ELK Stack inclui ferramentas que facilitam a criação das *dasdhboards* que permitem a visualização de informações e métricas recolhidas após a análise de *logs*. A presente secção visa mostrar as *dashboards* que foram desenvolvidas e estruturadas no desenvolvimento da solução apresentada. A primeira, representada pela Figura 3, mostra a informação das métricas do sistemas (*host*, CPU, disco, tráfico, SSH, estado dos *websockets*, estado do PHP, estado do MySQL). A segunda *dashboard*, correspondente à Figura 10, representa a informação das tentativas de conexão por SSH (tentativas de login, login com sucesso/insucesso). Na Figura 11 é possível visualizar as métricas da plataforma (pedidos GET e POST, fluxo da plataforma, métricas de login). A Figura 12 demonstra a visão global do Docker (containers, CPU, memória, *network* IO), tempo). A Figura 13 demonstra a visão global do MySQL (queries, logs de erro). A última dashboard, representada na Figura 14, é demonstrado as métricas relativas ao fluxo de acessos à base de dados da plataforma SmartEmotion (queries, *threads*, conexões, *buffer*, tráfico, cache, ficheiros, fluxos).

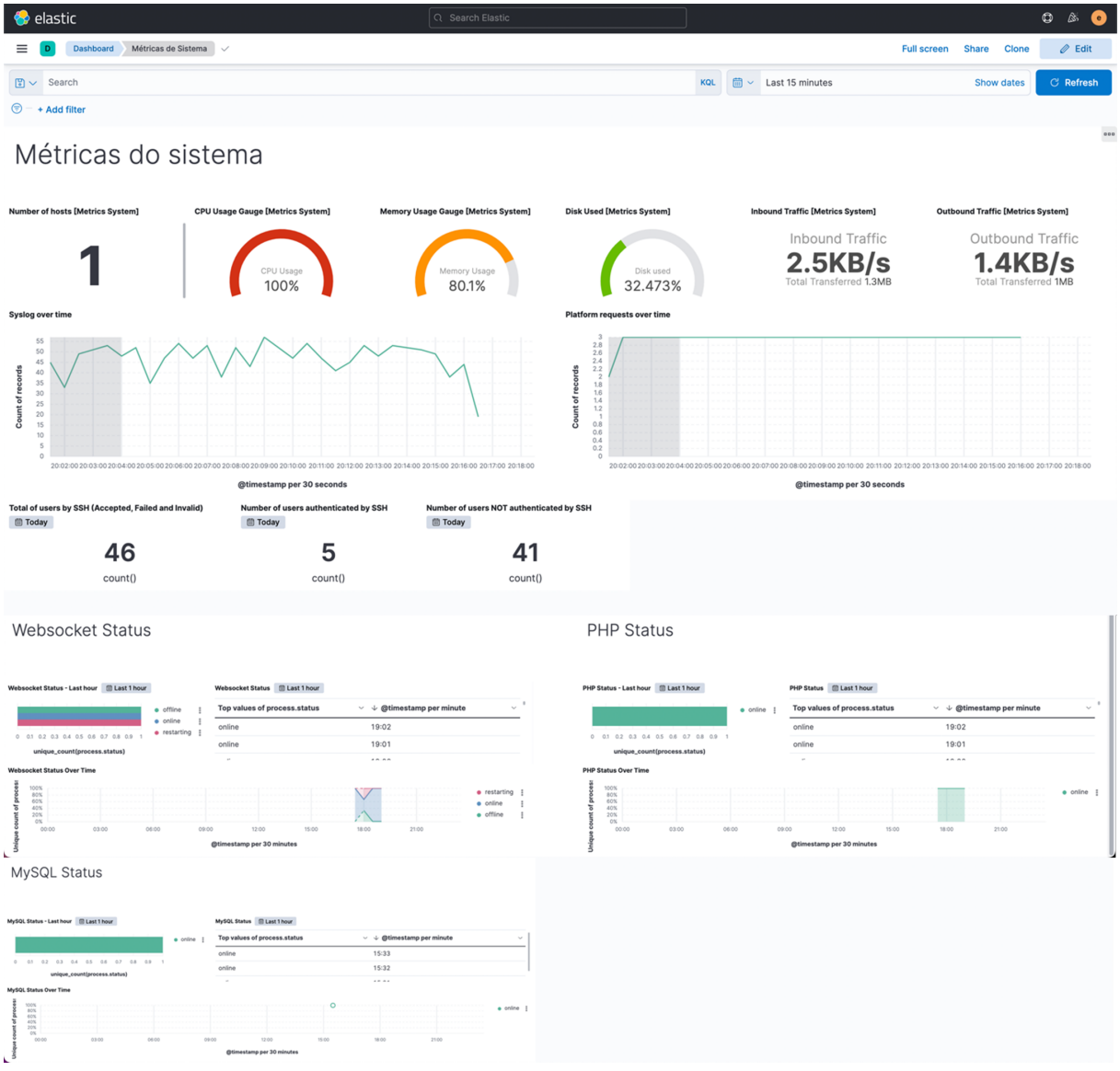


Figure 9: *Dashboard* - Métricas do sistema

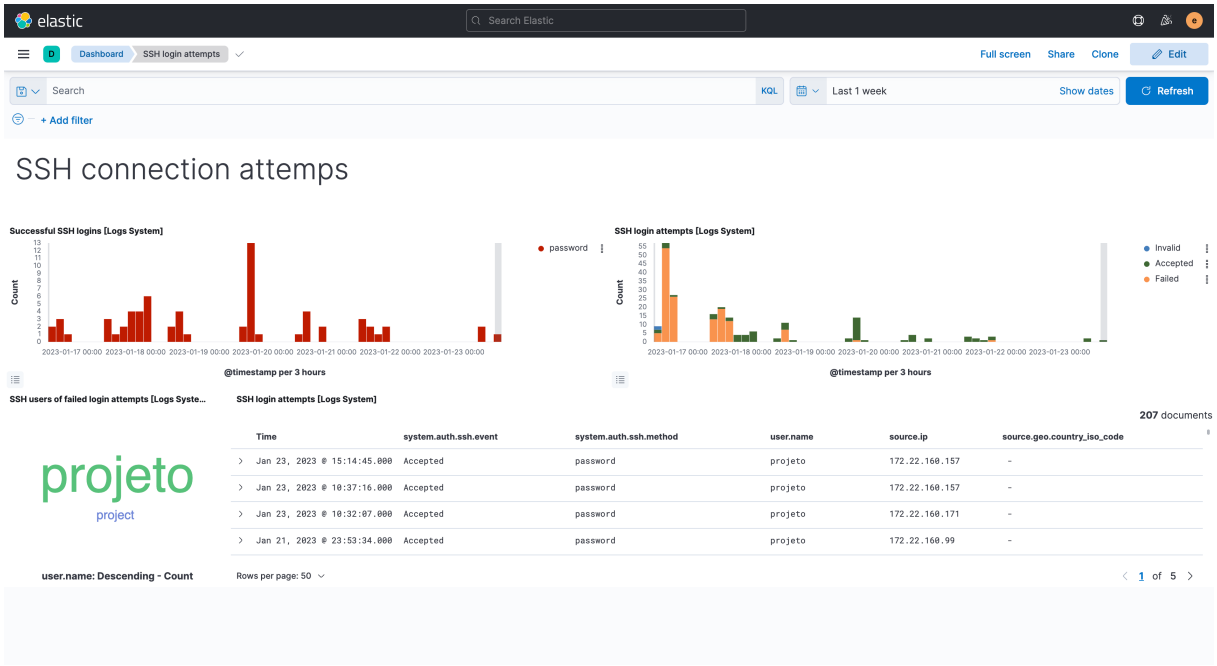


Figure 10: *Dashboard* - Tentativas de conexão por SSH

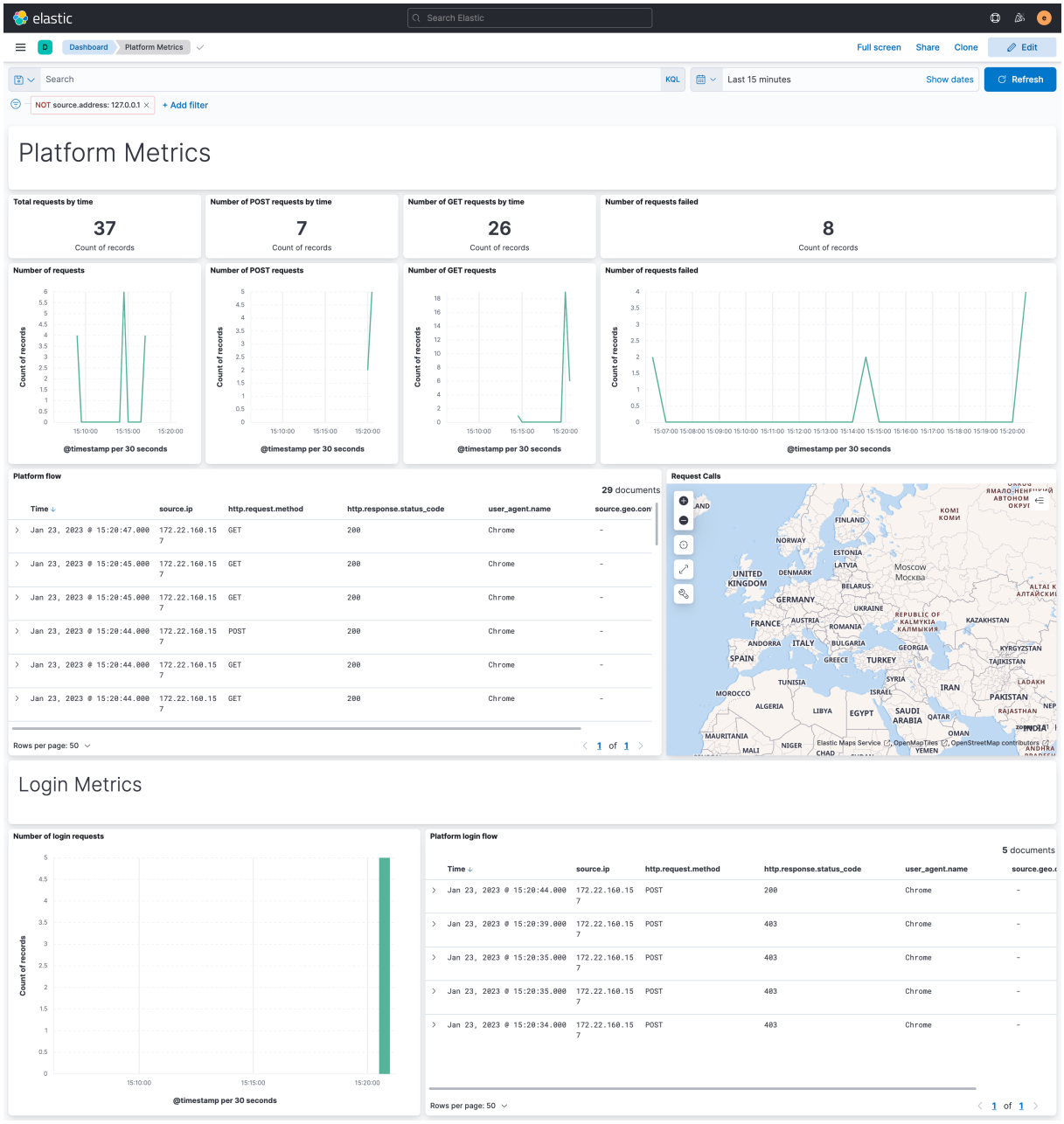


Figure 11: *Dashboard* - Métricas de plataforma

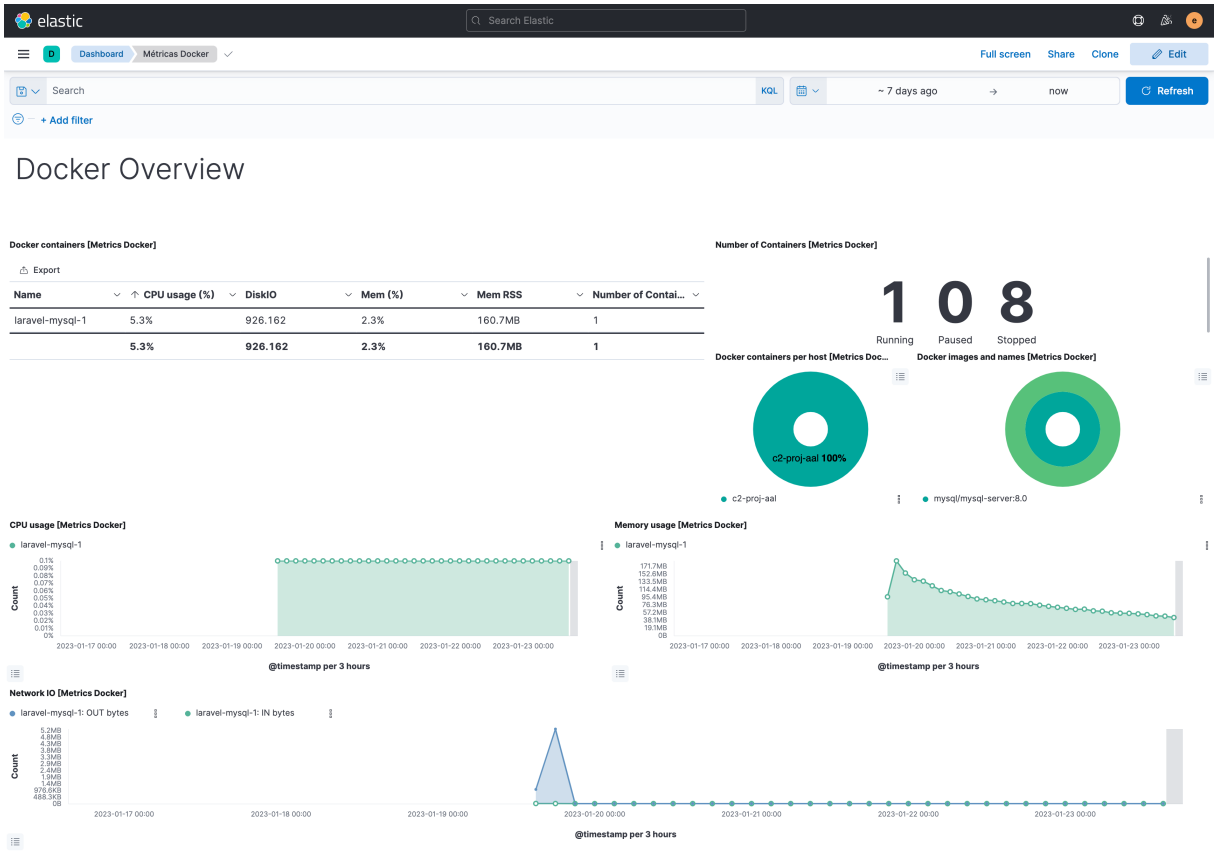


Figure 12: *Dashboard* - Docker Overview

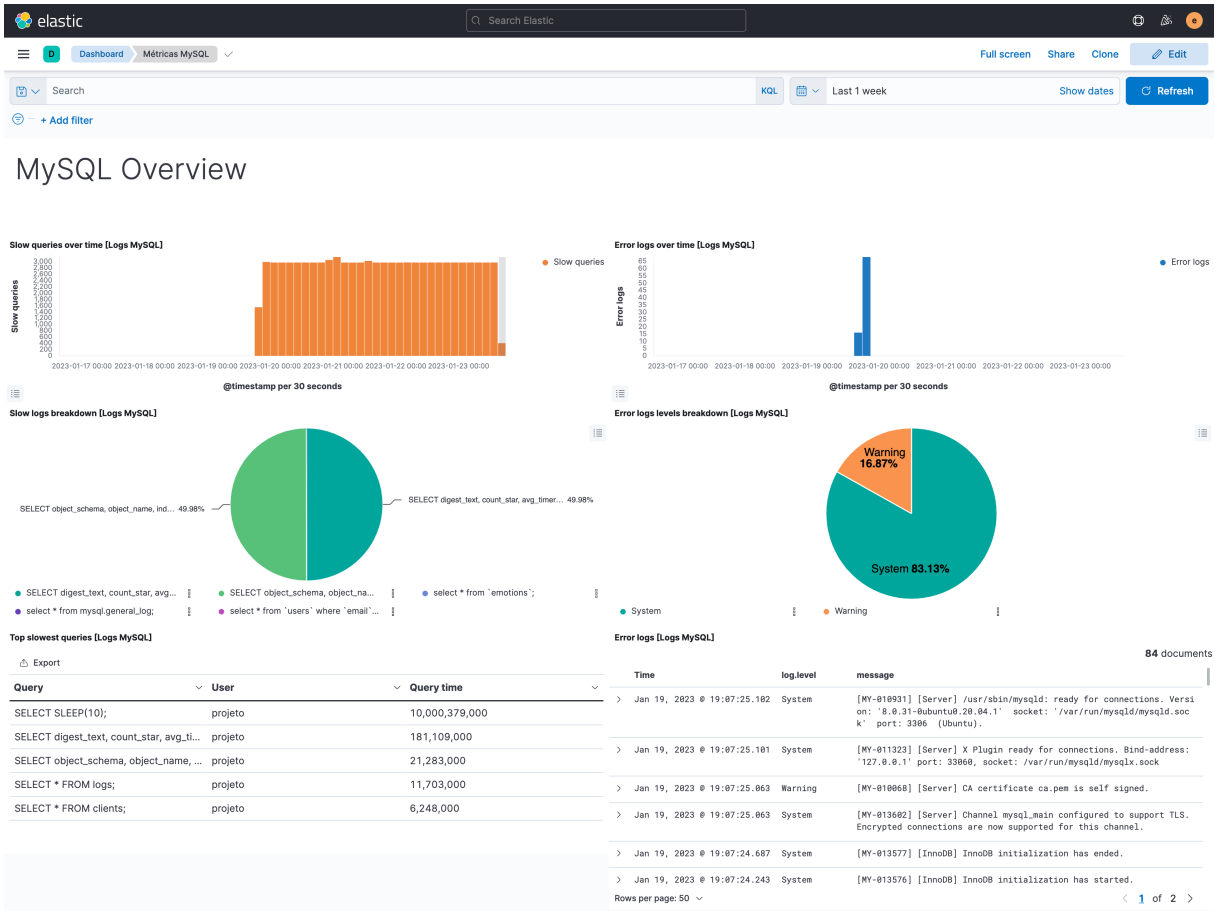


Figure 13: *Dashboard* - MySQL Overview

Figure 14: *Dashboard* - MySQL SmartEmotion Database