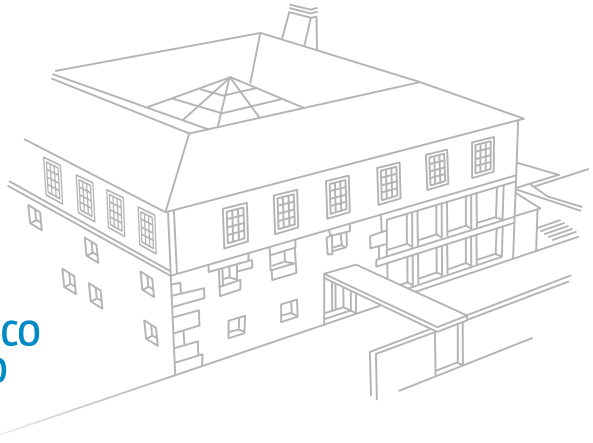

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Sonorização de eventos gerados por um SIEM

DESIGNAÇÃO DO MESTRADO
Engenharia Informática

AUTOR
Luís Manuel Magalhães de Sousa

ORIENTADOR(ES)
Prof. Doutor António Alberto dos Santos Pinto

ANO
2016

Sonorização de eventos gerados por um SIEM

Luís Sousa
Mestrado em Engenharia Informática
ESTG-IPP
8090228@estg.ipp.pt

4 de Janeiro de 2017

Agradecimentos

O desenvolvimento deste trabalho não ficaria completo sem agradecer a todos os que me ajudaram a tornar possível a realização do mesmo. Em primeiro lugar, quero agradecer ao meu orientador Professor Doutor António Pinto, pela sua disponibilidade, paciência, simpatia, ajuda e apoio na concretização deste projeto, sem a sua preciosa ajuda não seria possível. Os meus agradecimentos finais vão para a minha família, pelos incentivos ao longo da realização deste trabalho e sobretudo pela paciência, em especial aos meus pais, irmã e à minha avó que antes de nos deixar me tinha dito que iria conseguir, apoiando me sempre incondicionalmente.

Resumo

A informação gerada por sistemas de monitorização de redes de computadores é cada vez maior. A monitorização constante é imperativa mas muito difícil de realizar por várias razões. Em particular, existe alguma dificuldade em lidar com tanta informação em tempo real e de forma inteligível para o administrador da rede. Aplicações de Security Information Event Management, geram eventos correlacionados em função de ocorrências na rede. Estes eventos são classificados de acordo com a sua perigosidade. Uma aplicação que possibilite a sonorização dos eventos gerados por um Security Information Event Management, poderá facilitar o trabalho do administrador da rede, já que não necessitará de estar a consultar o serviço e bastará escutar o resultado da sonorização de tais eventos.

Palavras-chave: OSSIM, SIEM, Sonorização.

Abstract

The information generated by a network monitoring system is overwhelming. Monitoring is imperative but very difficult to accomplish due to several reasons. In particular, there is some difficulty in the handling of so much real-time information in a way that is intelligible for the network administrator. Security Information Event Management applications, generate events that correlate multiple occurrences on the network. These events are classified accordingly to their risk. An application that allows the sonification of events generated by a Security Information Event Management, can facilitate the work of the network administrator by avoiding the necessity of him constantly monitoring the service and allowing him to just listen to the result of the sonification of such events.

Keywords: OSSIM, SIEM, Sonification.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Acrónimos

APT: Advanced Persistent Threat
DDoS: Distributed denial-of-service
HIDS: Host Intrusion Detection System
IDE: Integrated Development Environment
IDS: Intrusion Detection System
IIS: Internet Information Services
InteNtion: Interactive Network Sonification
IPS: Intrusion Prevention System
IT: Information Technology
MAC: Media Access Control
MIDI: Musical Instrument Digital Interface
MySQL: My Structured Query Language
NIDS: Network Intrusion Detection System
NIPS: Network Intrusion Prevention System
OSC: Open Sound Control
OSSIM: Open Source Security Information Management.
Perl: Practical Extraction and Report Language
PRADS: Passive Real-time Asset Detection System
RUP: Rational Unified Process
SEM: Security Event Management.
SIEM: Security Information Event Management.
SIM: Security Information Management.
Snare: System Intrusion Analysis and Reporting Environment
SNMP: Simple Network Management Protocol
SSH: Secure Shell
SSL: Secure Sockets Layer
TCP: Transmission Control Protocol
TTL: Time To Live
UDP: Universal Datagram Protocol
USB: Universal Serial Bus

Capítulo 1

Introdução

A monitorização de uma rede para detetar intrusões, vulnerabilidades e ataques é uma tarefa necessária. O crescimento de uma rede de computadores, quer em número de equipamentos, quer na heterogeneidade de tais equipamentos, leva a que a monitorização de uma rede de computadores de tipologia média seja uma tarefa árdua e complexa. Todos os dias são desenvolvidas novas técnicas de ataque, descobertas novas vulnerabilidades, criados novos *malwares*, *spywares*, vírus, descobertas novas anomalias em protocolos, entre outros. *Hackers* conseguem quase sempre encontrar formas de penetrar numa rede ou num sistema, comprometendo o seu normal funcionamento. O mundo dos ataques e intrusões é um mundo sem fim.

O principal objetivo das redes da computadores e da Internet é permitir a troca de dados entre computadores, fornecendo um meio de comunicação entre pessoas, eventualmente dispersas geograficamente, de fácil acesso. A Internet também tem os seus contras. Se informação privilegiada estiver disponibilizada na Internet e não estiver devidamente protegida contra terceiros, pode facilmente ser acedida e usada por alguém mal intencionado, apagando-a ou tornando-a pública.

Esta situação torna-se crítica, quando uma empresa depende da sua infraestrutura de rede para funcionar. Nos dias de hoje, os equipamentos, servidores e serviços disponibilizados numa rede de computadores tende a ser o ativo de mais valor numa empresa, pois as redes de computadores são o núcleo das tecnologias de informação e da comunicação moderna. As redes de computadores tornam-se assim num dos componentes vitais de qualquer empresa de médio ou grande porte. As redes de computadores têm crescido, não só em tamanho e complexidade, mas também em significado e valor para as empresas. De um ponto de vista mais formal, uma rede de computadores, permite facilitar a comunicação, colaboração e transação de informação e ideias, fazendo a empresa crescer e desenvolver-se com mais facilidade.

A partir de um certo momento, em alguns casos, as rede de computadores, deixaram de ser uma parte fundamental da empresa e passaram a ser a própria empresa, devido a forma transversal com que estas integram a empresa. Empresas de comércio *online* ou que obtêm a totalidade dos seus lucros e objetivos através de um *web site*, são exemplo de empresas em que uma falha na rede, implica prejuízo. Sendo assim, existem certos serviços na rede de uma empresa que são críticos ao ponto de parar toda a empresa. Estes serviços devem ser os primeiros a ser protegidos, dada a sua importância para o funcionamento da empresa. Cada vez mais, nos momentos em que a rede se encontra inoperacional, um número significativo de operações do quotidiano deixam de ser realizáveis. Prejuízos causados pela inoperacionalidade da rede, ou pela perda de dados importantes sobre a empresa, poderão ser avultados. Mesmo um administrador de rede experiente, pouco poderá fazer se não tiver conhecimento do que se passa a cada

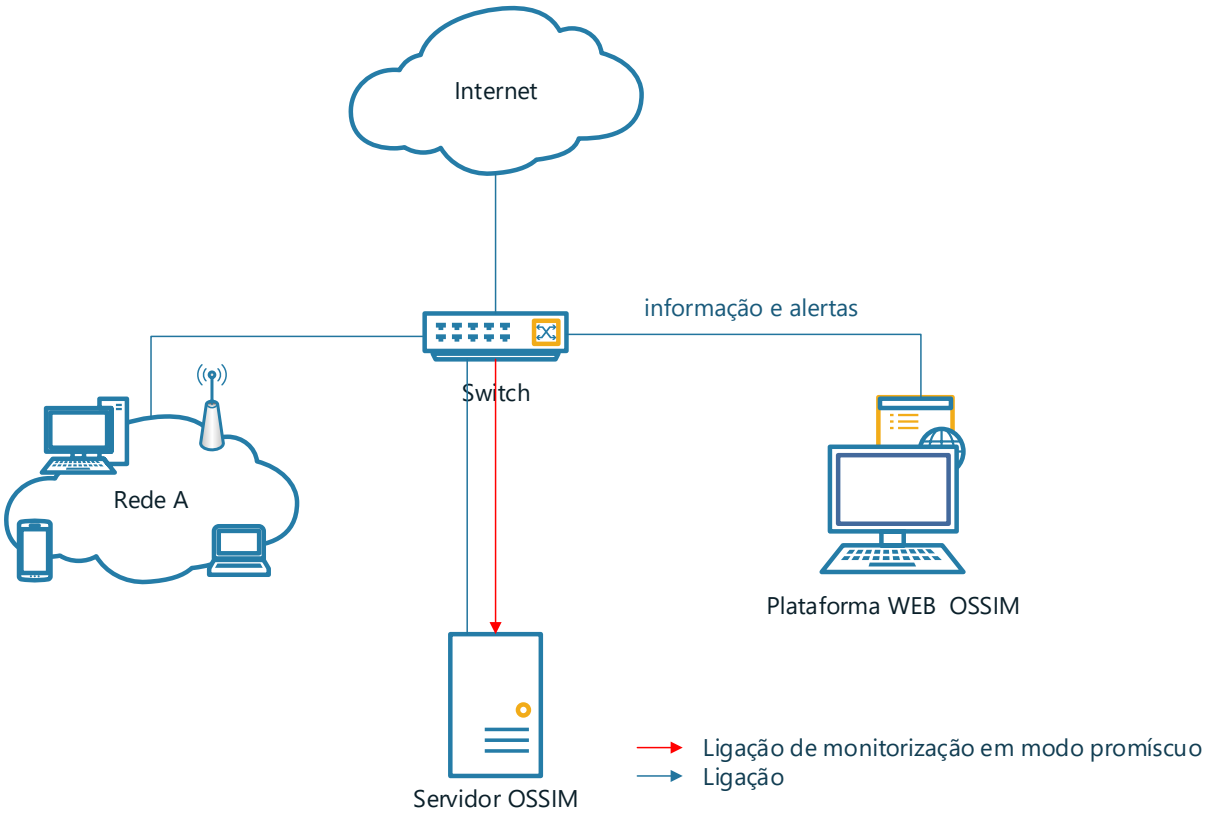

Figura 1.1: Diagrama básico de funcionamento do OSSIM.

momento na sua rede. Torna-se assim claro, que o cuidado com a monitorização da rede e a tentativa de previsão de problemas podem ser um dos melhores investimentos feitos por uma empresa.

Para combater estas dificuldades, é então necessário recorrer a certas aplicações ou plataformas que que possibilitem, de forma integrada, tanto a monitorização da rede como a geração de alertas em situações problemáticas. Existem várias aplicações ou plataformas *open source* que permitem ao administrador da rede ter toda a informação que necessita para monitorizar e detetar ataques na sua rede. Um exemplo recente é o Open Source Security Information Management (OSSIM) [3], que foi identificado como a solução *open source* mais viável e mais completa [6]. O OSSIM é um Security Information Event Management (SIEM) que permite ao administrador da rede, não só a monitorização da rede, como a geração e gestão de eventos de segurança. Informações principais sobre o estado da rede são apresentadas sobre a forma de um *dashboard*, onde a informação relevante é apresentada num só ecrã.

Na Figura 1.1, podemos ver como exemplo, um diagrama básico de funcionamento do OSSIM. Neste exemplo, todo o tráfego relativo à rede A, passa por um *switch*. Na rede A, estão ligados computadores, portáteis e telemóveis, entre os equipamentos a serem monitorizados. O servidor OSSIM, está ligado em modo promíscuo a uma porta do *switch*, permitindo ao OSSIM, processar todo o tráfego proveniente da rede A, e gerar os respetivos eventos. Esses eventos são processados e normalizados pelo servidor OSSIM, dando origem ou não a alertas e ou informação útil para o administrador de rede. Através de um *browser*, o administrador de rede, pode aceder á plataforma web do OSSIM e obter toda essa informação útil e reagir caso seja necessário.

Mesmo assim, a tarefa de um administrador de rede não deixa de ser complicada, já que, depen-

dendo um pouco do tamanho da rede e do nível de alerta configurado, o OSSIM poderá requerer atenção constante por parte de quem está a monitorizar a rede e a gerir os inúmeros eventos de segurança gerados por este. Facilmente se alcançam milhares de eventos diários que poderão ou não necessitar de atenção. Recentemente, vários investigadores [30, 47], têm procurado representações alternativas para a monitorização de redes e daí surgiram técnicas de sonorização (transformação de dados em música) que apresentam um conjunto de mais valias.

A sonorização de eventos poderá permitir que o administrador de rede não necessite de monitorizar constantemente o OSSIM, para retirar ilações sobre o que se esta a passar na rede. Assim, o administrador só teria de escutar a sonorização e reagir quando fosse mesmo necessário, podendo realizar outras tarefas em simultâneo, enquanto a situação da rede fosse estável. Ou seja, usaria todas as suas capacidades (auditivas e visuais) para facilitar a realização de várias tarefas ao mesmo tempo, sem comprometer a tarefa principal, a de monitorização da rede.

1.1 Objetivos do trabalho

O objetivo deste trabalho consiste na criação de uma forma alternativa de monitorização da rede que possibilite que o administrador da rede possa, em simultâneo com a monitorização da rede, efetuar as suas tarefas do dia-à-dia. A monitorização de uma rede é uma tarefa que deve ser realizada de forma contínua e sem distrações, afim de se perceber o que se está a passar na rede, para se detetar intrusões, vulnerabilidades e ataques prejudiciais ao funcionamento da mesma. Sendo assim, o trabalho em causa, tem como objetivo, criar uma aplicação que permita auxiliar o administrador na tarefa de monitorização da rede através da sonorização de eventos recolhidos pelo OSSIM. Deste modo, o administrador da rede só é alertado se for mesmo necessário, evitando estar com uma atenção exagerada na monitorização da rede, quando não existe nada a comprometer a mesma.

1.2 Resultados esperados

A solução proposta deverá ser capaz de:

1. **Recolha de alertas:** A proposta de solução deve recolher eventos de um servidor OSSIM, acedendo à sua base de dados relacional.

2. **Sonorização de alertas:** A proposta de solução deve sonorizar os eventos recolhidos.

3. **Multi-plataforma:** A proposta de solução deve sonorizar eventos provenientes de um SIEM, independentemente do sistema operativo em uso. Deverá funcionar em vários sistemas operativos como: Windows, Linux e Mac OS, podendo o administrador de rede instalar no seu computador de trabalho, independente do sistema operativo.

4. **Vários modos funcionamento:** A proposta de solução deve funcionar através de uma execução sem interface gráfica (linha de comandos) ou através de uma interface gráfica, permitindo ao utilizador escolher a forma como quer interagir com a aplicação.

5. **Auto-executável:** A proposta de solução deve funcionar imediatamente no arranque quando implementado num dispositivo próprio, num *Raspberry Pi* por exemplo.

1.3 Estrutura do relatório

O relatório está organizado em capítulos. O Capítulo 2 explica o que é um SIEM e quais são as suas vantagens. Fala ainda sobre o OSSIM e as suas principais funcionalidades. O Capítulo 3 explica em que consiste a sonorização e os resultados da sua aplicação em diversas áreas. Apresenta ainda alguns trabalhos sobre sonorização em redes de computadores. O Capítulo 4 descreve a solução proposta para o problema em causa. Apresenta todos os detalhes sobre a proposta de solução, o protótipo da mesma e ainda identifica os requisitos levantados. Expõe os diagramas que foram efetuados no desenvolvimento da proposta de solução. Exibe ainda a arquitetura e metodologia de trabalho adotada. O Capítulo 5 apresenta algumas imagens da aplicação MuSec em funcionamento. Descreve os testes de implementação da solução na ESTG, exibe e analisa os resultados obtidos. O Capítulo 6 faz uma conclusão sobre o trabalho realizado e apresenta o trabalho futuro.

Capítulo 2

Gestão de eventos de segurança da informação

Um SIEM, é uma aplicação que permite detetar eventos de segurança de informação em função de ocorrências na rede, equipamentos e aplicações, permitindo assim detetar ataques, vulnerabilidades e ou intrusões. Tais eventos, são classificados conforme o nível de alerta e de perigo, que apresentam para uma rede, para um equipamento, para a estabilidade de uma aplicação ou para um sistema operativo. O termo SIEM, primeiramente utilizado por Mark Nicolett e Amrit Williams em 2005 [12, 26], surge da combinação de Security Information Management (SIM) e Security Event Management (SEM). Os SIM, surgiram inicialmente como aplicações independentes. Adotados quando as organizações procuravam cumprir com certificações de segurança de informação [35]. Hoje em dia, plataformas SIM, podem ser integradas com soluções SIEM. Permite a recolha e gestão de *logs*, a partir de sistemas operativos, redes e dispositivos de segurança. Traduz os dados registados em formatos correlacionados e simplificados para o entendimento do utilizador, em que a informação é apresentada sobre a forma de relatórios abrangentes. São ideais também, para para a gestão de registos e armazenamento a longo prazo de grandes quantidades de *logs*, pois apresentam uma grande capacidade de compressão de informação. Existem no mercado várias soluções SIM: netForensics, nFX SIM One, Splunk, Symantec's Security Information Manager, LogLogic SIM, nFX SIM One, Trigeo SIM, entre outras. Já um SEM, analisa em tempo real dados de *logs*, para garantir resposta a qualquer incidente de segurança de informação. Os dados ou *logs*, podem ser recolhidos de dispositivos de segurança, redes, sistemas e até aplicações. O foco, recai sempre sobre a recolha e análise de dados relevantes, sob um prisma de segurança da informação. Os seguintes produtos, podem ser categorizados como SEM: ArcSight ESM, Sentinel Log Manager, Trustwave Security Management, Cisco MARS, SolarWinds Log and Event Manager, entre outros.

Um SIEM tem a capacidade de recolher e analisar informações. Geram relatórios sobre eventos detetados que incluem informação relativa à rede e dispositivos utilizados. A deteção de eventos assenta na correlação de informação com origem na rede, nos equipamentos de segurança perimétrica, nos sistemas operativos em uso, em bases de dados, em comportamento aplicacional, em *logs*, assinaturas de vírus e até em resumos criptográficos (MD5, SHA) de código malicioso. Após a deteção, este permitem a gestão do evento. Adicionalmente, também podem incluir procedimentos proativos de análise e pesquisa de vulnerabilidades. Na Figura 2.1, pode-se ver as principais atividades, objetivos e fontes de dados de um SIEM. Geralmente um SIEM, agrega informação de várias fontes como redes, aplicações, base de dados e outro tipo de armazenamentos. Depois é feita a gestão dessa informação, desses *logs*, priori-

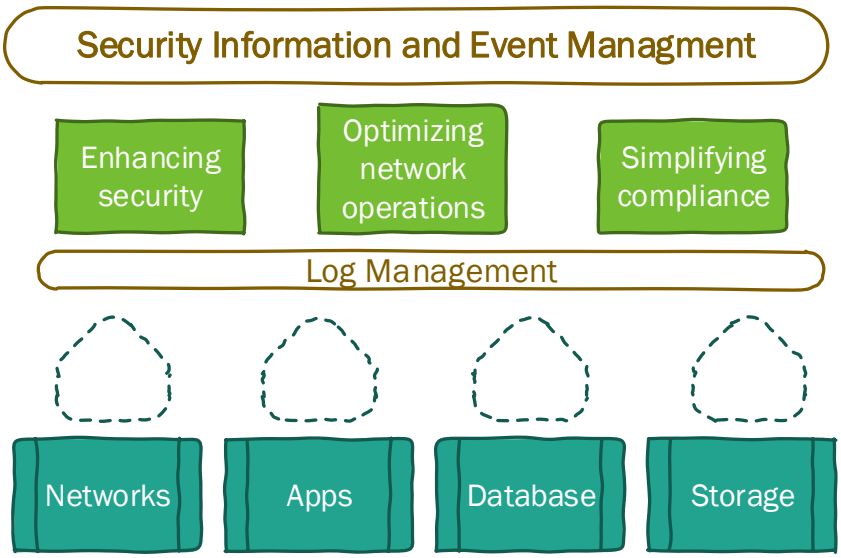

Figura 2.1: Principais atividades e recursos de um SIEM, adaptado de (Eva Kostrecová, 2015) [26].

zando os mesmos e gerando alarmes caso seja necessário. O objetivo, é aumentar a segurança, melhorar o funcionamento da rede e simplificar principalmente a vida do administrador da mesma.

O sucesso de um SIEM, depende sobretudo da qualidade da informação apresentada. Deverá ser usado em redes complexas, em que existe uma grande quantidade de informação proveniente das mesmas, não devendo ser utilizado em redes pequenas ou médias [12]. Um SIEM, acarreta custos de instalação (*hardware*), manutenção e de operação, relevantes. A sua instalação, requer uma análise aprofundada da rede. Após a sua instalação, a manutenção das bases de dados de ataques é importante para manter o sistema capaz de detetar eventos que usem *malware* mais recente. Uma equipa, para monitorizar e operar o SIEM, é também necessária, já que sem esta, o SIEM não realiza efetivamente o seu propósito. Um SIEM, requer equipamentos com capacidade de processamento significativos, embora dependa sempre da quantidade de eventos e de tráfego existente na rede. Tal equipamento, poderá demover a sua utilização em redes de pequena e média dimensão. Em 2012, haviam cerca de 85 aplicações SIEM, pagas e gratuitas [1].

2.1 Importância

Hoje em dia há cada vez mais informação e as empresas, em particular as de Information Technology (IT), estão a crescer cada vez mais, existindo uma grande variedade, complexidade e dificuldade na gestão das suas infraestruturas de rede. Isto faz com que as soluções SIEM ganhem um relevo extra [26].

O crescimento, aliado ao tamanho das empresas, leva a uma nova realidade e complexidade no mundo empresarial. Quanto maior é uma empresa, mais difícil é a gestão da mesma, quer ao nível da dimensão da equipa de IT, quer ao nível da gestão de informação entre os vários departamentos. As operações de IT são dispersas por diferentes grupos, por diferentes equipas e pessoas, umas com responsabilidades nos servidores, outras nas operações de rede, outras nas operações de segurança e outras no desenvolvimento aplicacional. Cada um desses departamentos usa as suas próprias ferramentas de monitorização, tal dificulta a partilha de informação e colaboração entre os vários departamentos quando ocorre algo inesperado. Ter toda esta informação num só sistema é uma grande mais valia para a organização. Um

dos principais objetivos de um SIEM é, precisamente, obter e concentrar a informação dos vários sistemas, analisando-a e disponibilizando-a, num único local. Tal permite, uma colaboração entre os vários departamentos, mais eficiente, nomeadamente em grandes empresas. O tamanho e complexidade das empresas, poderá levar à necessidade de adoção de um SIEM.

Novas falhas, que comprometem a fiabilidade de um equipamento de rede, de um *software*, ou até mesmo falhas de *hardware* que são exploradas, surgem diariamente, e em quantidades por vezes avassaladoras. Estas falhas, poderão levar à extração de informação importante de uma empresa, possivelmente comprometendo a sua credibilidade e operacionalidade.*Firewalls*, sistemas de deteção de intrusão, sistemas de prevenção de intrusão e soluções anti-vírus, permitem detetar atividades maliciosas, a partir de diferentes equipamentos numa rede. Contudo, a generalidade destas soluções não deteta ataques *zero day*, também conhecidos como ataques de dia zero. Estes ataques, são ataques rápidos que são exploram uma falha imediatamente após a sua descoberta e que ocorrem antes da comunidade de segurança, ou do fornecedor da aplicação, ou do fornecedor do equipamento, ter conhecimento da sua existência. Tal significa, incapacidade para reagir no imediato, não sendo capaz de a reparar ou evitar. Tais ataques, podem pôr em risco a empresa, dependendo sempre da exploração e do ataque realizado. Segundo um relatório recente disponibilizado por uma grande comunidade de especialistas em segurança de informação do *LinkedIn* [38], existe a necessidade de recorrer a melhores práticas e soluções de segurança. O mesmo concluiu que 74% das organizações são vulneráveis a ameaças internas, 56% dos profissionais de segurança afirmam que essas ameaças se tornaram mais frequentes no último ano e mais de 75% das organizações estimam que os custos de remediação dessas ameaças possam chegar aos 500.000 dólares. Este relatório afirma ainda que só 42% das organizações têm mecanismos apropriados para evitar ataques internos. Um SIEM tem um papel importante nesta prevenção, pois deteta atividades associadas a ataques, identificando também comportamentos anormais e não só assinaturas de ataques já conhecidos.

Uma investigação digital, eventualmente forense, é normalmente um processo longo e demorado que passa por várias etapas. Nestas etapas, incluem-se a recolha de evidências, correlação e análise das mesmas, para que se construa uma explicação para o evento em investigação. Em todo este processo, o investigador digital forense não só têm de interpretar os dados obtidos de *logs*, entre outros, para determinar o que realmente aconteceu, mas também deve prestar especial atenção ao processo de recolha de evidências, preservando-as de modo a que estas sejam admissíveis como prova em tribunal. Um SIEM permite que se efetuem investigações digitais forenses de forma rápida, completas e fidedignas. Os SIEM, armazenam e protegem registos e históricos, mas também fornecem ferramentas para visualizar e correlacionar dados, acelerando uma investigação deste tipo. Como os *logs* obtidos, representam as impressões digitais de toda a atividade que ocorre numa infraestrutura de informação, os mesmos podem ser extraídos e ajudar na segurança, operações e resolução de problemas de conformidade regulamentar. Sendo assim, um SIEM, é visto como um centro de recolha e de inteligência factual, com a capacidade de automatizar a monitorização de *logs*, correlacionar os mesmos, reconhecer padrões, produzir alertas e é bastante útil em investigações forenses.

Ameaças persistentes e avançadas, ou Advanced Persistent Threat (APT), têm recentemente, sido alvo de muitas noticias no mundo da segurança da informação e das organizações [1]. Muitos analistas na área, consideram que este tipo de ameaças, é responsável por introduzir falhas, em implementações do

[1]Um grupo russo disponibilizou dados médicos de vários atletas conhecidos, roubados da agência World Anti-Doping Agency (WADA)[41]. Vladimir Drinkman liderou uma campanha, chamada de Carbanak, atingindo mais de 100 bancos e instituições financeiras. As perdas chegaram a mil milhões de dólares. [42, 40].

algoritmo de criptografia RSA [9, 26]. APTs são caracterizadas, por serem ameaças cibernéticas, que visam a prática de espionagem pela Internet, recorrendo a técnicas sofisticadas de ataque e de recolha de informação. Tais informações, são por vezes, consideradas como extremamente valiosas. Usam uma combinação de métodos de ataque, dos simples aos avançados, para camuflar a sua existência e operação, visando também evitar a sua deteção. Para responder a tais ameaças, as organizações têm recorrido a *firewalls*, sistemas de deteção e prevenção de intrusos, autenticação por 2 fatores, *firewalls* internas, segmentação de rede, anti-vírus, entre outras [26]. Todas estas ferramentas, geram grandes quantidades de dados, o que dificulta a sua monitorização, gestão e interligação. A utilização de um SIEM, facilitará este aspeto, pois permitirá agrupar toda esta informação num único motor, proporcionando a capacidade de realizar uma monitorização contínua e correlacionar eventos em toda a amplitude e profundidade da empresa.

2.2 Módulos e funcionalidades

A lista de funcionalidades disponibilizada por um SIEM irá depender muito do seu fabricante e da sua arquitetura. Existe contudo um conjunto de funcionalidades que se encontram na generalidade dos SIEM. Em particular, as funcionalidades mais comuns são [12, 26, 35]: a agregação de dados, a correlação de eventos, a geração de alertas, a apresentação da informação, o reconhecimento de padrões, a reação a incidentes de segurança de informação, a retenção de *logs* e a geração de relatórios.

Um SIEM, deve ser capaz de agregar dados de diferentes fontes, desde redes de computadores, equipamentos de segurança perimétrica, servidores, base de dados e até aplicações importantes. Tais dados, devem ser apresentados de forma simples e consistente. A correlação de eventos, é uma característica fundamental, que deve estar presente num SIEM. Consiste, numa forma de tratamento de informação que interligue diferentes eventos, de diferentes partes físicas e ou elementos da rede. Com isto, após a execução técnicas de correlação, existe a capacidade de integração de diferentes fontes, com o objetivo de transformar os dados recolhidos, em informação útil.

Análise automática dos eventos correlacionados, deverá em caso de ataque, resultar em alertas. Alertas estes, que têm como a finalidade notificar o administrador da rede, ou pessoa responsável, de que algo anormal se está a passar na mesma, da existência de possíveis vulnerabilidades em equipamentos, da existência de intrusos ou da existência de outros problemas. Os alertas, devem ser categorizados, por exemplo numericamente, em função do risco, prioridade e outras características que os seus eventos apresentam para a rede.

As informações recolhidas e obtidas, por correlação de informação, devem ser apresentadas de forma concisa, preferencialmente num único ecrã. Devem ser apresentadas informações, de forma resumida, afim de permitir uma melhor compreensão por parte de quem está a usar o SIEM. As informações deverão ser agrupadas de acordo com o seu nível de risco, prioridade e data. As informações, deverão ainda, ser representadas textualmente e graficamente. Uma das particularidades da generalidade dos SIEM, é a maneira visual como a informação é apresentada ao utilizador. No artigo [32], são usadas técnicas e diferentes tecnologias de visualização, maioritariamente gráficos, que permitem uma melhor perceção e extensão, da informação que é disponibilizada por um SIEM.

Um SIEM deve, após ter identificado um ataque e de ter gerado o respetivo alerta, ser capaz de, em tempo real, gerar respostas automáticas previamente configuradas para cada caso. Toda a informação

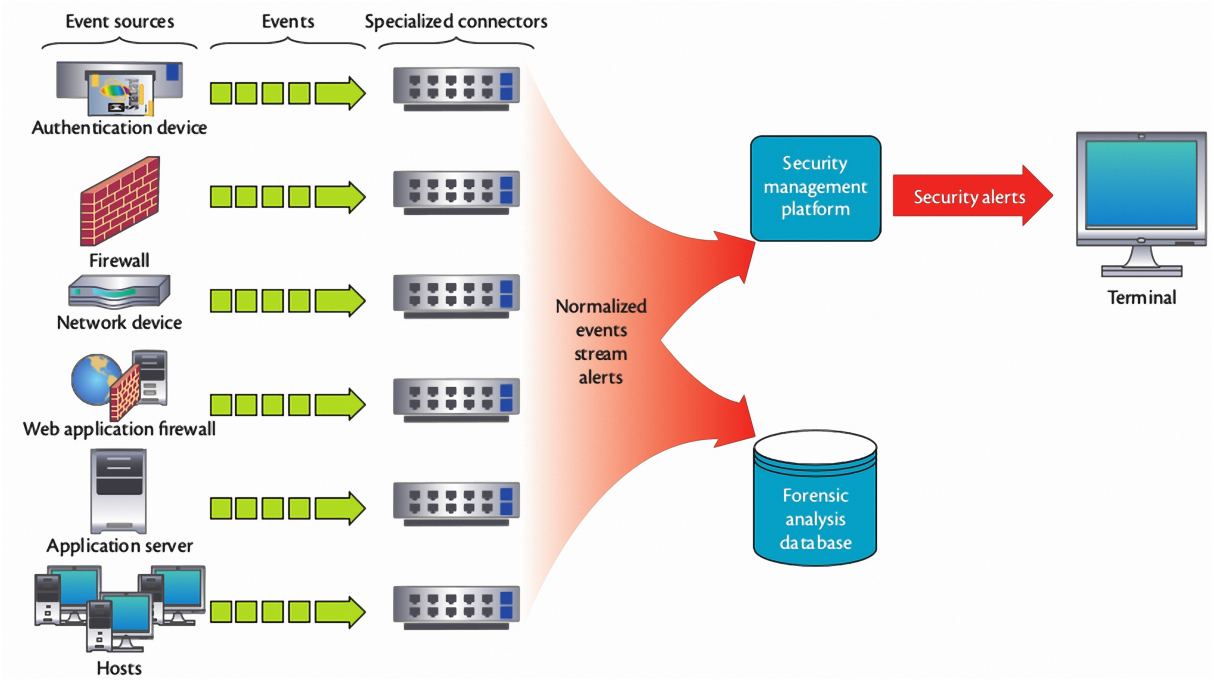

Figura 2.2: Arquitetura de um SIEM [8].

gerada, através da monitorização da rede, deve ser guardada, afim de ser possível consultar a mesma, em qualquer altura. Tal armazenamento facilita também a correlação dos dados, ao longo do tempo.

Finalmente, um SIEM, deve gerar relatórios sobre informação obtida e correlacionada, sobre os eventos detetados, sobre os alertas gerados e sobre as ações tomadas em resposta a incidentes de segurança de informação. Tal permite, a quem usa uma solução SIEM, perceber melhor situações críticas que se passaram na rede e tirar ilações de como as resolver e ou evitar.

2.3 Arquitetura SIEM

Uma análise às muitas soluções SIEM existentes no mercado, permite concluir que existe uma arquitetura genérica para os SIEM. Esta arquitetura genérica, é composta por vários componentes, sendo os mais comuns [35, 8]: dispositivos geradores de eventos, bases de dados de eventos, componentes de normalização de dados, componentes de gestão e componentes de monitorização.

Um SIEM, deve receber *logs* de várias fontes, onde se incluem as redes, dispositivos de segurança, sensores, *firewalls*, Intrusion Detection System (IDS), aplicações, aplicações *web*, servidores de autenticação e outros servidores. Um SIEM, deve assegurar a compatibilidade com a maior variedade de fontes de informação possível. Existe um grande número de fabricantes de equipamentos e *software*. Cada fabricante, adota a sintaxe de saída de dados que mais lhe convier. Tentar compatibilizar e abranger o maior número de equipamentos de rede e aplicações de segurança, é uma tarefa difícil e contínua.

A primeira tarefa de um SIEM, após obter os *logs*, é a normalização da informação. As diferentes representações de *logs*, obtidos de diferentes dispositivos e de diferentes fabricantes, leva à necessidade de conversão dos mesmos, para um formato comum. Só após a sua normalização, é que os dados prosseguem para outros componentes do SIEM. A normalização pode ser vista como a transformação de todos os *logs* obtidos de diversos equipamentos, num formato comum, utilizável pelo SIEM.

A plataforma de gestão de um SIEM, mantém e analisa os eventos. É este componente, que executa o motor de correlação baseado em regras. Tendo por base as regras existentes no SIEM e os eventos obtidos, são desencadeados alertas para o utilizador. Estes alertas, podem ser textuais ou representados graficamente, facilitando a sua compreensão por parte do utilizador. Uma regra, desencadeia um alerta, que é posteriormente apresentado num interface do próprio SIEM.

Os dados sobre os vários eventos, são guardados numa base de dados. Podem ser mais tarde, utilizados numa eventual investigação digital, para rever cenários e acontecimentos, para gerar relatórios de atividade, entre outras funções. Os SIEM, dispõem normalmente de um interface com o utilizador. Este interface é normalmente um interface *web*, que permite a monitorização do que está a acontecer na rede em tempo-real. O uso de *dashboards*, para facilitar a vida do utilizador e ou administrador de rede, é também habitual. Na Figura 2.2, podemos ver um exemplo de uma arquitetura genérica de um SIEM.

2.4 Soluções SIEM

As empresas estão cada vez mais a usar soluções SIEM para aumentar a segurança e a monitorização das suas redes [25]. Existem várias soluções, pagas e gratuitas. IBM QRadar, HP ArcSight, McAfee ESM, Splunk Enterprise, LogRhym, Cyberoam iView, OSSIM e ainda a solução Prelude, são boas opções, cada uma com as suas características, para monitorizar a rede de uma instituição [35, 27]. Mas existem, muito mais soluções SIEM no mercado. Existem alguns estudos credíveis, que visam apresentar a visão recente e futurista de soluções SIEM, ajudando empresas que se estão a inicializar nos SIEM, a analisar o mercado e escolher a sua solução. Um dos estudos, é realizado todos os anos, pela Gartner. A Gartner, é uma empresa que disponibiliza um leque considerável de análises, feitas sobre diversas áreas e tecnologias existentes no mercado [13]. Analisa as potencialidades de cada sistema e neste caso analisa todos os anos, soluções SIEM, utilizando uma metodologia chamada *Magic Quadrant* [24]. A metodologia *Magic Quadrant*, ou Quadrante Mágico, fornece um posicionamento gráfico competitivo de quatro tipos de fornecedores de tecnologia: Líderes, Visionários, Nichos de Mercado e Desafiadores. Segundo o relatório recente, realizado em Agosto de 2016, soluções comerciais como: IBM, Splunk, LogRhythm, HPE e Intel Security, são as soluções lideres do mercado. Na Figura 2.3, é possível ver o posicionamento dos quatro tipos de fornecedores de SIEM de 2016, segundo a Gartner. Dos gratuitos, a solução mais popular é o OSSIM [3]. Foi considerada pela Gartner como uma solução visionária que, segundo eles, ”oferece uma variedade de capacidades integradas de segurança, incluindo SIEM, monitorização da integridade de ficheiros, avaliação de vulnerabilidades, descoberta de ativos, e sistemas de deteção de intrusão baseados em *hosts* e na rede”. Foi ainda identificado como a solução *open source* mais viável e mais completa [6, 27].

Figura 2.3: Quadrante mágico aplicado à avaliação de soluções SIEM [24].

2.4.1 OSSIM

O OSSIM, é uma plataforma unificada desenvolvida pela AlienVault, grátis, de código aberto e baseada no sistema operativo Debian [2]. O OSSIM, atualmente encontra-se na versão 5.3.2 e é composto por quatro componentes principais [2, 19]: o sensor, o servidor, a interface *web* e a base de dados. A Figura 2.4, mostra a arquitetura do OSSIM.

O Sensor, que inclui um *rsyslog* e um agente OSSIM, recebe *logs* dos dispositivos da rede e guarda-os localmente através do *rsyslog*. O agente OSSIM, através de *plugins* próprios, analisa e normaliza cada *log* e envia-os para o servidor. O Servidor, desempenha as funções essenciais do SIEM, desde avaliação de risco, agregação e correlação de eventos recebidos do sensor, bem como guarda estes eventos na base de dados. A *Framework* fornece a interface *web* que permite a administração do OSSIM, liga e faz a gestão dos componentes e das ferramentas de segurança que o constituem. A Base de dados My Structured Query Language (MySQL), é utilizada para armazenar eventos, a configuração do sistema e toda a informação necessária à plataforma web do mesmo.

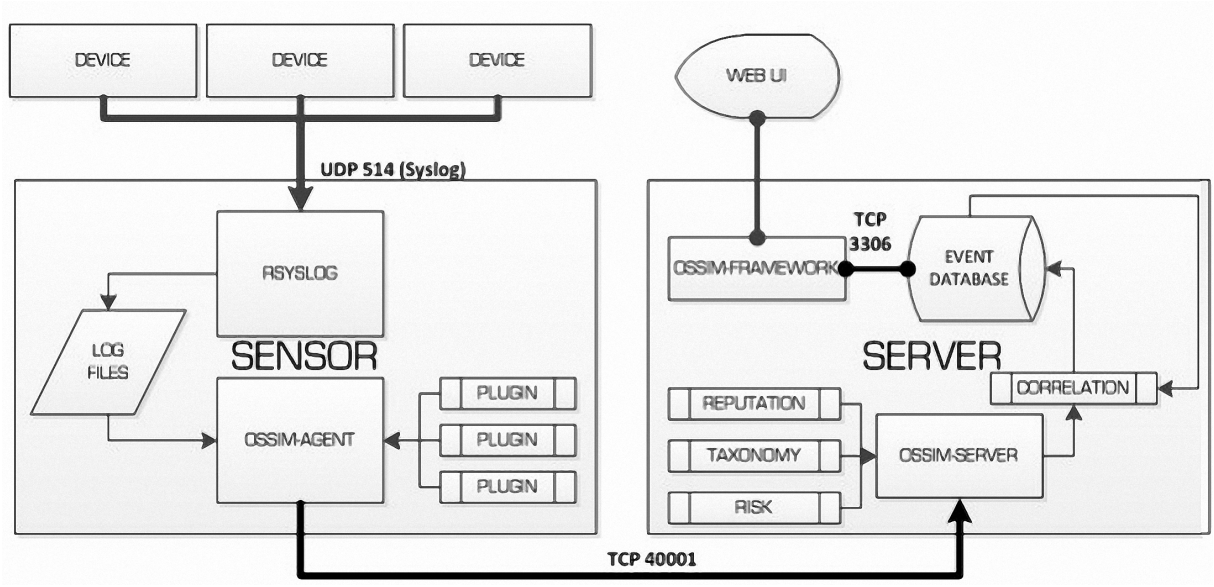

Figura 2.4: Arquitetura do OSSIM [2].

As funcionalidades principais do OSSIM [2], são a coleção e normalização de *logs*, a priorização de eventos e avaliação de risco, a análise e correlação de eventos, a geração de alarmes e ações de resposta e a análise de vulnerabilidades, deteção de intrusão e monitorização de redes. Na coleção e normalização de *logs*, todos os eventos que são capturados da rede, são guardados e depois analisados e normalizados.

Na priorização de eventos e avaliação de riscos, o servidor atribui valores de prioridade para os eventos registados. Permite assim saber o perigo/risco de um determinado evento, com a finalidade de alertar o utilizador. O risco de um evento, é calculado em tempo real através da fórmula [4]:

$$risco = (valor * prioridade * confiabilidade)/25$$

O *valor*, refere-se ao nível de importância (entre 0 e 5), da máquina que gerou o evento. É atribuído manualmente, por quem configurou o OSSIM. Caso não tenha sido atribuído, terá por omissão o valor 2. A *prioridade*, refere-se à importância do evento em si. É uma medida, que é usada para determinar o impacto que um evento, poderia ter na nossa rede. Situa-se entre 0 (sem prioridade) e 5 (elevada prioridade). A *confiabilidade*, é um valor (entre 0 e 10), inerente há certeza de um ataque ser real ou não. O OSSIM usa o valor 0 para falsos positivos e o valor 10 para sinalizar um ataque real. Como resultado da fórmula, o risco pode assumir valores entre 0 e 10, com a classificação: 0-2 (baixo), 3-4 (precaução), 5-6 (elevado), 7-8 (alto) e por fim 9-10 (muito alto).

Na análise e correlação de eventos, os eventos são analisados e relacionados entre si, para detetar possíveis ataques e anomalias. Um evento ou um conjunto de eventos, sob determinadas condições, geram alarmes. Os alarmes, podem ser vistos no interface *web* do OSSIM. O OSSIM, pode ser configurado também, para enviar alertas por *email* para o administrador do sistema ou para executar determinados *scripts*.

O OSSIM, implementa muitas das suas funcionalidades recorrendo a ferramentas *open source* populares. O *OpenVAS*, é uma *framework* poderosa para a deteção de vulnerabilidades. É considerada a ferramenta de código aberto, mais avançada na gestão e procura de vulnerabilidades. OpenVAS, é desenvolvido pela empresa Greenbone Networks GmbH, e encontra-se atualmente na versão 8.0. É

constituída, por um conjunto de vários serviços e ferramentas, que oferecem uma solução abrangente. Possui uma poderosa varredura de vulnerabilidades e gestão das mesmas. Permite descobrir e procurar vulnerabilidades, em múltiplos *hosts*, de forma concorrente.

O *Passive Real-time Asset Detection System (PRADS)*, é usado para identificar *hosts* e serviços, monitorizando o tráfego de rede de uma forma passiva. Reúne informações sobre *hosts* e serviços que vê na rede. A informação recolhida é usada para mapear a rede em causa, permitindo saber quais os serviços e *hosts* que estão "vivos" e em uso. Pode ser usado por um IDS para correlacionar eventos de um *host* ou serviço.

O *Nessus* é usado para detetar vulnerabilidades em equipamentos ligados à rede. É uma das ferramentas mais populares e com grande capacidade para a procura de vulnerabilidades. É desenvolvido pela empresa Tenable Network Security. Inicialmente, era uma aplicação grátis e de código aberto, mas em 2005 fecharam o código e em 2008 passaram a existir versões experimentais da mesma. Atualmente, existe uma versão grátis, denominada Nessus Home, mas é uma versão limitada e só é licenciada para ser usada na rede de casa. É usada por mais de 75.000 organizações em todo o mundo. É constantemente atualizada, contendo atualmente cerca de 70.000 *plugins*. O seu foco principal é a verificação de segurança em autenticações remotas ou locais, possuí uma arquitetura cliente-servidor com uma interface gráfica. Permite ainda que utilizadores escrevam os seus próprios *plugins*. Tem suporte para linguagens de *script*, para a procura de vulnerabilidades num número ilimitado de *IPs*, para a procura de vulnerabilidades em aplicações web, para exportar relatórios, para enviar notificações por email e ainda permite a programação da verificação de vulnerabilidades numa determinada hora, entre outras funcionalidades.

O *Nmap* é uma aplicação escrita em C++ por Gordon Lyon. É usada maioritariamente para analisar uma rede e encontrar portas abertas. A primeira versão surgiu em 1997 e atualmente encontra-se na versão 7.30. É uma ferramenta poderosa, gratuita e de código aberto. Permite identificar *hosts* disponíveis na rede, que serviços (nome de aplicações e suas versões) estão a ser usadas por esses *hosts*, que sistemas operativos usam, que tipo de *firewall* estão a usar, que tipo de dispositivos são e quais são os seus endereços Media Access Control (MAC), entre outras características. Foi desenvolvido com o objetivo de ser rápido na deteção em redes grandes e complexas. É flexível, portátil, funciona em vários sistemas operativos como: Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD, Sun OS, Amiga. Bem documentado e acima de tudo popular, conhecido e usado por muitos administradores de rede.

O *Snort* é um Network Intrusion Detection System (NIDS)/Network Intrusion Prevention System (NIPS) atualmente desenvolvido e mantido pela empresa SourceFire. É usado essencialmente para detetar intrusos na rede. É um software livre, com a capacidade de analisar o tráfego de rede em tempo real, de analisar protocolos e de detetar vários ataques como: *buffer overflows*, *stealth port scans*, ataques *Common Gateway Interface*, *SMB probes*, *OS fingerprinting*, entre outros. Contém boa documentação e bastantes regras de deteção de intrusos. Opera em modo *single thread*. Atualmente encontra-se na versão 2.9.83 e é compatível com sistemas Linux (Red Hat, Debian, Centos, Fedora, Slackware, Mandrake, etc.), OpenBSD, FreeBSD, NetBSD, Solaris, SunOS, HP-UX, AIX, IRIX, Tru64, Mac OS X e Windows.

O *Suricata* é usado para detetar intrusos na rede e é o NIDS/NIPS por omissão do OSSIM (a partir da versão 5). É um *software* de código aberto, escrito em C e desenvolvido pela empresa Open Information Security Foundation. Atualmente encontra-se na versão 3 e é compatível com sistemas FreeBSD,

OpenBSD, Linux, Mac OS X e Windows. A Alienvault decidiu deixar de dar suporte ao Snort em detrimento do Suricata pois, segundo eles, o Suricata apresenta vantagens sobre o mesmo. Em particular, apresenta melhor performance, já que opera em modo *multi-thread*. O Suricata, deteta cerca de 20% a 30% mais ameaças, usando as mesmas regras de deteção, que o Snort. Fornece um melhor visibilidade do tráfego, permitindo detetar melhor conteúdos maliciosos. É rápido na análise de tráfego HTTP. Deteta protocolos usados em portas não padrão, tem suporte para IPv6 e ainda possibilita a análise *offline* de ficheiros *pcap*.

O *TCPTrack* possibilita a monitorização de ligações Transmission Control Protocol (TCP). É um *software* utilizado para monitorizar ligações de rede, baseado no *tcpdump*. Opera em linha de comandos e permite, por exemplo, mostrar o consumo de tráfego IP em tempo real, mostrar o IP do cliente e a porta de origem, o IP do servidor e a porta de destino, o estado da ligação (”estabelecida”, ”fechada”, etc), tempo de ligação, tempo de inatividade da ligação e ainda a produção média de dados nas ligações existentes (largura de banda). É gratuito e de código aberto, a ultima versão (1.2.0) foi lançada em 2006, e funciona em sistemas operativos Linux e Mac OS X.

O *Nagios* é uma ferramenta popular de monitorização de redes. Foi desenvolvida por Ethan Galstad e uma basta comunidade de programadores. É uma ferramenta de código aberto. Atualmente encontra-se na versão 4.2.1. Com esta ferramenta escrita em C, é possível monitorizar *hosts* e seus serviços, alertando o utilizador quando ocorrem problemas. Desenhada para ser flexível e escalável, corre nativamente em sistemas Linux e Unix. Consegue monitorizar serviços de rede (SMTP, POP3, HTTP, etc), monitorizar remotamente via Secure Shell (SSH), permite que os utilizadores criem os seus próprios *plugins*, suporta deteção em paralelo e ainda notificação de alertas por email. A informação apresentada sobre computadores ou equipamentos inclui a percentagem de uso do processador, memória e disco, bem como os *logs* de sistema. Tem boa documentação e um interface *web* para a visualização de toda a informação, que esta consegue angariar.

O *NetFlow* é usado para mostrar o tráfego da rede e a largura de banda ocupada. Atualmente encontra-se na versão 5. Desenhado e publicado pela Cisco Systems, permite visualizar os fluxos de uma rede. A sua função é ajudar um administrador de rede a verificar o tráfego de entrada e de saída na mesma. Permite verificar o endereço IP de origem, o endereço IP de destino, o protocolo usado, porta de origem, porta de destino, o tipo de serviço, os *timestamps* do fluxo, numero de bytes, total pacotes verificados no fluxo, pacotes por segundo, bits por segundo, média de bits por segundo e ainda a duração do fluxo. Permite assim ao administrador conhecer o desempenho da rede, avaliar o impacto das aplicações na rede, otimizar a largura de banda, detetar tráfego não autorizado e resolver incidentes com mais rapidez.

O *Osiris* é um Host Intrusion Detection System (HIDS) que periodicamente monitoriza um ou mais *hosts*. Escrito maioritariamente na linguagem C. Tem suporte para uma gestão centralizada e adota uma arquitetura cliente-servidor. Mantém informação detalhada sobre mudanças no sistemas, utilizadores, grupos, módulos do *kernel*. Informação esta que pode ser enviada por email para o administrador da rede ou pode ser mantida para possíveis operações de análise forense. Mantém o administrador informado de possíveis ataques. O foco é dado a mudanças que indiquem que um sistema foi invadido ou comprometido. Pode ser utilizado para monitorizar *hosts* Linux, Unix, Mac OS X e Windows.

O *OSSEC* é outro HIDS de código aberto, passível de utilização no OSSIM que também permite a deteção de intrusos em *hosts*. Atualmente encontra-se na versão 2.8. Suporta a análise de *logs*, deteção

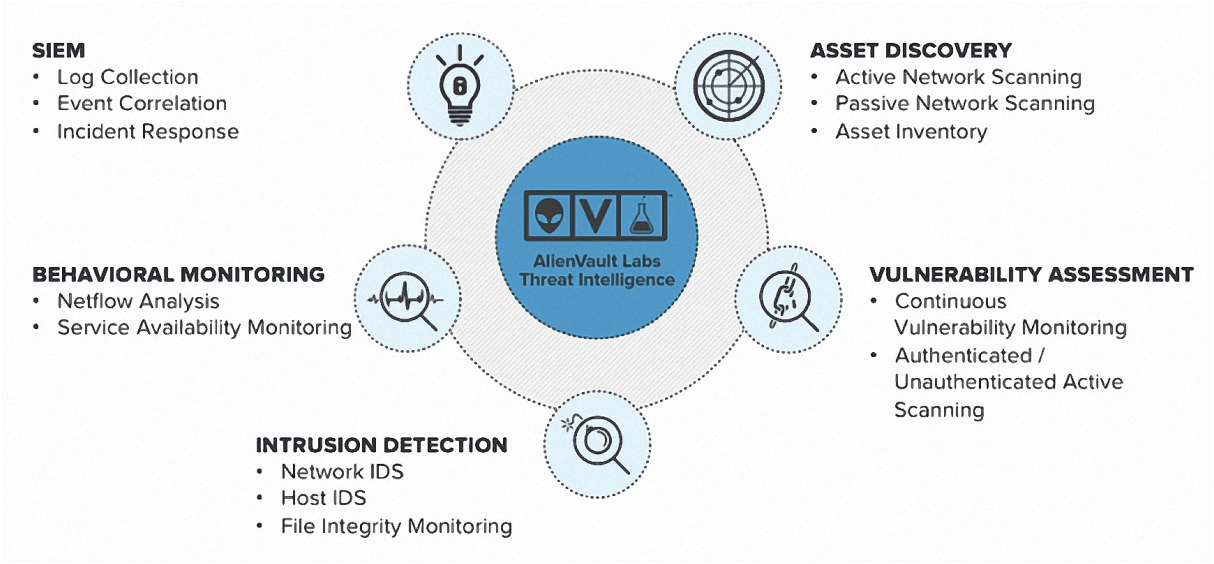

Figura 2.5: Principais funcionalidades do OSSIM (Alienvault, 2015).

de integridade do sistema e de ficheiros de *logs*, deteção de *rootkits*, monitorização de processos, entre outras funcionalidades. É um sistema de resposta ativa já que, após detetar uma ameaça, pode responder às mesmas usando para tal uma lista negra de IPs. Quando os ataques acontecem, o OSSEC alerta o administrador por email. O OSSEC também pode exportar alertas para qualquer sistema SIEM, via *syslog*, para que o administrador possa obter análises em tempo real e introspeções sobre os eventos de segurança do sistema. Funciona em múltiplas plataformas: Linux, Solaris, AIX, HP-UX, BSD, Windows Mac e ainda VMware ESX.

O *System Intrusion Analysis and Reporting Environment (Snare)* é composto por um conjunto de agentes mantidos pela empresa Intersect Alliance. Atualmente encontram-se na versão 4.0.2.0. Estes agentes permitem juntar dados de uma grande variedade de sistemas operativos e aplicações, centralizando-os e facilitando assim a sua análise. Permite colecionar *logs* de sistemas operativos Unix, Linux, Mac OS X, Windows, Solaris e aplicações como Microsoft SQL Server, vários *logs* aplicacionais em formato txt e é ainda compatível com Apache e servidores Internet Information Services (IIS). Os agentes Snare são leves, já que apenas são necessários 20 MB de memória para que estes executem. Foram projetados, para colecionar dados de *logs* de várias fontes e os enviar rapidamente para um ou mais servidores, para serem arquivados.

A principal vantagem do OSSIM, é a centralização da informação. Aproveita a diversidade das ferramentas antes mencionadas, concentra-as e permite a sua utilização conjunta e coesa por intermédio de um único interface *web*. Na Figura 2.5 é possível ver as principais funcionalidades da plataforma OSSIM.

2.5 Conclusão

Um SIEM, é uma plataforma unificada que permite monitorizar a rede de uma instituição, os seus equipamentos, os seus *firewalls*, os seus anti-vírus, os seus processos de sistemas operativos, bem como outros equipamentos e softwares. Permite, monitorizar diversas fontes de dados e alertar o utilizador para eventuais vulnerabilidades, ataques e intrusões. Os alertas, são apresentados num interface *web* do SIEM, sob

a forma de um *dashboard*. São também apresentados gráficos e relatórios, para serem consultados pelo administrador da rede. Muitas empresas, já começaram a implementar soluções SIEM nas suas redes, com a finalidade de possibilitar a gestão de eventos de segurança de informação.

A implementação de um SIEM, é aconselhada para empresas que dependam da rede para funcionar, com redes complexas ou de grande dimensão. O OSSIM é um SIEM gratuito. O OSSIM, contêm o conjunto de funcionalidades que um SIEM deve ter, permitindo ao administrador de rede, monitorizar a rede sem precisar de outras ferramentas para tal. A arquitetura do OSSIM, foi descrita, bem como o seu funcionamento e a forma como este calcula e categoriza, através de uma fórmula matemática, o perigo associado aos eventos recolhidos da rede, dos seus equipamentos e de softwares de segurança. Estes eventos, depois de categorizados, são apresentados ao administrador de rede, para o mesmo possa perceber o que se está a passar e reagir caso seja necessário.

Capítulo 3

Sonorização

Sonorização é uma forma de transformar dados e seus relacionamentos num sinal acústico para fins de interpretação e ou comunicação [30]. Segundo o artigo [17], a sonorização só pode ser chamada de sonorização, se o som for objetivo, se a transformação do mesmo for sistemática e se for reproduzível. Os resultados sonoros têm de ser estruturalmente idênticos para a mesma entrada de dados. Na Figura 3.1 pode-se ver todas as fases do processo de sonorização de dados, desde de a obtenção dos dados, passando pela sonorização (representação, geração e propagação de som) e até à compreensão do mesmo pelo ouvinte.

3.1 Vantagens

A sonorização ou exibições auditivas, apresentam uma série de vantagens, principalmente em processos de monitorização, através de alertas e alarmes sonoros, usando as capacidades auditivas de um ser humano. As capacidades da audição humana são diferentes das competências de reconhecimento de padrões visuais. Os humanos, têm uma resolução temporal maior em ouvir do que em ver. Conseguem assim, lidar melhor com informação sobreposta no domínio auditivo do que no domínio visual [21].

Os humanos podem habituar-se a padrões sonoros e continuar suscetíveis a mudanças, mesmo que estas sejam meramente subtis [21]. Perante uma situação sonora comum, os humanos conseguem detetar mudanças com alguma facilidade, mesmo que essas mudanças sejam insignificantes. Visualmente, um ser humano já tem mais dificuldade em detetar mudanças num padrão textual, e tem ainda mais dificuldade se as informações forem apresentadas textualmente de forma aglomerada e em grande quantidade.

A sonorização aparenta ser uma solução adequada para sistemas de monitorização, pois permite que a atenção visual seja focada noutras tarefas ou em outros trabalhos a realizar paralelamente[18, 21, 20]. Permite assim, que uma pessoa concilie pelo menos duas tarefas ao mesmo tempo. Uma tarefa processada pelas competências visuais, e outra tarefa entregue às competências auditivas de um ser humano. Quanto mais tarefas um ser humano realiza ao mesmo tempo, maior é a probabilidade de falhar ou de comprometer alguma tarefa. Esta probabilidade aumenta exponencialmente, se as tarefas a realizar forem só submetidas para um único campo, seja auditivo ou visual.

Quando os dados a monitorizar são apresentados visualmente de uma forma muito complexa com muitos dados num só ecrã, a sua leitura torna-se difícil. Uma exibição auditiva fornece um complemento útil e por vezes até substituto de uma exibição visual [18]. Quando existe uma grande quantidade de dados a analisar visualmente, e essa tarefa tem de ser efetuada com rapidez e certeza, o ser humano

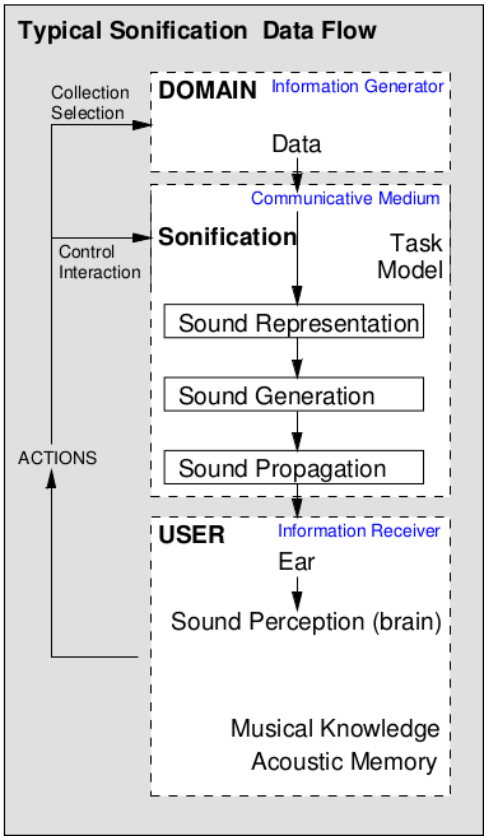

Figura 3.1: Fluxo de dados num processo de sonorização [16].

têm algumas dificuldades, inerentes ao processamento de várias tarefas ao mesmo tempo. Assim, a sonorização de dados pode ser um apoio para os humanos no processo de análise de uma grande quantidade de dados. Uma representação auditiva não interfere com uma exibição visual e vice-versa [16]. A sonorização tem todos os requisitos para aumentar o desempenho, gerir a carga de trabalho, e diminuir o stress em tarefas visualmente intensas [30]. O stress e a intensidade diminuem quando o ser humano dá uso, ao mesmo tempo, às suas competências auditivas e visuais.

O áudio é excelente para atrair, orientar ou encaminhar o ouvinte para dados chave [18, 20]. Perante um aglomerado de informação, por vezes existe informação excessiva ou informação que não é interessante num determinado momento. A sonorização da informação mais importante, ou da informação mais relevante, para a tarefa em causa, permitirá ao ouvinte realizar a sua tarefa com mais certeza e precisão, pois só os dados chave são sonorizados. A informação menos importante, será processada pelo ouvinte visualmente, enquanto que a mais importante ficará ao encargo da sua audição. Exibições auditivas permitem ainda uma liberdade extra aos olhos de um ser humano. Podem ser usadas quando os olhos estão ocupados em outras tarefas ou quando é impossível usar exibições visuais (pessoas com problemas visuais ou cegas)[16].

Os seres humanos são capazes de se habituar a certos padrões sonoros e podem, por isso, não atribuir uma atenção tão rigorosa a esses padrões. Contudo têm uma consciência auditiva e intelectual, passível de se alarmar caso o som seja diferente do habitual. Essa propriedade é chamada de *backgrounding* [16]. O sistema auditivo humano executa a decomposição das frequências do som, sendo bastante sensível a ritmos e suas mudanças [16]. Os seres humanos têm memória auditiva, que permite recuperar muitos sons familiares e caso seja necessário, rever certos sons, para fazer uma comparação num determinado

momento. Um exemplo disso, é a lembrança de uma música que era ouvida na infância [16].

Um monitorização usando som, pode ajudar a aumentar o desempenho na realização de uma determinada tarefa, reduzir a fadiga, reduzir o tempo de aprendizagem e ainda aumentar o entusiasmo [16]. A sonorização é apropriada para transmitir informação que muda ao longo do tempo, tal como eventos de rede ou até mudanças do estado de um sistema [20].

3.2 Limitações

A sonorização de eventos apresenta algumas desvantagens. Podem surgir problemas relacionados com o contexto em que a sonorização é usada, ou do ambiente onde é usada [16]. Adicionalmente, o processamento e perceção da informação depende das capacidades auditivas do ouvinte, podendo assim, variar de ouvinte para ouvinte [18].

Questões de estética e musicalidade são um tema em aberto na área da sonorização. O que para um ser humano é um som agradável, para outro pode não o ser. O uso de sons musicais, em vez de tons de ondas sinusoidais puras, têm sido recomendados por causa da facilidade com que os mesmos são aprendidos pelos humanos. Quanto mais cuidadosa for a atenção dada à estética musical, mais fácil de ouvir será. Tal irá, por sua vez, promover a compreensão da mensagem pretendida [18].

Diferenças e dificuldades pessoais são também um fator importante no processo de sonorização. Existem limitações por parte dos humanos, pois cada um tem as suas capacidades auditivas, as suas habilidades musicais e cognitivas e ainda as suas capacidades de perceção [18]. A capacidade de compreender e interpretar um som depende de cada humano. Enquanto a compreensão visual, pode ser facilmente ensinada, uma compreensão auditiva, é muito mais subjetiva [16]. Por um lado, os seres humanos têm capacidades de interpretar o som através da sonorização, mas por outro tem de existir algum esforço para interpretar o significado de uma determinada exibição auditiva [16]. A sonorização de informação, pode ser uma abordagem nova para o utilizador e pode ser inicialmente uma barreira, pois provavelmente será necessário um processo de aprendizagem e habituação [18].

O meio ambiente também poderá ser uma entrave num processo de sonorização. Em locais de trabalho abertos, e sem o uso de auscultadores de ouvido, é impossível ignorar o som proveniente de outros locais. Isto prejudica o processo de escuta da sonorização. Por outro lado, o uso de colunas de som poderá prejudicar, ou comprometer o trabalho e a privacidade dos nossos parceiros ou dos utilizadores daquele espaço [16].

3.3 Monitorização

A sonorização, é ideal para o processo de monitorização, pois oferece algumas melhorias [21] nesse campo. Apresenta melhoria no tempo de resposta em situações criticas. Num processo de monitorização em tempo real, os utilizadores têm de manter uma atenção ativa e focada a tempo inteiro, ou seja, 24 horas por dia. No mundo real, raramente existe uma pessoa paga só para monitorizar algo a tempo inteiro. Estas pessoas normalmente executam outras tarefas, passando a monitorização para segundo plano. Nestes casos, como o som é processado rapidamente (mais rapidamente que representações visuais), o uso de alertas auditivos, permite informar os utilizadores sobre situações críticas e assim diminuir o tempo de reação. O tempo de reação é um fator importante na execução de um processo de resposta a um incidente.

O uso de sonorização no processo de monitorização permite uma certa liberdade laboral, para quem está encarregue de fazer a monitorização. Permite centrar a atenção visual em outras tarefas, enquanto ao mesmo tempo, permite ouvir os resultados da sonorização. Como o som pode ser processado mais passivamente do que o visual, o resultado auditivo permitirá uma perceção discreta, sobre o estado da monitorização e não distrairá o utilizador na realização de outras tarefas. Em suma, uma exibição sonora não impedirá a realização de outras tarefas que possam ser processadas visualmente.

Atualmente, os sistemas de monitorização são desenvolvidos para gerar alertas que são transmitidos (seja visualmente ou sonoramente) sempre que, por exemplo, um valor pré-definido seja ultrapassado ou considerado crítico. Se os valores pré-definidos forem conservadores, podem correr situações indesejadas antes de ser gerado um alarme. Se os valores pré-definidos forem demasiado liberais, o risco de falsos-positivos aumenta e a geração de alarmes também aumenta, podendo sobrecarregar o utilizador com informações desnecessárias. Nestas situações, os utilizadores podem decidir ignorar os alarmes. A maioria dos utilizadores, prefere falsos-positivos a não serem avisados convenientemente antes de surgir uma situação crítica [21]. Em todo caso, o uso de sonorização de eventos e valores de um sistema de monitorização, poderá antever situações criticas e indesejadas.

A sonorização, afigura-se como viável para monitorizar redes de computadores, possibilitando a execução de outras tarefas em simultâneo.

3.4 Fatores de aplicação

Devem ser considerados alguns fatores aquando da aplicação da sonorização. A origem dos dados a sonorizar e a tarefa do ouvinte, são exemplos de tais fatores. Exemplos destes fatores incluem [18].

- Que tarefas se espera que o utilizador realize?
- Que informação deve ser sonorizada e que permita ao utilizador desempenhar a sua tarefa?
- Que quantidade de informação é necessária?
- Que tipo de exibição sonora é necessária (alertas simples, indicador de estado ou uma sonorização mais complexa)?
- Como manipular os dados (filtragem, transformação ou redução dos mesmos)?

Num processo de sonorização, existe sempre a necessidade de analisar que dados irão ser sonorizados e que dados o ouvinte irá ter de escutar e compreender, com vista a reduzir a complexidade e aumentar o desempenho da sua tarefa. As informações podem ser classificadas de várias formas, sendo as mais comuns a forma quantitativa (forma numérica) e a qualitativa (forma mais verbal). A criação de uma representação sonora de dados quantitativos pode ser bastante diferente de uma representação qualitativa. Os dados também podem ser sonorizados em termos de escala, categoria, intervalos, entre outros [18]. Podem ser representados por um fluxo contínuo de informações ou por informações mais discretas, como eventos que tenham ocorrido. Os investigadores da área estão cientes que os diferentes tipos de dados a sonorizar poderão influenciar todo o processo e deixam algumas sugestões a usar. Segundo os mesmos, deve-se fazer uso do timbre para dados categorizados, e altura (*pich*) ou sonoridade (*loudness*) para representar intervalos de dados [18]. No entanto, continua a faltar uma maior quantidade e qualidade de

pesquisas, que estudem, fatores e tipos de dados, que possam afetar a perceção ou compreensão auditiva dos mesmos. Em suma, para o desenvolvimento de um sistema de sonorização deve-se entender em primeiro lugar, que dados estão disponíveis e como estes podem ser categorizados ou medidos.

Adicionalmente, é também importante considerar quais serão as funções a executar pelo ouvinte dentro de um sistema de sonorização. Apesar de geralmente o ouvinte receber as informações de uma sonorização, os objetivos e as funções do mesmo afetam a forma como é implementada a sonorização. A sonorização escolhida, pode ainda restringir os tipos de tarefas a realizar pelo ouvinte e vice-versa. Existem alguns tipos de tarefas a ter em consideração [18]:

- Monitorização: o ouvinte escuta o resultado de uma uma sonorização para detetar eventos e identifica o significado de cada evento no contexto da operação do sistema.
- Exploração de dados: poderá submeter o ouvinte a sub-tarefas, afim de explorar os dados ouvidos pela sonorização. A sonorização escolhida, poderá então, restringir os tipos de tarefas quotidianas que podem ser realizadas pelo ouvinte, aquando da escuta da sonorização. Se o objetivo da sonorização, é a exploração de dados, primeiro deve-se estudar a melhor abordagem a seguir.
- Identificação de tendências: o utilizador tenta identificar aumentos e diminuições de dados. Tenta identificar que quantidades de dados são apresentados sonoramente em um determinado momento.
- Identificação da estrutura dos dados: a tarefa do ouvinte poderá envolver a identificação da estrutura dos dados e suas relações. Através da sonorização, o ouvinte espera extrair o máximo de informação possível sobre as relações e a estrutura de dados representada sonoramente.
- Verificação de dados: a sonorização poderá ser viável para examinar dados, sendo uma tarefa que exibe menos esforço do que uma monitorização, pois a mesma normalmente, não tem questões associadas. A inspeção de dados com som, pode revelar padrões e anomalias que não eram percetíveis em representações visuais.
- Múltiplas tarefas: em muitas aplicações de sonorização, o objetivo passa pela possibilidade de os utilizadores poderem realizar outras tarefas, enquanto se escuta a sonorização de certos dados.

3.5 Tecnologias

Atualmente existem algumas tecnologias a ter em consideração e que permitem a sonorização de dados de uma forma facilitada como: a linguagem de programação áudio Chuck (linguagem orientada a objetos com estrutura semelhante ao Java) e o protocolo Open Sound Control (OSC) que possibilita a comunicação com várias tecnologias como é o exemplo da comunicação entre uma aplicação Java e um objeto desenvolvido em linguagem Chuck.

3.5.1 Chuck

Chuck é uma linguagem de programação áudio, criada em 2002, por Ge Wang e Perry Cook [45]. Chuck é também uma linguagem de programação concorrente, com sintaxe familiar e é orientada a objetos. Requer uma máquina virtual [44] para funcionar, aparentando assim algumas semelhanças com linguagens como Java. Funciona a partir de classes e objetos, existem variáveis sejam do tipo inteiro ou string,

métodos, *arrays*, operadores lógicos, estruturas de controlo: (*if*, *while*, *until*, *for*), manipulação de *threads* e eventos. Apesar da sintaxe ser semelhante à do Java, o propósito desta linguagem é bem diferente. Tem como objetivo ser uma linguagem de programação para síntese de som e criação musical em tempo real. É uma linguagem de código aberto e funciona em Windows, Linux e Mac OS X. Chuck dá relevância a tempos e aspetos de concorrência (*threads*, por exemplo), aspetos estes que são necessários para a modificação de código em tempo real. Para quem tem algum conhecimento de programação orientada a objetos, é uma linguagem fácil de aprender, que oferece a compositores, pesquisadores e artistas, uma ferramenta de programação para construir e experimentar síntese de áudio e música interativa em tempo real.

3.5.2 OSC

OSC é um protocolo que oferece flexibilidade e permite a comunicação entre computadores, sintetizadores de som e outros dispositivos multimédia [50, 51]. Foi desenvolvido e revelado em 1997 por Matt Wright e Adrian Freed [36], na necessidade de colmatar algumas lacunas do protocolo de comunicação entre instrumentos musicais eletrónicos, computadores e outros dispositivos relacionados, o Musical Instrument Digital Interface (MIDI). O MIDI permite que vários instrumentos sejam tocados a partir de um único controlador, normalmente um teclado, o que tornava as configurações de palco muito mais portáteis. Foi um avanço significativo em 1983, mas como tinha algumas lacunas, principalmente na velocidade das mensagens trocadas entre os diversos dispositivos, levou a que Matt Wright e Adrian Freed, desenvolvessem, o protocolo OSC. O OSC é uma atualização ao MIDI que resolve os problemas decorrentes da falta de velocidade no transporte das mensagens.

3.6 Sonorização em redes de computadores

Existem soluções, que aplicam a sonorização a dados provenientes de uma rede de computadores. Cada uma destas soluções, utiliza uma abordagem distinta quer quanto à recolha dos dados a tratar, quer quanto à forma de sonorizar os dados recolhidos.

3.6.1 SOC Sonification System

Em [43] é proposto um sistema denominado *SOC Sonification System* que sonoriza uma rede em tempo real. Permite alertar os administradores, das operações que estão a ser realizadas na rede. Tanto o tráfego anormal como o normal são sonorizados. O objetivo é o de possibilitar que administradores monitorizem a rede enquanto realizam outras tarefas. Permite assim, aumentar os operadores disponíveis, sem sobrecarregar as funções cognitivas individuais dos mesmos, evitando situações de *stress*. Foi desenvolvido, usando a linguagem de programação áudio Pure Data [37] e também usando um *script* escrito na linguagem Python, auxiliado pela biblioteca socket. Primeiramente o tráfego é recolhido, usando um programa colecionador de pacotes, como por exemplo o *Wireshark*. Sobre o tráfego recolhido, eram analisadas as 4 variáveis a tratar (número de pacotes recebidos, número de pacotes enviados, número de pacotes recebidos, número de bytes enviados) num determinado intervalo com recurso ao *script* Python. O *script* cria *logs* e retorna as variáveis para a ferramenta de sonorização. A parte de áudio é manipulada e desenvolvida usando a linguagem Pure Data. Os dados recebidos são transformados em som, através da reprodução de um *loop* pré gravado e reproduzido continuamente ou da geração de um sinal sintetizado.

A amplitude do som, ou nível de som, é alternada de acordo com os valor das variáveis recolhidas sobre o tráfego de entrada. Quanto menor o valor, mais silencioso é o som gerado. O timbre é manipulado por um filtro chamado *band pass* que atribui um efeito abafado quando os valores provenientes dos *logs* são valores baixos e um som mais brilhante quando estes são elevados.

3.6.2 Peep

No artigo [14] é apresentado um sistema de monitorização, denominado de *Peep*, que permite que os administradores da rede identifiquem cargas elevadas, tráfegos excessivos na rede e *spam* de *emails*, através da transformação dos eventos da rede em sinais acústicos, nomeadamente em sons naturais como pássaros a chilrar, grilos, bicadas de pica paus, entre outros. Segundo os autores, sons naturais têm vantagens sobre os sons musicas, pois resultam quase sempre em qualquer combinação, semelhante à da natureza. O sistema permite que o administrador da rede se foque noutras tarefas, enquanto monitoriza a rede. Assenta em três princípios básicos: eventos, estados e *heartbeats*. Os eventos são naturalmente representados por um único pio ou chilro de um pássaro. O estado da rede altera o tipo, volume ou posição estéreo de um som de fundo em curso. Os *heartbeats* representam a existência ou a frequência de ocorrências de um estado da rede, tocando um som em intervalos variáveis (pio de um pássaro em intervalos mais pausados ou intervalos menos pausados). Assume uma arquitetura cliente/servidor. O cliente recolhe eventos na rede e envia-os, usando um *script* escrito em Practical Extraction and Report Language (Perl), para o servidor, usando pacotes Universal Datagram Protocol (UDP). O servidor é o encarregue de reproduzir o som, conforme os eventos recebidos. Consequentemente, o administrador pode escutar o resultado da sonorização reproduzida pelo servidor e reagir caso seja necessário. O sistema Peep, foi ainda projetado para tirar proveito das ferramentas de administração de sistemas existentes.

3.6.3 InteNtion

O projeto Interactive Network Sonification (InteNtion) [15] assume uma abordagem inovadora. Inovadora sobretudo na monitorização de tráfego de rede, que assenta na adição de uma nova dimensão, o som. O tráfego é recolhido recorrendo à biblioteca *SharpPCap*, analisado e transformado em som, para ajudar o administrador a detetar eficientemente intrusos na rede. Estatísticas da rede como protocolos usados(TCP, UDP), o Time To Live (TTL) dos pacotes, o tamanho dos pacotes, endereços origem e destino, tipos de serviço e portas, são calculadas. O tráfego recolhido é convertido em mensagens MIDI que depois são enviadas para sintetizadores dedicados que geram o som. O tamanho do pacote foi mapeado para ser usado como a frequência do som. Um pacote pequeno é representado por um som agudo e com uma frequência alta, já um pacote grande é representado por um som grave com uma frequência baixa. O TTL é usado para representar a duração da nota. A largura de banda da rede influencia o som através da aplicação de um filtro, que isola ou filtra certas frequências, num sintetizador apropriado para o efeito. Por fim, a distância entre o IP de origem e o de destino são parâmetros que influenciam o tamanho de reverberação (reflexão da onda sonora) do som. O som produzido, mostra assim a atividade dos utilizadores na rede. Isso significa que os utilizadores podem agir de forma interativa com os sons apenas por navegar na Internet, partilhar dados no Facebook ou sincronizar o seu Dropbox.

3.6.4 Songs of cyberspace

No projeto *Songs of cyberspace* [7] são usadas técnicas de sonorização para examinar o fluxo de dados provenientes da rede. A sonorização é usada, para suportar todo o ambiente envolvente e as decisões a serem tomadas no momento. Foi desenvolvido na linguagem de programação *SuperCollider* [31, 46]. Esta é uma linguagem desenhada para manipular áudio. É uma linguagem versátil, com capacidade para processar sinais em tempo real e de composição de algoritmos. É eficiente, tem capacidade de processamento de *arrays* ou listas e permite gerar eventos musicais. O objetivo do projeto é o de complementar um projeto anterior e usa os *logs* do mesmo. Um dos objetivos, era o de detetar e sonorizar *worms*, *scan* de portas e ataques Distributed denial-of-service (DDoS). As ligações são classificadas conforme as portas de destino e de origem, os protocolos (TCP, UDP) usados, tipo de pedido e até se é uma comunicação interna ou comunicação externa. Num primeiro nível, é usada uma lista com as portas comuns e o tipo de pedido para cada porta. Caso um pedido na rede seja efetuado para uma porta que não conste da lista, é reproduzido um zumbido que evidencia uma atividade incomum. Caso existam pedidos sistemáticos para uma porta que não se encontra na lista, tal retrata um sinal de tentativa de intrusão, sonorizado com uma melodia ou ritmos percetíveis como tal para o utilizador. No segundo nível, o objetivo era criar um som familiar para o utilizador, mas que permitisse a percetibilidade de pequenas mudanças no mesmo. Foi descartada a utilização de sons percussivos porque, segundo o autor, tornam-se facilmente uma distração e tendem a criar uma experiência de audição desagradável. O mapeamento entre o som e as informações dos pacotes de rede foi conseguida com uma associação entre os endereços IP de origem e destino, com um instrumento chamado gongo. Os quatro valores do endereço IP de origem, são parâmetros que são convertidos num timbre estrondoso. O chiar do gongo é criado a partir dos quatro valores do endereço IP de destino. O valor da força de cada batida aumentava e dependia do número de vezes que a mesma ligação era estabelecida.

3.6.5 NeMoS

O projeto NeMoS [29] consiste numa aplicação cliente-servidor para monitorizar um sistema distribuído com som, usando o protocolo Simple Network Management Protocol (SNMP). Para além de monitorizar a rede, pode monitorizar impressoras e outros equipamentos que disponham de SNMP. O servidor captura os dados, e o cliente analisa e produz o som, conforme a captura recebida. A sonorização, é efetuada associando eventos de rede com faixas MIDI. NeMos, é um projeto versátil, facilmente configurável, em que os eventos a capturar e a sonorizar são definidos pelo utilizador. É um sistema distribuído com suporte para múltiplos clientes e múltiplos servidores. Foi escrito na linguagem Java e usa Java Sound API [33]. Os eventos de rede capturados são agrupados em 7 categorias: segurança, rede, recursos da máquina, processos, serviços UDP e TCP, estado do dispositivo e impressoras. Conforme a categoria de cada evento, é gerada música de fundo com ficheiros MIDI. O administrador do sistema, pode configurar canais diferentes constituídos por um número de eventos a monitorizar ou por um componente específico da rede. Cada evento, é associado a uma ou mais faixas de um ficheiro MIDI. Musicalmente, é encontrada uma combinação de faixas MIDI que representam uma estrutura musical neutra no que diz respeito aos temas musicais habituais e convencionais. Foram usados timbres MIDI neutros, pouco comuns. O objetivo é não originar fadiga mental ou musical, perante um longo período de utilização.

3.6.6 NetSon

O projeto NetSon [49] é um sistema que permite monitorizar metadados numa rede em tempo real com sonorização. Devido ao volume de dados gerados a cada 24 horas numa grande organização, só aspetos relevantes são considerados e processados. O objetivo é detetar mudanças na rede que afetem o desempenho e a fiabilidade da rede. Toda a informação da rede é recolhida pela aplicação *sFlow*. O *SFlow* é uma ferramenta que permite detetar congestionamentos e problemas na rede, atividades não autorizadas, esboço de rotas e fluxos da rede, entre outras. O projeto assenta em duas vertentes: uma versão ”física”que permite o processamento em multi-canais da sonorização da rede, e uma versão ”online”em que o processamento era renderizado em modo *stereo*. É possível ver ainda, um vídeo que acompanha a sonorização, com efeitos visuais, em tempo real (*stream*). Informações sobre o *timestamp* do pacote, endereços IP de origem e destino, e carga da rede são utilizadas e mapeadas no processo de sonorização. No processo de sonorização é usada uma *framework* escrita em *Python*, de código-aberto, denominada de *SoniPy* [48]. O *SoniPy* lê a informação de um determinando pacote, como sua origem e seu destino, previamente identificado pelo *sFLow*, e faz uma reprodução sonora do mesmo. O tom da sonorização sobe e desce conforme a duração dos fluxos da rede. Aspetos como endereços IP conhecidos (origem dentro da organização) e desconhecidos (que chegavam de ou tinham como destino localizações fora da rede), também influenciam o resultado da sonorização. No final, um acompanhamento visual do resultado da sonorização era feito através da receção de mensagens UDP, provenientes do *SoniPy*.

3.6.7 SonNet

O projeto SonNet [47] é um projeto multi-plataforma, desenvolvido em Java e de código fonte aberto. Captura pacotes numa rede e transforma-os em som conforme a informação de cada pacote. Executa um processo de captura de pacotes, permitindo sonorizar dados provenientes de comunicações UDP ou TCP. Informações como o endereço IP origem, endereço IP de destino, porta de origem, porta de destino, *timestamp* do pacote, tipo de pacote, *flags*, protocolo, numero de sequência, direção, estado da ligação, número de pacotes por período, a média de pacotes enviados para um determinado IP, entre outras informações, são usadas para manipular a sonorização. Os pacotes capturados são enviados através do protocolo OSC para um objeto escrito na linguagem Chuck [45, 23]. OSC é um protocolo que oferece flexibilidade e permite a comunicação entre computadores, sintetizadores de som e outros dispositivos de multimédia [50, 51]. A comunicação é feita com mensagens OSC, em que cada mensagem contêm um número variável de argumentos com dados sobre os pacotes capturados (IP origem e de destino, protocolo, ...). O protocolo OSC, não impõem um número predefinido de argumentos e o formato da mensagem é independente da camada de transporte [52]. O objeto Chuck então criado, recebe as mensagens OSC com informações sobre cada pacote capturado e cria sons em tempo real. Uma das particularidades deste projeto, é que o som ouvido pode ser programado e modificado por um utilizador que perceba da linguagem de programação Chuck. Para tal basta modificar um *template* já definido pelos autores do SonNet que contem a estrutura básica de funcionamento. O utilizador está sempre dependente do tipo de sons e música que a linguagem Chuck permite. Num dos exemplos, apresentados pelos autores, para cada pacote, as 5 *flags* TCP e os 4 números relativos aos endereços IP de origem e destino são usados para criar um ritmo de 5 notas e uma sequência de 4 acordes. Percorrendo esta sequência isorítmica, a batida descansa se a *flag* for 0 e toca um acorde se a *flag* for 1. Outras informações, sobre pacotes na rede, influenciam as propriedades do som, as notas e o número de vezes que um ritmo se

repete. Conforme os pacotes são enviados e recebidos, criam-se repetições sonoras, originando uma melodia. A linguagem Chuck foi adotada, por ser uma linguagem de programação de áudio que permite criar som e música em tempo real, por ser gratuita, *open source* e estar disponível para Mac OS X, Windows e Linux.

3.6.8 Análise Comparativa

A Tabela 3.1 resume e caracteriza os vários projetos recentes, anteriormente apresentados, que usam processos de sonorização em redes de computadores. É apresentada uma comparação entre eles, a nível de objetivos, tipo de informação recolhida da rede, ferramentas e tecnologias usadas desde de o processo de recolha de dados da rede até ao processo de sonorização, o tipo e estilo de sonorização que é possível ouvir e ainda que componentes sonoros dão origem a essa sonorização. De referir que nenhum dos projetos, utiliza ou sonoriza, eventos gerados por um SIEM.

Tabela 3.1: Projetos sobre sonorização em redes de computadores.

Projeto	Objetivos	Informações sonorizadas	Tecnologias	Resultados sonoros
SOC Sonification System	ajudar administradores a realizar outras tarefas ao mesmo tempo	pacotes recebidos, pacotes enviados	Pure Data, scripts Python	*loops* pré gravados, sons sintetizados
Peep	realizar outras tarefas aquando da monitorização, spam de emails	estado da rede, pacotes de rede	mensagens UDP, scripts em Perl	sons naturais: pássaros a piar, grilos, bicadas de pica-paus
InteNtion	monitorizar a rede com som, detetar intrusos, abusos de infraestrutura	estado da rede, tipo serviço, portas, etc	SharpPcap, MIDI	sons sintetizados, sons com eco e reverbados
Songs of cyberspace	detetar worms, scan de portas, ataques DDoS	IPs, portas, protocolos comunicação interna ou externa, etc	SuperCollider	zumbidos, melodias com gongos
NeMoS	complementar um sistema de monitorização visual: rede e dispositivos	estado rede e dispositivos, tipo de serviços, eventos na rede, etc	MIDI, Java Sound API, Protocolo SNMP	faixas MIDI, timbres neutros, incomuns
NetSon	detetar mudanças na rede, segurança da rede	IPs conhecidos e desconhecidos, rotas, fluxos na rede, etc	SoniPy, SFlow	sem informação
SonNet	sonorização de dados de uma rede informática	pacotes recebidos, pacotes enviados, IPs, protocolos, portas, estado ligação, etc	Java, protocolo OSC, Chuck	sons sintetizados e isoritmicos

3.7 Conclusão

A sonorização de eventos contém vantagens, desvantagens e limitações. A sonorização pode substituir ou complementar uma captura de resultados de forma visual. O áudio é excelente em atrair, orientar, ou encaminhar o ouvinte para dados chave. Todos os dados são úteis num processo de monitorização de uma rede, pois a sonorização é apropriada para transmitir informação que muda ao longo do tempo, tal como eventos de rede ou até mudanças do estado de um sistema. A sonorização tem vários campos de aplicação e funções. Existem contudo, alguns fatores a ter em conta no momento de se especificar uma solução de sonorização. Existem vários projetos que recorrem à sonorização de redes de computadores, distintos a nível de objetivos, informações sonorizadas, tecnologias usadas e sobretudo nos resultados sonoros apresentados ao administrador da rede.

Capítulo 4

Proposta de Solução

A proposta de solução Music-enabled Security (MuSec), tem como objetivo ligar música e a segurança informática, através da sonorização de alertas que são detetados por um SIEM. A finalidade principal é permitir ao administrador da rede fazer outras tarefas do seu quotidiano, enquanto monitoriza a rede escutando os resultados sonoros da aplicação. De uma forma simplificada, existe a possibilidade de perceção se algo normal ou anormal está acontecer na rede, através de uma sonorização calma ou agitada, dependendo da situação. Na análise feita ao estado da arte, verificou-se que existem trabalhos sobre sonorização de redes de computadores, mas ainda não existe, uma sonorização baseada em eventos SIEM, o que faz desta proposta, uma solução inovadora. O principal objetivo é que o MuSec, seja útil aos administradores de rede e que simplifique o processo de deteção de ataques, invasões e intrusões, usando para tal as capacidades do OSSIM e as capacidades auditivas de um ser humano.

4.1 Metodologia de trabalho

A Tabela 4.1 descreve as principais fases de desenvolvimento da solução proposta. A metodologia usada durante o desenvolvimento foi a metodologia Rational Unified Process (RUP) [22] com as seguintes fases:

1. **Levantamento de requisitos**
2. **Especificação da solução**
3. **Implementação da solução**
4. **Testes à solução**
5. **Relatório**

A primeira fase consistiu na elaboração de uma lista de requisitos funcionais e não funcionais a cumprir pela proposta de solução. Foi instalado um servidor OSSIM para se perceber melhor o seu comportamento e funcionamento. O servidor OSSIM foi instalado e configurado numa fase inicial, porque é um componente essencial e preponderante na especificação da solução, no desenvolvimento e realização de testes na proposta de solução. Construíram-se ainda diagramas de casos de uso, para obter uma melhor compreensão de todos os requisitos funcionais.

Tabela 4.1: Fases do projeto.

ID	Tarefa	Inicio	Fim	Duração	T. Anterior
1	Levantamento Requisitos	14/03/2016	11/04/2016	25 dias	
2	Especificação da solução	18/04/2016	13/05/2016	20 dias	1
3	Implementação da solução	16/05/2016	15/07/2016	45 dias	2
4	Teste da solução	18/07/2016	05/08/2016	15 dias	3
5	Relatório	08/08/2016	31/10/2016	61 dias	1,2,3,4

A segunda fase incidiu sobretudo na análise do problema, no levantamento das dificuldades de implementação que poderiam surgir e quais as regras de negócio a aplicar de acordo com a lista de requisitos não funcionais elaborada anteriormente. Nesta fase foram definidas a arquitetura, as tecnologias, as linguagens e os protocolos a usar. Foram ainda elaborados os diagramas de classes, de atividades e de sequência, úteis para a compreensão do *workflow* da proposta de solução e no desenvolvimento da mesma.

A terceira fase consistiu na implementação da proposta de solução especificada na fase anterior, constituída pelas componentes Java MuSec e Chuck MuSec. Primeiro foi efetuado o desenho da interface gráfica em JavaFX. A proposta de solução foi ainda planeada para trabalhar sem interface gráfica, através de argumentos de linha de comandos. Seguiu-se o desenvolvimento da componente de acesso à base de dados do servidor OSSIM para recolha de eventos, a componente Java MuSec. Concluindo esta tarefa, foi implementada a comunicação entre a componente Java MuSec e o componente Chuck MuSec escrito na linguagem Chuck, tendo em vista a sonorização dos eventos recolhidos.

A quarta fase consistiu na realização de testes à proposta de solução. Foram realizados testes unitários, testes funcionais, testes de execução, testes de carga e ainda testes de estabilidade.

A quinta e última fase, consistiu na realização do relatório da aplicação e no levantamento do estado da arte dos vários temas em causa.

4.2 Identificação de requisitos

Um levantamento geral das características consideradas relevantes na perspetiva do utilizador de forma a detalhar quais as funcionalidades que devem estar presentes na proposta de solução, é fundamental para o sucesso da mesma. Tanto os requisitos funcionais como os não funcionais devem ser identificados e satisfeitos, pois são necessários para que o MuSec execute sem falhas e que cumpra com as necessidades a que se remete.

4.2.1 Requisitos funcionais

Requisitos funcionais descrevem comportamentos pretendidos para a proposta de solução. Ou seja, funcionalidades que a solução disponibiliza para os seus utilizadores. Os requisitos funcionais identificados para a proposta de solução são:

- **Recolha de alertas:** A proposta de solução deve recolher eventos de um servidor OSSIM, acedendo à sua base de dados MySQL.
- **Sonorização de alertas:** A proposta de solução deve sonorizar os eventos recolhidos.

4.2.2 Requisitos não funcionais

Requisitos não funcionais descrevem restrições na implementação dos requisitos funcionais e estão, tipicamente, relacionados com aspetos gerais do sistema. Os requisitos não funcionais presentes na aplicação são:

- **Multi-plataforma:** A proposta de solução deve sonorizar eventos provenientes de um SIEM, independentemente do sistema operativo em uso. Deverá funcionar em vários sistemas operativos como: Windows, Linux e Mac OS, podendo o administrador de rede instalar no seu computador de trabalho, independente do sistema operativo.
- **Vários modos funcionamento:** A proposta de solução deve funcionar através de uma execução sem interface gráfica (linha de comandos) ou através de uma interface gráfica, permitindo ao utilizador escolher a forma como quer interagir com a aplicação.
- **Auto-executável:** A proposta de solução deve funcionar imediatamente no arranque quando implementado num dispositivo próprio, num *Raspberry Pi* por exemplo.

4.3 Especificação da proposta

A linguagem Java e a interface gráfica JavaFX foram escolhidas porque permitem o desenvolvimento de aplicações multi-plataforma, requisito essencial a cumprir nesta solução. Aliás, a tecnologia JavaFX, permite a criação de aplicações com interface gráfica para *desktop*, navegadores web e até para dispositivos móveis como *android*, *iOS* [5], deixando a possibilidade de desenvolvimento para outras plataformas em aberto. São utilizadas ainda outras tecnologias, linguagens e protocolos, como a linguagem Chuck e o protocolo OSC.

A proposta de solução, de nome Music-enabled Security (MuSec), funciona em paralelo com o OSSIM e tira proveito das capacidades auditivas do ser humano. A Figura 4.1 apresenta a arquitetura da proposta de solução. A proposta de solução é formada por dois componentes principais: Java MuSec e Chuck MuSec. A componente Java MuSec, desenvolvida em Java [28], pode ser executada via linha de comandos ou através de uma interface gráfica simples e intuitiva, desenvolvida em JavaFX 8 [34]. A componente acede, através de uma ligação Secure Sockets Layer (SSL), à base de dados MySQL do OSSIM para recolher informação sobre os eventos capturados, extraindo informações úteis sobre cada evento. De seguida, com recurso à *framework* JavaOSC [39] e através do protocolo OSC, comunica com o componente Chuck MuSec, desenvolvido na linguagem Chuck. Nessa comunicação são enviadas mensagens OSC com informações sobre um determinado evento, em particular, são enviadas características como o risco, o valor da máquina de origem e de destino, a prioridade e a confiabilidade, que foram recolhidas anteriormente da base de dados do OSSIM, mais propriamente na base de dados alienvault_siem. Estas características são recolhidas da tabela acid_event, nos campos ossim_risk_c, ossim_asset_src, ossim_asset_dst, ossim_priority e ossim_reliability. A Figura 4.2 apresenta a estrutura da tabela acid_event.

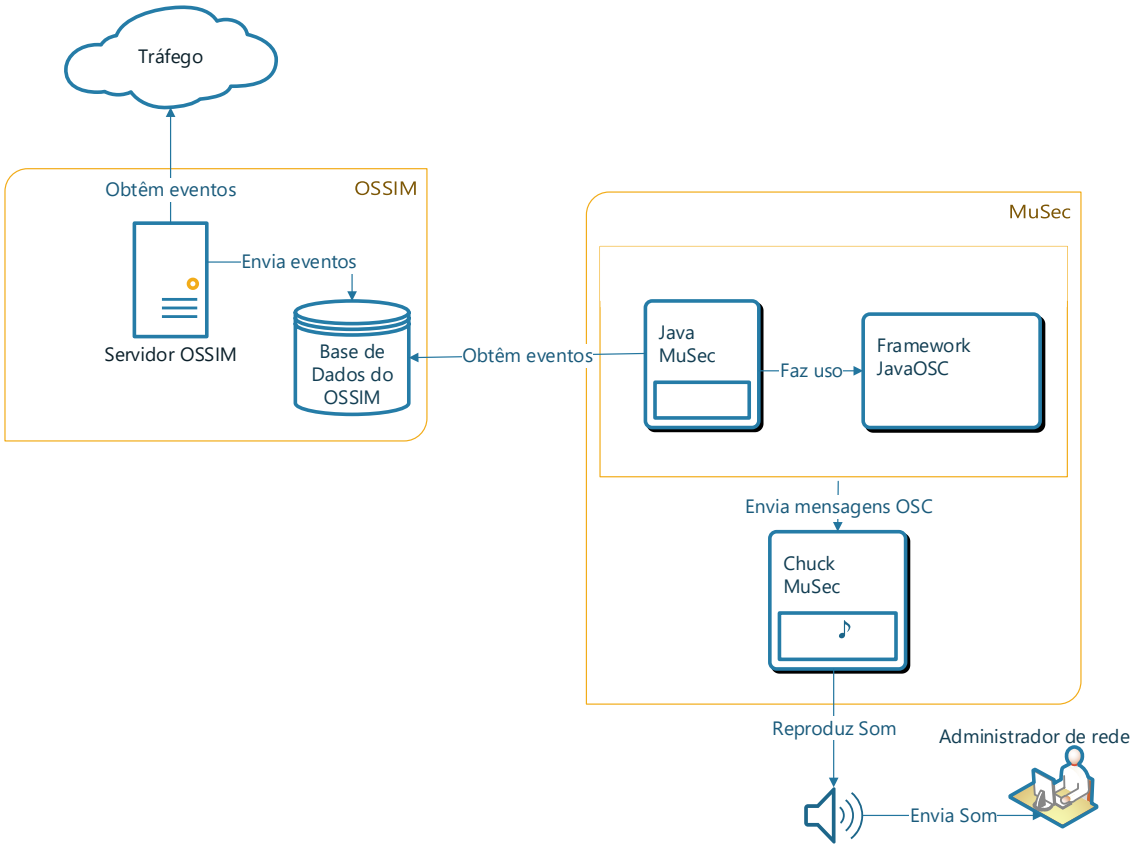

Figura 4.1: Arquitetura da aplicação MuSec.

timestamp	ip_dst	ip_src	ip_proto	layer4_sport	layer4_dport	ossim_priority	ossim_reliability	ossim_asset_src	ossim_asset_dst	ossim_risk_c
2015-11-09 12:14:58	c0a80109	c0a80109	6	0	0	1	1	2	2	0
2015-11-09 12:14:58	c0a80109	c0a80109	6	0	0	1	1	2	2	0
2015-11-09 12:14:58	c0a80109	c0a80109	6	0	0	1	1	2	2	0
2015-11-09 12:14:58	c0a80109	c0a80109	6	59024	0	1	1	2	2	0
2015-11-09 12:14:58	c0a80109	c0a80109	6	0	0	1	1	2	2	0

Figura 4.2: Tabela acid_event do OSSIM.

O componente Chuck MuSec, em função dessas características recebidas, transforma o evento em música. Música esta que será escutada pelo administrador da rede. A música transmitirá sensação de calma e relaxamento, caso os eventos recolhidos tenham um risco baixo, ou de agitação caso os eventos recolhidos sejam de risco elevado. A música é produzida através de *loops* musicais [1]. Cada evento tem um risco e esse risco tem um *loop* musical associado. A troca de *loop* musical, acontece caso um risco maior que o que está atualmente a tocar, seja detetado. Essa mudança é permitida, através da adição e remoção de máquinas virtuais, funcionalidade característica da linguagem Chuck. A musicalidade de cada *loop*, permitirá ao administrador perceber se algo se está a passar de grave na rede ou não, através das suas propriedades sonoras.

O servidor OSSIM captura o tráfego de uma ou mais redes e equipamentos, fazendo depois a normalização e tratamento interno de cada evento capturado. Categoriza numericamente os eventos, con-

[1] Música repetida, reproduzida através de ficheiros de música em formato *wav*, que mudam caso o risco atual seja superior ao anterior.

forme o seu nível de risco, prioridade e confiabilidade, entre outras características, e guarda as respetivas informações numa base de dados MySQL. O OSSIM também gera *logs* do seu próprio funcionamento.

4.3.1 Diagrama de casos de uso

Os casos de uso surgiram da análise e comparação de aplicações de sonorização em redes de computadores, com o objetivo de observar comportamentos e funcionalidades que poderiam ser importantes no desenvolvimento da proposta de solução. Estes casos de uso servem sobretudo para mostrar qual a utilidade do sistema, capturar os requisitos funcionais do sistema e especificar o contexto. Permite verificar em que casos de utilização surgem os diferentes atores presentes no sistema. Com recurso a diagramas de casos de uso descreve-se todo o funcionamento da proposta de solução. Na Figura 4.3 é possível ver o diagrama de casos de uso da proposta de solução. A partir da análise do diagrama de casos de uso, podemos observar quais as funcionalidades presentes no MuSec.

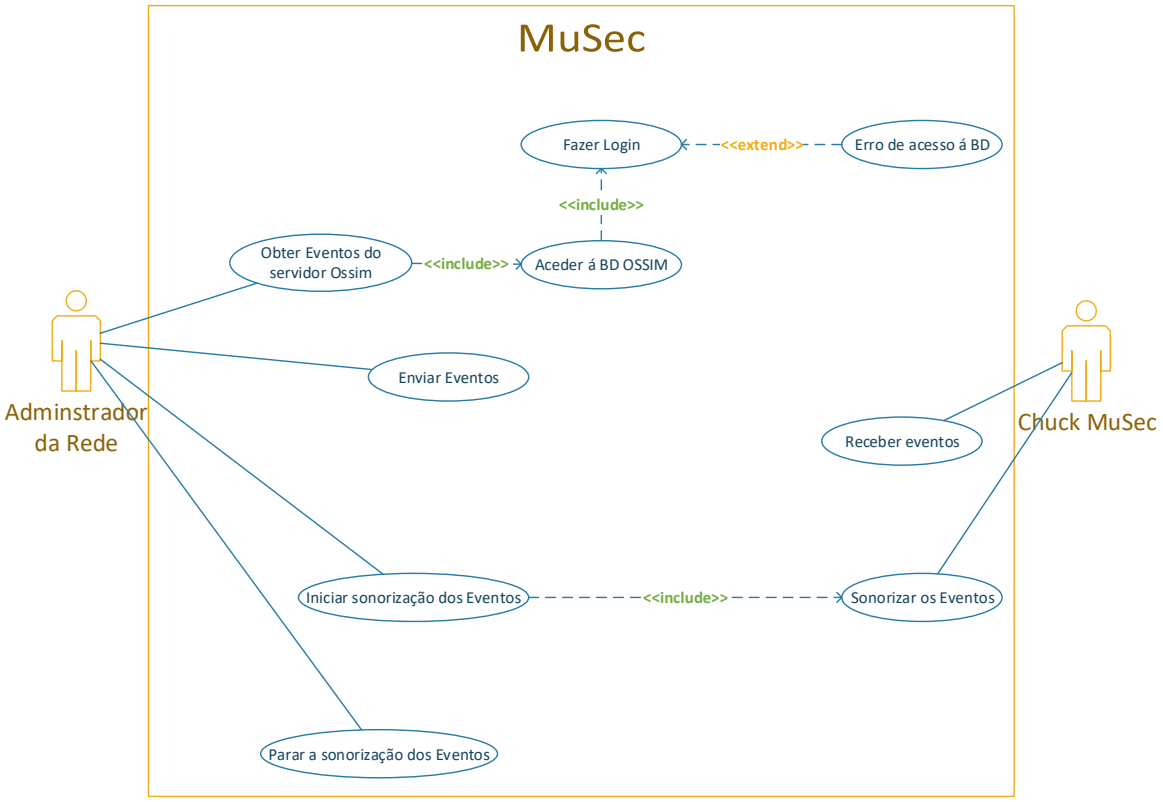

Figura 4.3: Diagrama de casos de uso

O administrador da rede, através da componente Java MuSec, terá acesso a quatro funcionalidades:

- Obter eventos do OSSIM.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.
  Para obter eventos do OSSIM, o administrador deve introduzir na componente Java Musec, as credenciais de acesso à base de dados do OSSIM.
  Pós condição: Visualização na componente Java MuSec dos eventos capturados da base de dados.
- Enviar eventos.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.

Para enviar eventos para o componente Chuck MuSec, o administrador deve introduzir na componente Java MuSec, as credenciais de acesso à base de dados do OSSIM e iniciar a sonorização.
Pós condição: Eventos recebidos pelo componente Chuck MuSec.

- Iniciar sonorização.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.
  O administrador deve clicar no botão de *"start"* caso inicie a componente Java MuSec através da interface gráfica. Caso inicie através da linha de comandos, deve definir o argumento de estado, igual a "*start*", bem como as credenciais de acesso ao servidor OSSIM.
  Pós condição: Sonorização de eventos provenientes do OSSIM.

- Parar sonorização.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.
  O administrador deve clicar no botão de *"stop"*, caso inicie a componente Java MuSec através da interface gráfica. Caso inicie através da linha de comandos, deve fazer Ctrl C, na janela onde foi iniciada.
  Pós condição: Paragem da sonorização dos eventos.

O componente Chuck ficará encarregue de:

- Receber eventos.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.
  O componente Chuck MuSec, recebe os eventos através do envio de mensagens OSC da componente Java MuSec para o mesmo.
  Pós condição: Visualização dos eventos recebidos.

- Sonorizar eventos.
  Pré condição: Solução MuSec e acesso ao servidor OSSIM.
  O componente Chuck MuSec, recebe os eventos e sonoriza os mesmos conforme o nível de risco de cada um. O administrador da rede, escuta o resultado da sonorização.
  Pós condição: Sonorização dos eventos.

4.3.2 Diagrama de classes

Um diagrama de classes, serve para modelar o vocabulário de um sistema, do ponto de vista do problema ou da solução. Ao longo das fases de desenvolvimento do projeto, o diagrama foi sofrendo ajustes devido à necessidade de acrescentar alguns atributos não identificados no levantamento de requisitos inicial. Foram feitos três diagramas de classes. Dois referentes à componente Java MuSec e outro referente ao componente Chuck MuSec, responsável pelo processamento e sonorização dos eventos.

Na Figura 4.4 pode observar-se uma versão simplificada do diagrama de classes da componente Java MuSec. Apenas se representam as classes e as suas relações. A versão completa deste diagrama consta da Figura G.1, em anexo ao relatório. A versão completa, além das classes e seus relacionamentos, apresenta também os atributos e os métodos de cada classe.

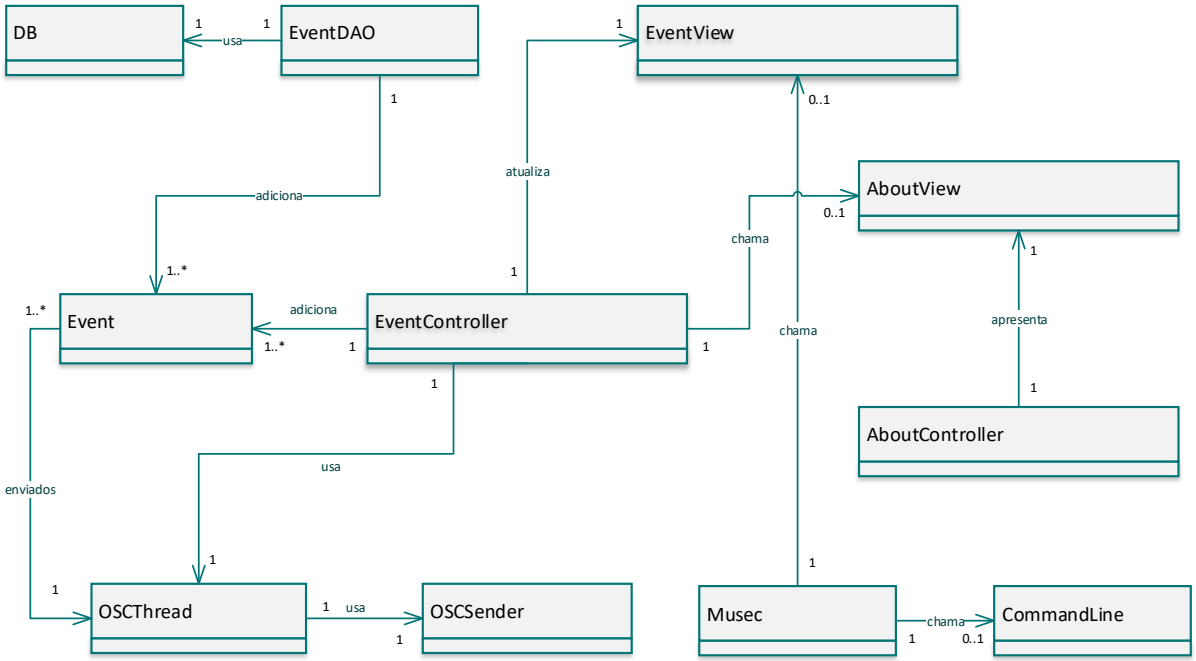

Figura 4.4: Diagrama de classes simplificado da componente Java MuSec

A classe Musec é a classe que inicia todo o processo e permite executar a componente Java MuSec em modo gráfico ou em modo linha de comandos. Se esta classe receber argumentos, a componente Java MuSec será executa em modo linha de comandos, senão será executada em modo gráfico com uma interface JavaFX. Na Listagem 4.1, é possível ver o código que desencadeia este funcionamento na componente Java MuSec.

Listagem 4.1: Excerto código da classe Musec

```
1  public static void main(String[] args) {
2
3      if (args.length > 0) {
4          //start command line mode
5          CommandLine commandLineMode = new CommandLine(args);
6          commandLineMode.start();
7      } else {
8          //start javafx interface mode
9          launch(args);
10     }
11 }
```

A classe CommandLine executa todo o fluxo de trabalho da componente Java MuSec, quando em linha de comandos. É nesta classe que são iniciadas duas *threads*. A primeira é responsável por aceder à base de dados do OSSIM e recolher os eventos lá contidos, adicionando-os depois a uma lista e mostrando-os, um a um, na linha de comandos. A segunda *thread* envia estes eventos para o componente Chuck. Na Listagem 4.2 é possível ver um excerto de código, sobre o funcionamento destas duas *threads*.

Listagem 4.2: Excerto código da classe CommandLine

```
1  public boolean start() {
2  (...)
3  if (oscSender != null && DB.getConnection(ip, user, pass) !=
       null) {
4      oscThread = new OSCThread(eventList, oscSender);
5      oscThread.setDaemon(true);
6      oscThread.start(); //start a thread that send events
7      //to Chuck via OSC messages.
8
9      Task task = new Task<Void>() {
10         @Override
11         public Void call() throws Exception {
12             ...
13             while (!Thread.interrupted()) {
14                 EventDAO dao = new EventDAO(ip, user, pass);
15                 if (eventCt == 0) { //first event
16                     last_Timestamp =
                           dao.getLastEventTimestamp(); //get
17                     //last timestamp event
18                 } else {
19                     Thread.sleep(1000);
20                     //if have new events in Ossim server database
21                     if (dao.haveNewEvents(last_Timestamp)) {
22                     EventDAO dao2 = new EventDAO(ip, user, pass);
23                     Event new_event =
                           dao2.getNewEvent(last_Timestamp);
24                     eventList.add(new_event);//add event in list
25                     Platform.runLater(() -> {
26                     System.out.println(new_event.getTimestamp()
                           + " | " + new_event.getRisk());
27                     });
28                     last_Timestamp = new_event.getTimestamp();
29                     //update timestamp
30                     }
31                 }
32                 eventCt += 1;
33             }
34             return null;
35         }
36     };
37     EventsUpdateThread = new Thread(task);
38     EventsUpdateThread.setDaemon(true);
```

```
39      EventsUpdateThread.start(); //start a thread that
40      //add events in eventList and show in cmd
41      (...)
42    }
43    (...)
44  }
```

A classe EventView cria e apresenta a janela principal, quando a componente Java MuSec é iniciada através da interface gráfica JavaFX. Na Listagem 4.3 é possível ver o método que inicializa a janela principal.

Listagem 4.3: Excerto código da classe EventView da componente Java MuSec

```
1  public void start(final Stage stage) throws Exception {
2      Parent root =
           FXMLLoader.load(getClass().getResource("Event.fxml"));
3      Scene scene = new Scene(root);
4      stage.setTitle("MuSec - Ossim Plugin");
5      stage.show();
6      EventView.stage = stage;
7  }
```

A classe EventController representa o controlador para vista EventView. Este controlador está encarregue de atualizar a vista EventView e também de inicializar todo o fluxo de trabalho da componente JavaFX, desde aceder à base de dados do OSSIM, bem como, inicializar e parar a sonorização. Está assim encarregue de iniciar duas *threads*, como as que foram referidas anteriormente na classe CommandLine.

A classe Event representa o objeto evento, permitindo posteriormente guardar vários eventos num *ArrayList*. Na Listagem 4.4 é possível ver o método construtor do objeto evento, bem como todos os parâmetros associados.

Listagem 4.4: Excerto código da classe Event da componente Java MuSec

```
1  public Event(String id, int priority, int reliability, int
       asset_src, int asset_dst, int risk, String timestamp) {
2      this.id = id;
3      this.priority = priority;
4      this.reliability = reliability;
5      this.asset_src = asset_src;
6      this.asset_dst = asset_dst;
7      this.risk = risk;
8      this.timestamp = timestamp;
9  }
```

A classe EventDAO permite executar consultas na base de dados do OSSIM, com o objetivo de obter novos eventos e guardá-los num *ArrayList*. A Listagem 4.5 apresenta um dos métodos que fazem consultas à base de dados. Neste caso, o método apresentado permite a obtenção de novos eventos, a partir de um determinado momento.

Listagem 4.5: Excerto código da classe EventDAO da componente Java MuSec

```
1  public Event getNewEvent(String timestamp) {
2      Event event = null;
3      try {
4          ResultSet rs;
5          String sql = "SELECT id, ossim_priority,
               ossim_reliability, ossim_asset_src, ossim_asset_dst,
               ossim_risk_c, timestamp FROM acid_event where
               timestamp>? GROUP BY id ORDER BY ossim_risk_c DESC
               LIMIT 1";
6          stmt = this.connection.prepareStatement(sql);
7          this.stmt.setString(1, timestamp);
8          rs = stmt.executeQuery();
9          while (rs.next()) {
10             //create a new event
11             event = new Event(rs.getString("id"),
                   rs.getInt("ossim_priority"),
                   rs.getInt("ossim_reliability"),
                   rs.getInt("ossim_asset_src"),
                   rs.getInt("ossim_asset_dst"),
                   rs.getInt("ossim_risk_c"),
                   rs.getString("timestamp"));
12         }
13         //close
14         rs.close();
15         stmt.close();
16         this.connection.close();
17     } catch (SQLException ex) {
18         System.err.println(ex);
19     }
20     return event;
21 }
```

A classe DB permite fazer uma ligação SSL à base de dados do servidor OSSIM (ver Listagem 4.6), através do método getConnection.

Listagem 4.6: Excerto código da classe DB da componente Java MuSec

```
1  public static Connection getConnection(String host, String
       userName, String password) {
2      Connection con = null;
3      try {
4          //validate certs
5          System.setProperty("javax.net.ssl.keyStore",
```

```
            "src/cert/keystore");
6           System.setProperty("javax.net.ssl.keyStorePassword",
            "PASSWORD");
7           System.setProperty("javax.net.ssl.trustStore",
            "src/cert/truststore");
8           System.setProperty("javax.net.ssl.trustStorePassword",
            "PASSWORD");
9
10          Class.forName(DRIVERCLASSNAME);
11          DriverManager.setLoginTimeout(10);
12          //create connection
13          con = DriverManager.getConnection("jdbc:mysql://" + host
            + ":" + PORT + DBNAME + "?verifyServerCertificate=true
14          &useSSL=true&requireSSL=true", userName, password);
15      } catch (ClassNotFoundException | SQLException ex) {
16          System.err.println(ex.getMessage());
17      }
18      return con;
19  }
```

A classe OSCThread representa a *thread* que ficará encarregue de enviar mensagens, via OSC, para o componente Chuck MuSec. Estas mensagens representam os eventos recolhidos do servidor OSSIM. A Listagem 4.7 apresenta um excerto do código de execução desta *thread*.

Listagem 4.7: Excerto código da classe OSCThread da componente Java MuSec

```
1   public void run() {
2       while (!OSCThread.interrupted()) {
3           if (events.size() > 0) { // if have events in list
4               Object[] play_Message =
                    {events.get(0).getTimestamp(),
                    events.get(0).getPriority(),
                    events.get(0).getReliability(),
                    events.get(0).getAsset_src(),
                    events.get(0).getAsset_dst(),
                    events.get(0).getRisk(), "play"};
5               this.osc_Sender.sendMessage(play_Message);//send event
6               events.remove(0); //remove the event sent.
7           }
8       }
9   }
```

A classe OSCSender é a classe responsável por criar um emissor de mensagens (protocolo OSC). O código do método construtor do emissor é apresentado na Listagem 4.8.

Listagem 4.8: Excerto código da classe OSCSender da componente Java MuSec

```
1  public OSCSender(InetAddress remoteIP, int remotePort, String
       address) {
2      this.remoteIP = remoteIP;
3      this.remotePort = remotePort;
4      this.address = address;
5      try {
6          this.sender = new OSCPortOut(remoteIP, remotePort);
7      } catch (Exception e) {
8          System.out.println("Error:" + e.getMessage());
9      }
10 }
```

O diagrama de classes do componente Chuck MuSec é apresentado na Figura 4.5. Neste diagrama podem observar-se as principais classes relacionadas com a sonorização dos eventos, referente à linguagem Chuck, bem como os relacionamentos entre classes que colaboram para a execução do processo de sonorização de eventos, recebidos da componente Java MuSec via protocolo OSC.

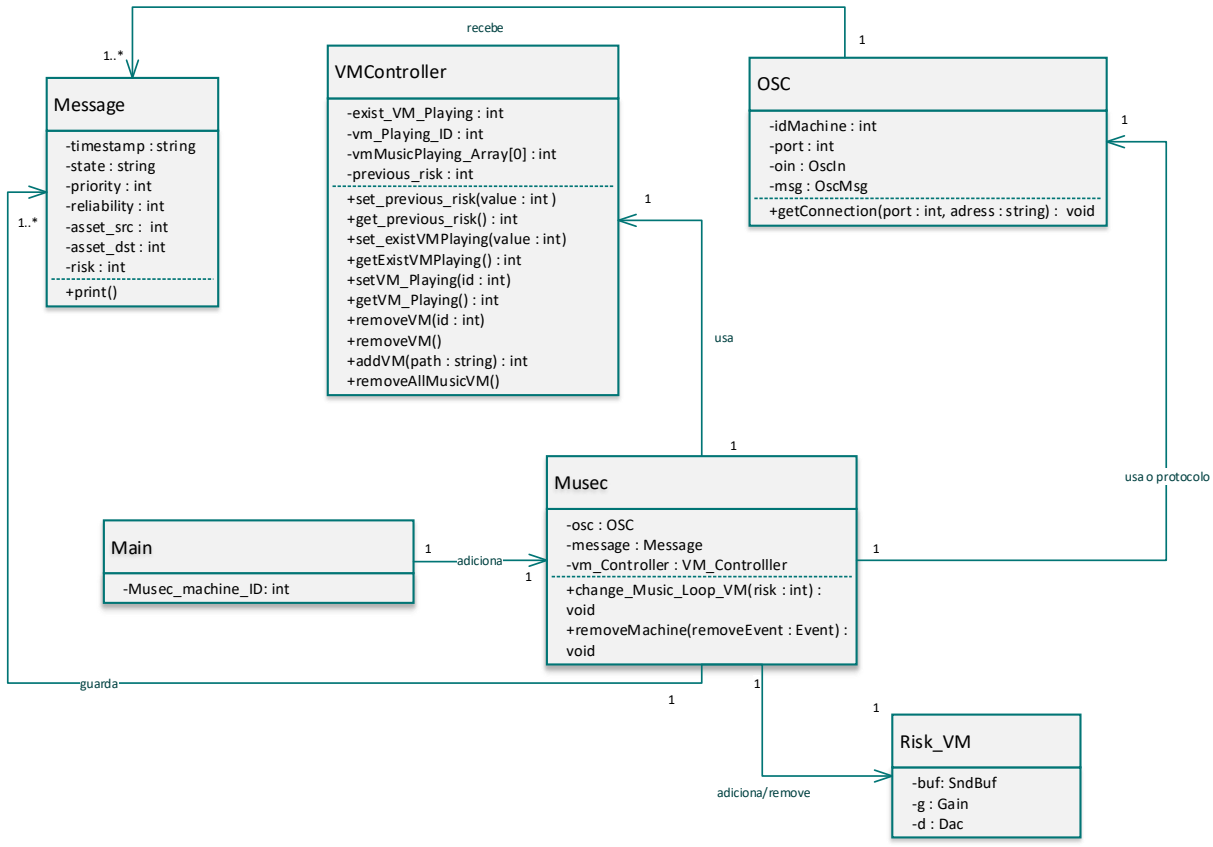

Figura 4.5: Diagrama de classes da componente Chuck MuSec

A classe Main é a classe responsável por adicionar todas as máquinas virtuais necessárias para executar o componente Chuck MuSec (ver Listagem 4.9).

Listagem 4.9: Excerto código da classe Main do componente Chuck MuSec

```
Machine.add(me.dir()+"/OSC.ck") => int OSC_machine_ID;
Machine.add(me.dir()+"/Message.ck") => int Message_machine_ID;
Machine.add(me.dir()+"/Musec.ck") => int Musec_machine_ID;
Machine.add(me.dir()+"/VM_Controller.ck") => int
   VM_Controller_machine_ID;
```

A classe Message é utilizada para armazenar as mensagens obtidas via protocolo OSC, que contêm os eventos recolhidos do servidor OSSIM. Todo o código desta classe é apresentado na Listagem 4.10.

Listagem 4.10: Excerto código da classe Message do componente Chuck MuSec

```
public class Message {
    string timestamp, state;
    static int priority, reliability, asset_src, asset_dst, risk;

    /*
    * Print message
    */
    fun void print(){
     <<< priority, reliability, asset_src, asset_dst, risk,
        timestamp, state>>>;
    }
}
```

A classe OSC representa um recetor de mensagens OSC. Permite que o componente Chuck MuSec receba os eventos enviados pela componente Java MuSec. Na Listagem 4.11 é apresentado o código referente á classe OSC.

Listagem 4.11: Excerto código da classe OSC do componente Chuck MuSec

```
public class OSC {
    int port;
    OscIn oin;
    OscMsg msg;

    fun void getConnection(int port, string address){
         // use port
        port => oin.port;
        // create an address in the receiver
        oin.addAddress(address);
    }
}
```

A classe Musec é utilizada para receber mensagens, através do recetor OSC e escolher a máquina virtual com a sonorização apropriada para cada evento. Na Listagem 4.12 é apresentado um excerto do

código referente a esta classe. Em particular, é mostrada a função que permite mudar de sonorização, através da alternância entre máquinas virtuais.

Listagem 4.12: Excerto código da classe Musec do componente Chuck MuSec

```
1  fun void change_Music_Loop_VM(int risk){
2     vm_Controller.set_previous_risk(risk);
3     vm_Controller.set_existVMPlaying(1);
4
5     //<<< "adding machine">>>;
6     if(risk == 0){
7         vm_Controller.addVM(me.dir()+ "/risk/risk_0.ck") =>
              vmLoopID;
8     }else if(risk == 1){
9         vm_Controller.addVM(me.dir()+ "/risk/risk_1.ck") =>
              vmLoopID;
10    }else if(risk == 2){
11        vm_Controller.addVM(me.dir()+ "/risk/risk_2.ck") =>
              vmLoopID;
12    }else if(risk == 3){
13        vm_Controller.addVM(me.dir()+ "/risk/risk_3.ck") =>
              vmLoopID;
14    }
15    (...)
16    vm_Controller.setVM_Playing(vmLoopID);
17 }
```

A classe VMController, permite controlar as máquinas virtuais. Possibilita a adição ou remoção de máquinas virtuais, com a respetiva sonorização de um evento associada. Na Listagem 4.13 é possível ver um excerto de código da classe VMController com as funções de remoção e adição de máquinas virtuais.

Listagem 4.13: Excerto código da classe VMController do componente Chuck MuSec

```
1  /* Remove VM playing */
2  private void removeVM(){
3     Machine.remove(this.vm_Playing_ID);
4     vmMusicPlaying_Array.popBack();
5  }
6
7  /* Add a specific VM */
8  private int addVM(string path){
9      Machine.add(path) => int id;
10     vmMusicPlaying_Array << id;
11     return id;
12 }
```

A classe Risk_VM representa um conjunto de classes que podem ser invocadas e executadas pela classe Musec. Cada classe está encarregue de reproduzir um ficheiro, em formato wav, com a sonorização pretendida para cada nível de risco. Na Listagem 4.14 é possível ver um enxerto de código referente à sonorização de um evento com risco de nível 1.

Listagem 4.14: Excerto código de classe Risk_1 do componente Chuck MuSec

```
1  // buffer for read an file
2  SndBuf buf => Gain g => dac;
3  // read wav
4  me.dir(-1) + "sounds/loops/risk_1_instrumental_guitar.wav" =>
       buf.read;
5
6  // set the gain (volume)
7  1 => g.gain;
8
9  // play all music right now
10 buf.length() => now;
11
12 VM_Controller  vm_Controller;
13 vm_Controller.set_existVMPlaying(0); //music end
14
15 //if not exist new events, play risk 0 VM in loop, until have new events
16 vm_Controller.set_previous_risk(0);
17 vm_Controller.set_existVMPlaying(1); //exist one VM playing
18 vm_Controller.addVM(me.dir()+ "/risk_0.ck") => int vmLoopID;
19 vm_Controller.setVM_Playing(vmLoopID); //VM ID playing right now
```

4.3.3 Diagrama de atividades

O objetivo do diagrama de atividades é o de mostrar o fluxo de atividades num processo. O diagrama mostra a interdependência entre atividades, mas também o relacionamento entre os vários casos de usos e os componentes da proposta de solução. A Figura 4.6 apresenta o diagrama de atividades da proposta de solução. Inicialmente o administrador de rede, através da componente Java MuSec, tenta comunicar com a base de dados MySQL do OSSIM. Para isso envia as credenciais de *login* (endereço IP, utilizador e palavra-passe) para a base de dados do OSSIM. Caso as credenciais de acesso estejam incorretas o processo é interrompido e é apresentada um mensagem de erro. Caso o *login* exista, a componente Java MuSec, recebe eventos recolhidos da base de dados e apresenta-os ao administrador de rede. Caso o processo de sonorização seja para continuar, ou seja, o administrador de rede não ”pediu”nenhuma paragem no processo de sonorização, a componente Java MuSec envia os eventos recolhidos para o componente Chuck MuSec, em que depois são recebidos e sonorizados adequadamente por este. Enquanto o administrador de rede não indicar, através da componente Java MuSec, que se deve parar, todos este processo desde a receção de um evento até a sonorização desse evento é repetido. Caso o administrador não queira continuar o processo de sonorização, a componente Java MuSec envia uma mensagem ao componente

Chuck MuSec, a indicar a paragem de todo o processo e todo o processo é interrompido.

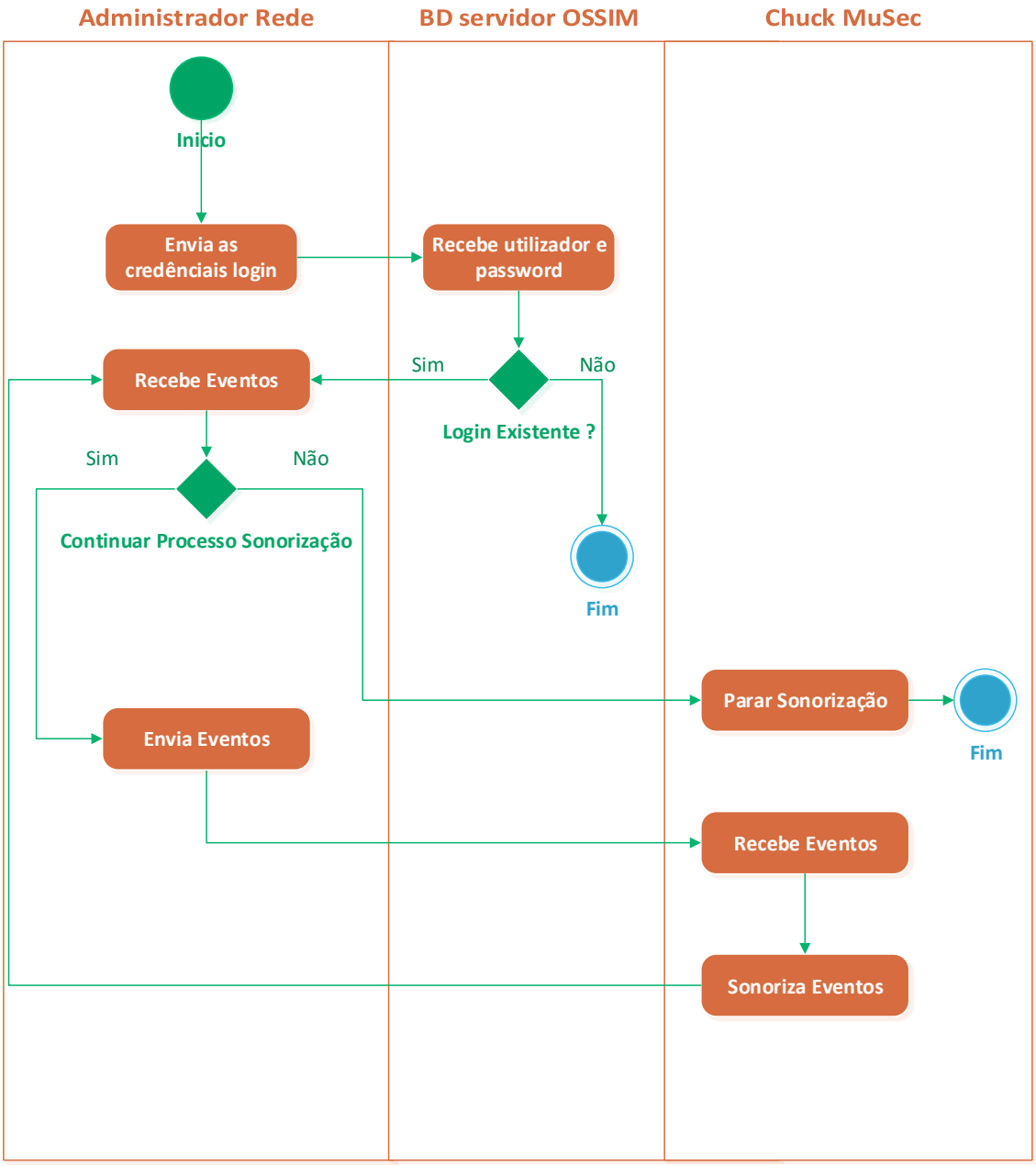

Figura 4.6: Diagrama de atividades do MuSec

4.3.4 Diagrama de sequência

Nesta parte, são apresentados os vários diagramas de sequência, dos vários casos de uso identificados anteriormente. Na Figura 4.7 é possível ver o diagrama de sequência relativo a todo o processo de sonorização de eventos que surge quando o administrador de rede, executa a componente Java MuSec. Pode ver-se que administrador inicia o processo, clicando no botão ”start”, que permite enviar as credencias de acesso à base de dados do servidor OSSIM. A base de dados responde com uma mensagem de sucesso, caso as credenciais estejam corretas. De seguida, o administrador efetua o pedido para obtenção de novos eventos da base de dados, através da componente Java MuSec. A este pedido, a base de dados responde, enviando novos eventos. Depois a componente Java MuSec, envia o evento recebido para o

componente Chuck MuSec. Este por sua vez, recebe, processa e sonoriza esse evento. Por fim, o componente Chuck MuSec fica a espera de novos eventos, provenientes da componente Java MuSec, para os sonorizar.

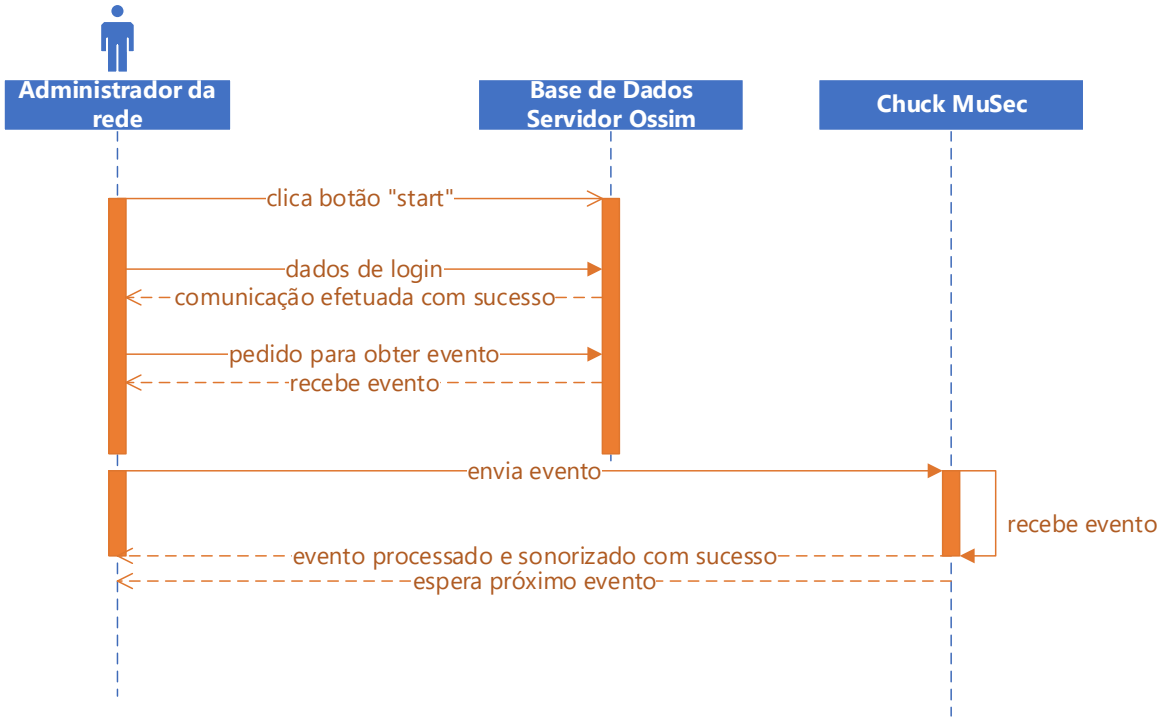

Figura 4.7: Diagrama de sequência: Processo de sonorização de eventos.

Na Figura 4.8 é possível ver o diagrama de sequência relativo à paragem da sonorização pelo administrador da rede, usando a aplicação MuSec. Pode ver-se que administrador de rede, inicia o processo através da componente Java MuSec, clicando no botão ”stop”. De seguida, a componente Java MuSec envia uma mensagem a informar o componente Chuck MuSec para suspender todo o processo de sonorização. O componente Chuck responde com a paragem de toda a sonorização. O administrador, a partir deste momento, não escutará mais sons resultantes da sonorização.

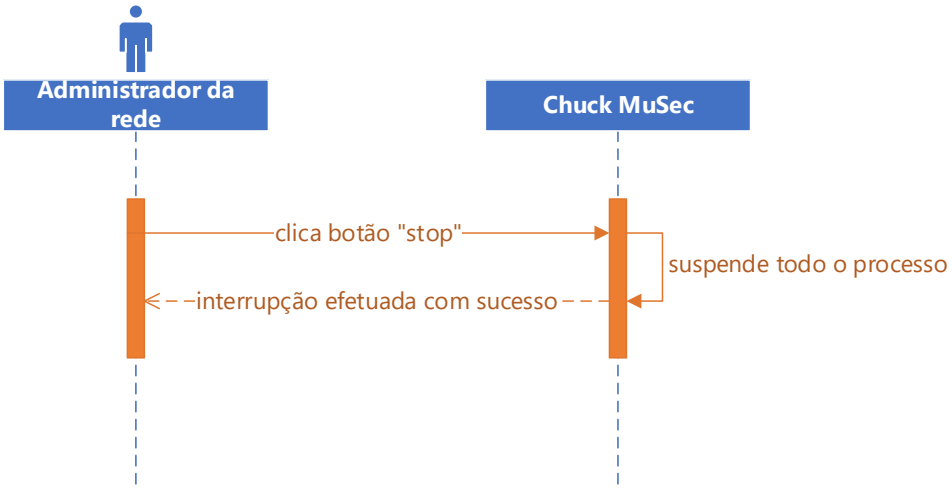

Figura 4.8: Diagrama de sequência: paragem da sonorização.

4.4 Conclusão

A proposta de solução foi caracterizada e foram identificados os seus requisitos. Foi descrita a metodologia de desenvolvimento adotada. Foi ainda apresentada a arquitetura e todas as tecnologias envolventes, bem como foram divulgados os principais diagramas usados para auxiliar o desenvolvimento da solução.

A solução proposta tem como objetivo ajudar um administrador de rede a realizar outras tarefas do seu quotidiano enquanto consegue monitorizar a rede com alguma simplicidade. Para tal, o MuSec sonoriza eventos de rede gerados por um SIEM. Aproveitam-se assim todas as funcionalidades e capacidades de monitorização do SIEM, em conjunto com todas as capacidades auditivas de um ser humano. A proposta de solução acede à base de dados do OSSIM para assim obter toda informação necessária sobre os vários eventos identificados pelo mesmo.

A descrição dos eventos inclui informação como o risco, entre outros valores numéricos, que depois são utilizados para sonorizar esses eventos. Cada nível de risco foi mapeado num determinado som. Estes sons são *loops* musicais em formato *wav* que representam sons calmos e relaxados, em níveis de riscos baixos, ou sons mais agitados, como *loops* de *heavy metal* e *hard rock*, que representam níveis altos de risco. A discrepância sonora entre esses sons, permite alertar eficientemente o administrador de rede, se algo está afetar a rede ou não.

Capítulo 5

Validação da solução proposta

O OSSIM foi instalado e configurado antes mesmo de se iniciar a especificação de uma solução. Tal foi necessário, para compreender melhor o OSSIM, o seu funcionamento e estrutura de base de dados. O OSSIM foi instalado e configurado na ESTG, mais propriamente no centro informático da mesma. A Figura 5.1 apresenta o diagrama da rede onde se inclui a instalação e os componentes de monitorização do OSSIM na ESTG. O servidor OSSIM monitoriza a rede de servidores e a rede sem-fios *eduroam*. Também foi instalado um agente de monitorização no servidor do *moodle* da instituição. O Anexo A descreve o processo de instalação e configuração inicial que foram realizados no OSSIM.

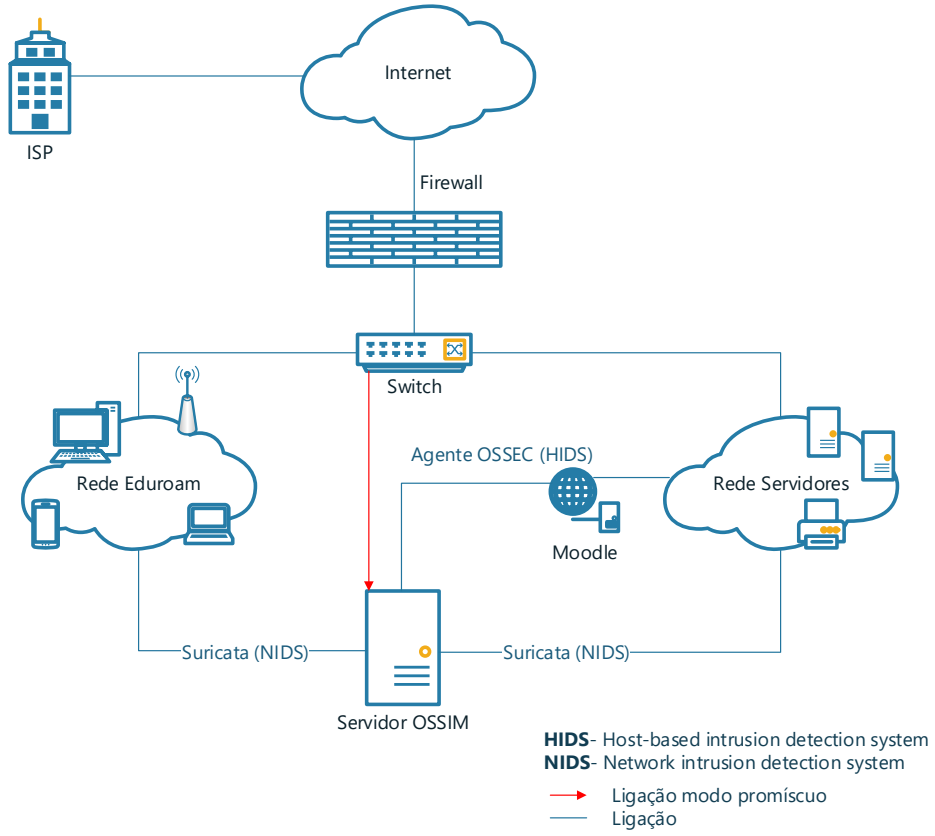

Figura 5.1: Diagrama de instalação do OSSIM na ESTG.

5.1 Testes funcionais

A execução da proposta de solução requer a execução das componentes Java MuSec e Chuck Musec. A componente Java MuSec pode ser usada através de um interface gráfica, escrita em JavaFX, ou em linha de comandos. Os passos para executar a componente Java MuSec com ou sem interface gráfica e para executar o componente Chuck MuSec serão demonstrados de seguida. Neste exemplo, a proposta de solução foi testada num sistema Windows 10 a 64 bits, num sistema Linux Mint 17.3 a 32 bits e num sistema Mac OS X El Capitan a 64 bits.

A execução da proposta de solução é efetuada com recurso a uma linha de comandos e implica a execução prévia da componente Chuck MuSec. Os comandos para lançar a execução de ambos os componentes são apresentados na Listagem 5.1. Após a execução destes comandos, deverá surgir a interface gráfica da componente Java MuSec que irá permitir iniciar a sonorização.

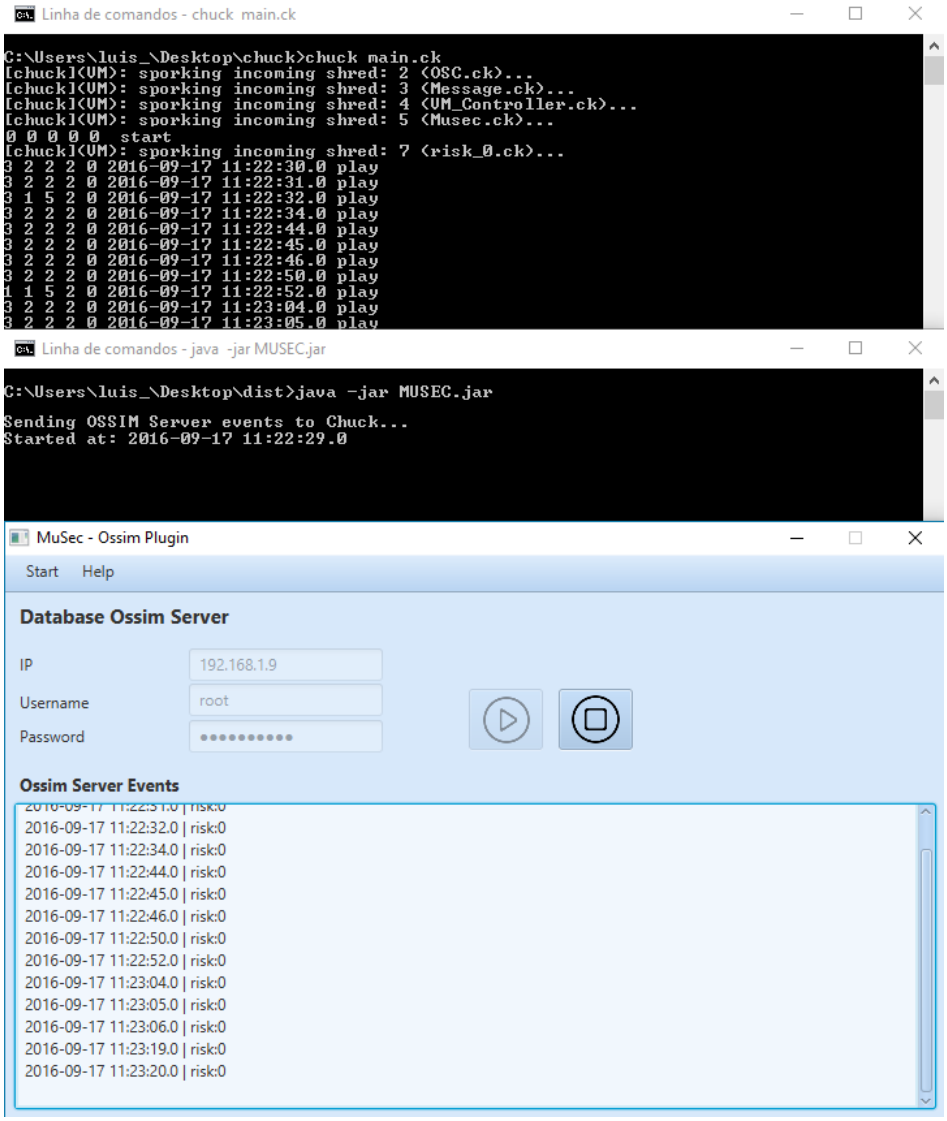

Figura 5.2: MuSec em Windows 10 x64.

A Figura 5.2 demonstra a execução destes comandos num sistema Windows, bem como apresenta a interface gráfica da componente Java MuSec. Com ambos os componentes em execução, o administrador

só terá de introduzir o endereço IP do servidor OSSIM, o utilizador e a password da base de dados do mesmo e clicar no botão *start*, dando inicio à sonorização.

Listagem 5.1: Execução da proposta de solução com interface gráfica.

```
1 chuck CAMINHO/main.ck
2 java -jar CAMINHO/MUSEC.jar
```

A execução da proposta de solução em ambiente linha de comandos é similar à execução com ambiente gráfico, requerendo a execução prévia da componente Chuck MuSec. Os comandos necessários para o efeito são apresentados na Listagem 5.2. Diferindo apenas no comando de execução da componente Java MuSec que requer informação adicional. A saber, requer a passagem como argumento do endereço do servidor OSSIM, do utilizador e da *password* para aceder à base de dados do OSSIM.

Listagem 5.2: Execução da solução em linha de comandos.

```
1 chuck CAMINHO/main.ck
2 java -jar CAMINHO/MUSEC.jar start IP UTILIZADOR PASSWORD
```

A Figura 5.3 demonstra a execução da proposta de solução em ambiente linha de comandos, num sistema Mint.

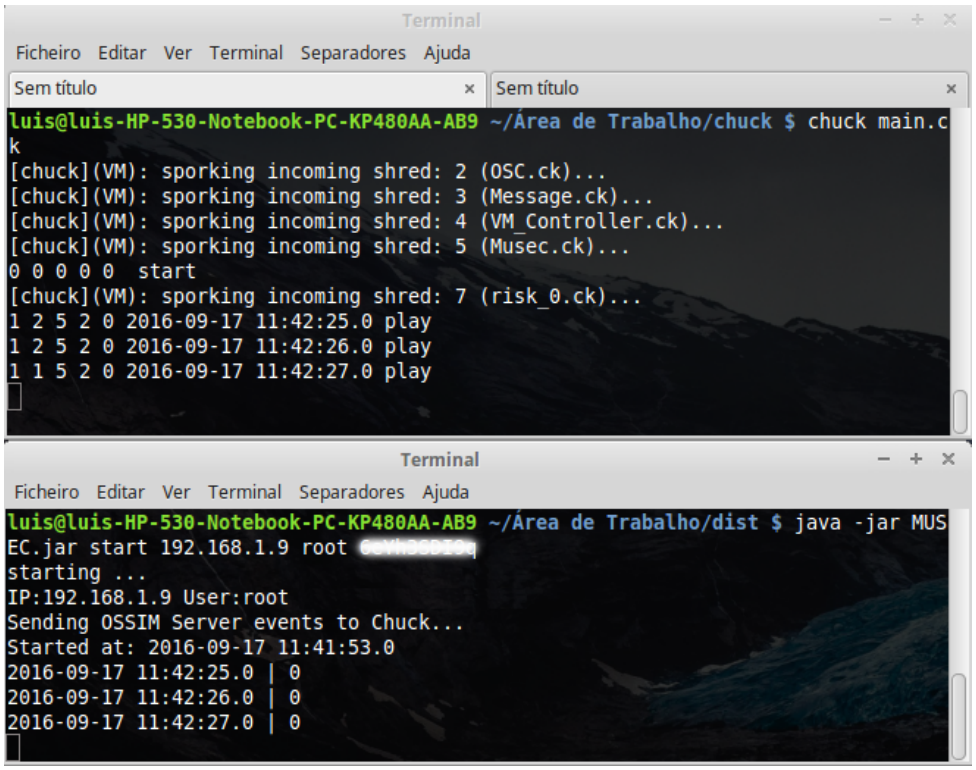

Figura 5.3: MuSec em linha de comandos em Linux Mint 17.3 x86.

Na Figura 5.4, é apresentada a mesma execução, mas agora num sistema Mac OS X El Capitan.

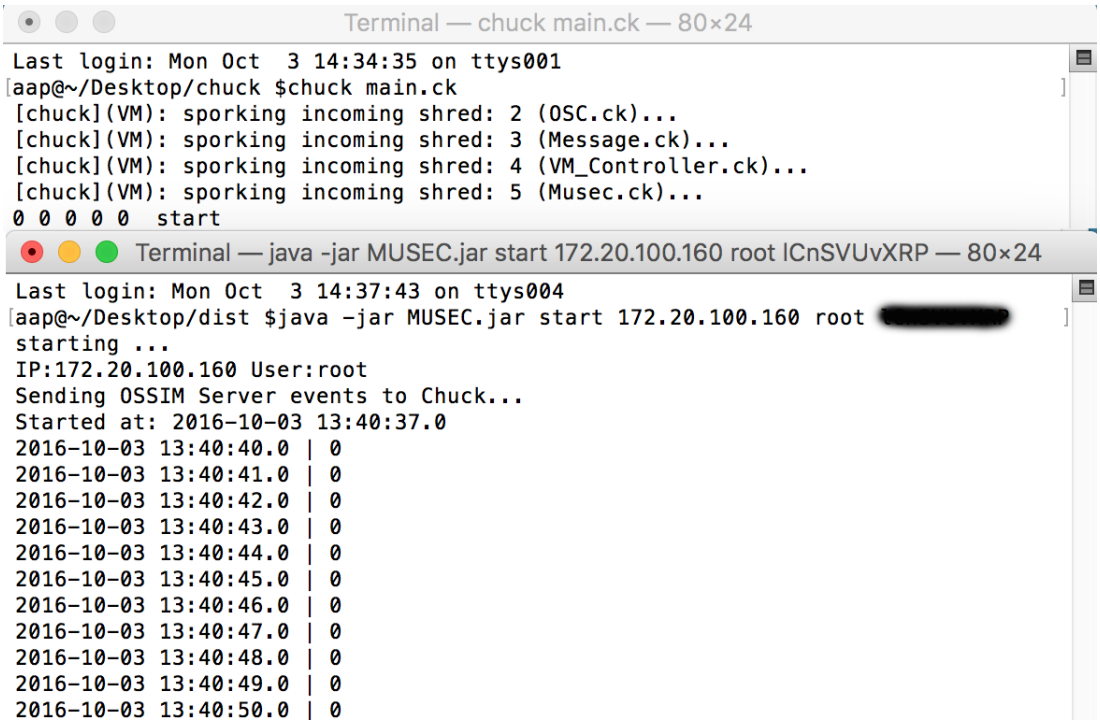

Figura 5.4: MuSec em modo linha de comandos, via sistema Mac OS X El Capitan x64.

5.2 Implementação na ESTG

A versão em linha de comandos foi projetada para que fosse possível disponibilizar a proposta de solução como um produto, ou seja, como uma *appliance*. Para tal, recorreu-se a um sistema *Raspberry Pi 2*, como mecanismo de implementação do produto de monitorização. A ideia base era a de simplificar o processo de instalação, bastando só ligar o equipamento e escutar a sonorização.

A Figura 5.5 expõe um diagrama simplificado da implementação da aplicação MuSec na ESTG, na sua versão *Raspberry Pi*. O servidor OSSIM utilizado é o mesmo que foi previamente instalado, encarregue de monitorizar o tráfego da rede *eduroam* e da rede de servidores. O *Raspberry Pi 2*, com a proposta de solução instalada, consulta a base de dados de eventos do OSSIM e sonoriza-os. Estes eventos são sonorizados pela componente Chuck MuSec em conformidade com o seu nível de risco. O administrador consegue assim, através de um *Raspberry Pi 2* e de umas colunas, ouvir o resultado da sonorização dos eventos identificados pelo OSSIM, enquanto realiza as suas tarefas do quotidiano.

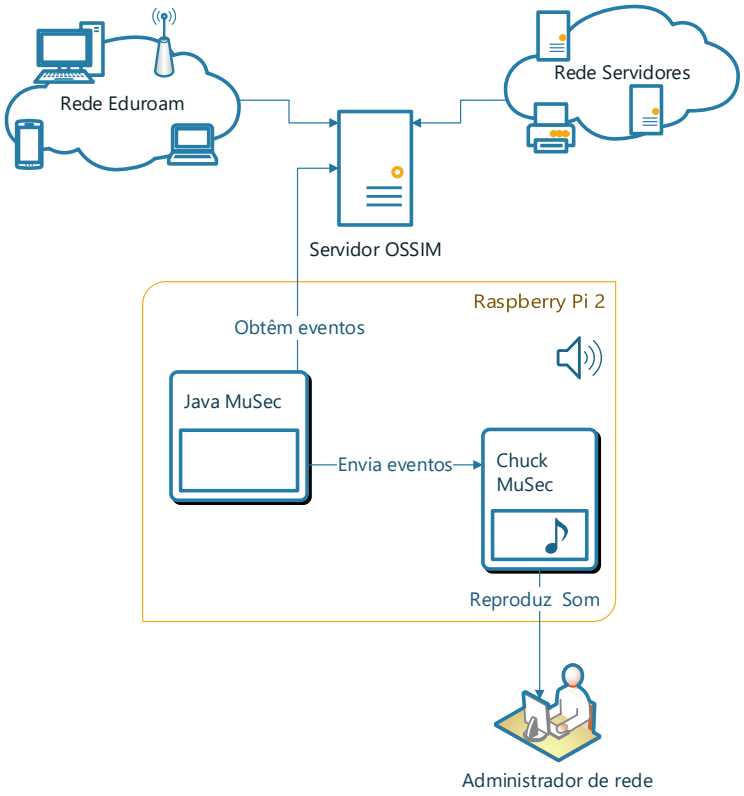

Figura 5.5: Diagrama de implementação do MuSec, na ESTG.

Na Figura 5.6 podemos ver a proposta de solução a funcionar num *Raspberry Pi 2* com o sistema operativo Raspbian Jessie. De notar que, para o MuSec funcionar, foram previamente instaladas as dependências (Java e Chuck). Estas foram instaladas com base na descrição feita para o Linux Mint, na secção F dos Anexos.

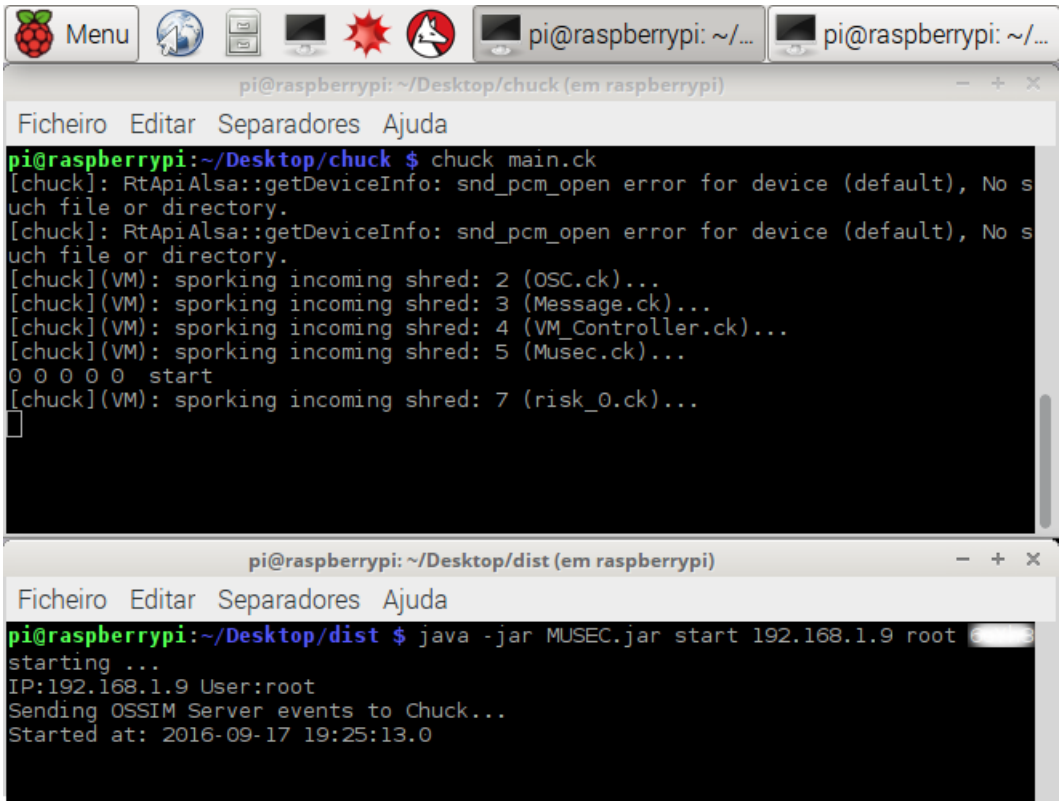

Figura 5.6: MuSec a funcionar num *Raspberry Pi 2*.

A implementação e funcionamento da proposta de solução num *Raspberry Pi 2* requer que a mesma seja automaticamente iniciada, aquando do arranque do dispositivo. Caso a energia falhe, o dispositivo também deve reiniciar a solução de forma automática. Para que a proposta de solução execute automaticamente foram criados *scripts* para serem executados no arranque do *Raspberry Pi 2*. O *script* apresentado na Listagem 5.3 permite executar o componente Chuck MuSec automaticamente no arranque do *Raspberry Pi 2*.

Listagem 5.3: Script para inicialização do Chuck.

```
1  #!/bin/bash
2
3  #kill all processes
4  killall chuck &> /dev/null
5
6  #write to a log file
7  echo "----------" >> /home/pi/Desktop/logs/chuck.txt
8  date >> /home/pi/Desktop/logs/chuck.txt
9
10 #run the chuck and write the output to a file
11 chuck /home/pi/Desktop/chuck/main.ck >>
      /home/pi/Desktop/logs/chuck.txt 2>&1
```

O *script* exposto na Listagem 5.4 permite executar a componente Java MuSec automaticamente no arranque do *Raspberry Pi 2*.

Listagem 5.4: Script para inicialização do MuSec.

```
#!/bin/bash

connection=0

#try to connection with OSSIM Server
while [[ "$connection" < 1 ]]
        do
           ping -q -c1 172.20.100.160 > /dev/null
           if [ $? -eq 0 ] #exist connection
           then
           connection=$(( connection+1 ))
           else
           echo "No connection."
fi
       done

if [ $connection == 1 ]
then
   #kill all processes
   killall java &> /dev/null

   #write to a log file
   echo "---------------------------------" >>
      /home/pi/Desktop/logs/musec.txt date >>
      /home/pi/Desktop/logs/musec.txt

   #start aplication
   java -jar /home/pi/Desktop/dist/MUSEC.jar start
      172.20.100.160 root PASSWORD >>
      /home/pi/Desktop/logs/musec.txt
fi
```

Ambos os *scripts* (Listagens 5.3 e 5.4) foram inseridos no *crontab* do sistema. Assim conseguese o objetivo de os executar automaticamente no arranque do dispositivo. A Listagem 5.5 mostra as configurações utilizadas no *crontab*. A sonorização é assim automaticamente iniciada.

Listagem 5.5: *Crontab* Raspberry Pi 2 - iniciar MuSec.

```
#Iniciar automaticamente o MuSec (componente Java e Chuck)
@reboot /home/pi/Desktop/Start_Chuck_MuSec.sh
@reboot /home/pi/Desktop/Start_MuSec_ESTGF.sh
```

Depois de implementada a proposta de solução na ESTG, foram efetuados alguns testes para analisar o comportamento da mesma, perante situações diversas. Primeiramente, foi efetuado um teste com o

objetivo de verificar se o sistema reagia a todos os riscos possíveis, provenientes do servidor OSSIM. Aquilo que se pretendeu foi testar o MuSec, sob condições anormais de uso, e a sonorização de todos os níveis de riscos possíveis. Para tal sobrecarregou-se o OSSIM com tentativas de autenticação SSH. O OSSIM foi configurado para reconhecer este comportamento como ataques com a introdução de uma regra (*diretive*). A regra criada foi configurada com nível 5 de prioridade (prioridade máxima) e tinha como objetivo detetar ataques que usem uma autenticação *bruteforce* por SSH (ver Figura 5.7).

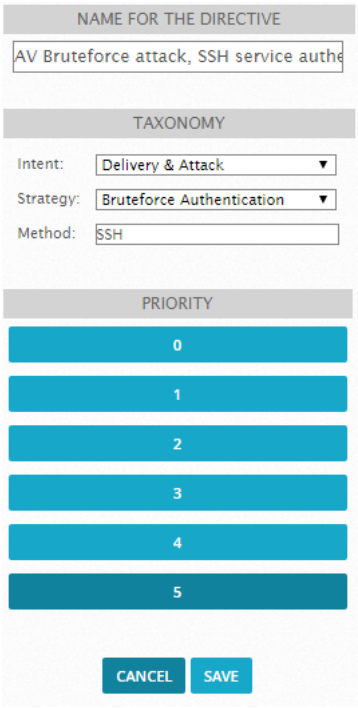

Figura 5.7: Definição da regra

A regra gera alertas, caso um utilizador falhe várias autenticações por SSH num determinado *host*. A confiabilidade sobe conforme o número de tentativas de autenticação falhadas, aumentando gradualmente o nível de risco, como é possível ver na Figura 5.8.

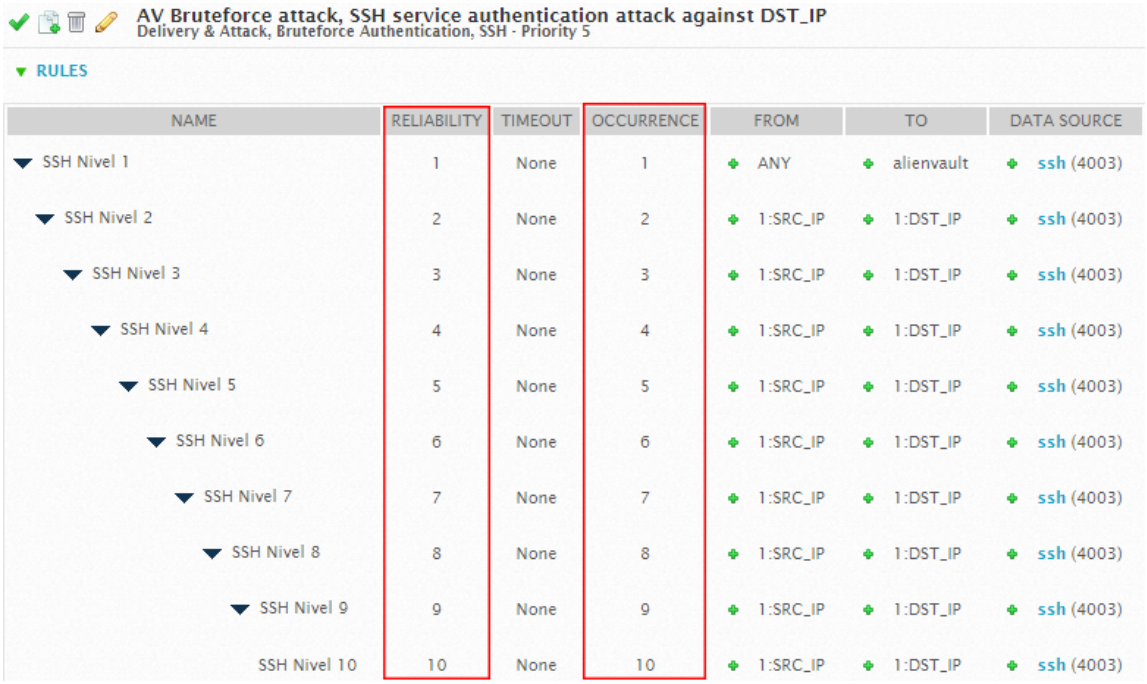

Figura 5.8: Regra SSH

Após a configuração do OSSIM, foi elaborado um *script* para fazer uso da regra adicionada (ver Listagem 5.6. O *script* escrito em *bash*, segundo a segundo, tenta automaticamente ligar-se via SSH a uma maquina predefinida. Este *script* foi executado durante 15 minutos.

Listagem 5.6: *Script* para testar sonorização

```
1  #!/bin/bash
2
3  function alarmes(){
4     while [ true ]
5           do
6               expect_sh=$(expect -c "
7               spawn ssh-keygen -R $1
8               spawn ssh root@$1
9               expect \"Are you sure you want to continue
                   connecting (yes/no)?\"
10              send \"yes\r\"
11              expect \"password:\"
12              send \"password\r\"
13              expect \"password:\"
14              send \"password\r\"
15              expect \"password:\"
16              send \"password\r\"
17              send \"exit\r\"
18              ")
19              #run the expect script
```

```
20              echo "$expect_sh"
21          echo "A gerar todos os niveis de alarme no OSSIM..."
22          done
23  }
24
25  #menu
26  PS3="Introduza a opcao: "
27  options=("Gerar todos niveis" "Sair")
28  select opt in "${options[@]}"
29  do
30      case $opt in
31          "Gerar todos niveis")
32                  alarmes '172.20.100.160'
33                  ;;
34          "Sair")
35                  break
36                  ;;
37          *) echo "Opcao invalida...";;
38      esac
39  done
```

Durante o teste foi possível constatar a subida gradual do risco no OSSIM e, consequentemente, foi possível ouvir o MuSec a sonorizar cada nível de risco, conforme o esperado. Foi possível ouvir a sonorização dos vários riscos, assinalando as tentativas falhadas de autenticação SSH, durante todo o tempo do teste.

A estabilidade da operação prolongada do MuSec foi também alvo de testes. Foram desenvolvidos vários *scripts* para recolher informações do estado do sistema (Listagem 5.7), dos processos utilizados pelo MuSec (Listagem 5.8) e ainda dos processos que usam mais memória e mais processador no sistema operativo (Listagem 5.9). Estes foram configurados para serem executados no arranque do sistema. A Listagem 5.7 apresenta o *script* que permite obter informação sobre a memoria, o disco e o processador usado num determinado momento.

Listagem 5.7: Script- Estado do sistema Raspbian

```
1  #!  /bin/bash
2  printf "Date\t\t\t\tMemory\t\tDisk\t\tCPU\n" >>
       /home/pi/Desktop/logs/stats.txt
3
4  while [ true ]; do
5  MEMORY=$(free -m | awk 'NR==2{printf "%.2f%%\t\t", $3*100/$2 }')
6  DISK=$(df -h | awk '$NF=="/"{printf "%s\t\t", $5}')
7  CPU=$(top -b -n2 | grep "Cpu(s)"|tail -n 1 | awk '{print $2 + $4
       d}')
8  DATA=$(date)
```

```
9  echo "$DATA    $MEMORY$DISK$CPU%" >>
       /home/pi/Desktop/logs/stats.txt
10 sleep 5
11 done
```

O *script* apresentado na Listagem 5.8, monitoriza os dois componentes usados pelo MuSec (java e chuck). O processo java é referente ao componente Java MuSec e o processo chuck é referente ao componente Chuck MuSec.

Listagem 5.8: Script- Processos MuSec

```
1  #! /bin/bash
2
3  while [ true ]; do
4  date >> /home/pi/Desktop/logs/process_monitor.txt; ps -C java -o
       %cpu,%mem,cmd >> /home/pi/Desktop/logs/process_monitor.txt
5  ps -C chuck -o %cpu,%mem,cmd >>
       /home/pi/Desktop/logs/process_monitor.txt
6  echo "--------------------------------------------" >>
       /home/pi/Desktop/logs/process_monitor.txt
7  sleep 5
8  done
```

O *script* exposto na Listagem 5.9 permite obter informação sobre os processos mais usados no sistema operativo Raspbian.

Listagem 5.9: Script- Top Processos

```
1  #! /bin/bash
2  while [ true ]; do
3  date >> /home/pi/Desktop/logs/top_memory_cpu.txt; ps -eo
       pid,ppid,cmd,%mem,%cpu --sort=-%mem | head >>
       /home/pi/Desktop/logs/top_memory_cpu.txt
4  echo "--------------------------------------------" >>
       /home/pi/Desktop/logs/top_memory_cpu.txt
5  sleep 5
6  done
```

Todos estes *scripts*, criam ficheiros de *log* que contêm as informações recolhidas no *Raspberry Pi 2*. Foi ainda criado um *script* que envia, diariamente, todos esses *logs* para um email concebido para o efeito (ver Listagem 5.10).

Listagem 5.10: Script- Envio de *logs* do MuSec.

```
1  #! /bin/bash
2  name=$(date '+%d-%m-20%y')
3  tar -zcvf /home/pi/Desktop/"$name.tar.gz" /home/pi/Desktop/logs
4  sleep 5
```

```
5 sudo echo "Backup diario" | mutt -s "Backup do Pi2 ESTG" -a
     /home/pi/Desktop/$name.tar.gz -- musecestg@gmail.com
6 sleep 10
7 sudo rm /home/pi/Desktop/$name.tar.gz
8 sudo rm -rf /home/pi/Desktop/logs
9 sudo mkdir -p /home/pi/Desktop/logs
```

Os *scripts* para a recolha e envio por email de ficheiros de *log*, foram também associados no *crontab* do sistema, (ver Listagem 5.11). A monitorização do MuSec e do comportamento do sistema operativo em causa, é iniciada assim que o dispositivo arranca. Todos os dias à meia noite, são enviados por email, os *logs* recolhidos.

Listagem 5.11: *Crontab* Raspberry Pi 2 - *logs* recolhidos.

```
1  #Iniciar scripts automaticamente no arranque do sistema para recolha de logs
2
3  #script estado do sistema
4  @reboot /home/pi/Desktop/stats.sh
5
6  #script processos MuSec
7  @reboot /home/pi/Desktop/process_monitor.sh
8
9  #script top processos
10 @reboot /home/pi/Desktop/top_memory_cpu.sh
11
12 #Enviar os logs por email
13 0 0 * * * /home/pi/Desktop/backup_logs.sh >/dev/null 2>&1
```

O MuSec foi testado e monitorizado durante um mês, no centro informático da ESTG, instalado num *Raspberry Pi 2*, com o sistema operativo Raspbian Jessie de 32 bits. De notar que durante a monitorização, o *Raspberry Pi 2* reiniciou várias vezes devido a falta de energia elétrica, mas conseguiu sempre reiniciar a sua tarefa de sonorização automaticamente. Funcionou perfeitamente sem falhas, sendo possível ouvir a sonorização do OSSIM, durante todo o tempo de teste, sem problemas.

5.3 Conclusão

A proposta de solução funciona em vários sistemas operativos (Windows, Linux e Mac OS X) tanto em modo gráfico, como em modo linha de comandos. Em modo gráfico é apresentada uma interface gráfica simples, desenvolvida em JavaFX. No modo linha de comandos, basta executar o jar da proposta de solução com argumentos de acesso ao servidor OSSIM. A proposta de solução foi testada com sucesso em Windows 10 a 64 bits, Linux Mint a 32 bits e Mac OS X El Capitan a 64 bits. A proposta de solução foi ainda implementada na ESTG, num *Raspberry Pi 2* com sucesso. Para tal foram criados *scripts*, que são executados na *crontab* do sistema, na altura do arranque do mesmo. Por fim foram apresentados os resultados dos testes de carga e estabilidade da proposta de solução.

Capítulo 6

Conclusão

A monitorização de uma rede complexa é uma tarefa árdua e muito custosa. Impede a realização de outras tarefas ao mesmo tempo, pois necessita sempre de atenção contínua por parte do administrador de rede. A solução proposta visa dotar o administrador da rede, de uma forma alternativa de monitorização usando processos de sonorização de alertas provenientes de um SIEM. A ideia é simplificar a tarefa do administrador e permitir-lhe a realização de outras tarefas enquanto continua a monitorizar ativamente a rede, através da sonorização de dados provenientes do OSSIM. Essa sonorização permitirá ao administrador tirar ilações rápidas sobre o que se está a passar na rede.

6.1 Revisão do trabalho

O principal objetivo desta tese é permitir a sonorização de eventos gerados por um SIEM, mais propriamente do OSSIM. O Capítulo 2 endereça o tema gestão de eventos de segurança da informação e explica o que é um SIEM. São apresentadas as principais atividades e recursos de um SIEM, a sua importância em certas organizações, as principais funcionalidades e seus módulos e ainda é exibida uma arquitetura geral sobre o funcionamento de um SIEM. Existem várias soluções SIEM a ter em conta, uma das quais é o OSSIM. O OSSIM foi considerado como sendo uma das melhores na categoria de soluções gratuitas e de código aberto. A formula que permite calcular o risco de um determinado evento, gerado pelo OSSIM, foi discutida e ainda foram apresentadas as principais ferramentas de rede incluídas no OSSIM. A implementação de um SIEM é aconselhada para empresas que dependam da rede para funcionar e ou sejam redes de grande dimensão ou complexas.

O Capítulo 3 foca-se na sonorização, apresentando as suas vantagens, desvantagens e limitações. Explica ainda como poderá ser aplicada num processo de monitorização, identificando fatores a serem considerados aquando da aplicação de um processo de sonorização. São identificadas também as tecnologias que podem ser aplicadas num processo de sonorização. Foi efetuado um levantamento de vários projetos e aplicações que usam sonorização em redes de computadores. Cada uma destas soluções utiliza abordagens distintas quer a nível da recolha dos dados a tratar quer a nível da forma como é feita a sonorização. No final do capítulo é apresentada uma tabela que permite fazer um resumo dos vários projetos apresentados.

O Capítulo 4 descreve a proposta de solução e os requisitos identificados para a mesma. Tanto os requisitos funcionais como os não funcionais foram apresentados. Foi descrita e traçada a metodologia de desenvolvimento adotada. É apresentada também a arquitetura e todas as tecnologias, protocolos

e linguagens usadas no processo de sonorização de eventos obtidos do OSSIM. Ao mesmo tempo, é evidenciada a forma como todas as tecnologias, protocolos e linguagens funcionam entre si, com o objetivo de aproveitar as capacidades auditivas de um ser humano para ajudar o administrador de rede no processo de monitorização desta. Pretende-se ajudar o administrador de rede, a realizar outras tarefas do seu quotidiano enquanto consegue monitorizar a rede com alguma acessibilidade. Foram ainda expostos os vários diagramas elaborados durante o desenvolvimento da proposta de solução, de que são exemplo, o diagrama de casos de uso, os diagramas de classes (componente Java MuSec e Chuck MuSec), os diagramas de atividades e ainda diagramas de sequência.

O Capítulo 5 valida a solução proposta, evidenciando que todos os requisitos da proposta de solução foram cumpridos. Foi demonstrado que a solução proposta funciona em Windows, Linux e Mac OS X, em modo gráfico através de um interface gráfico em JavaFX e em modo linha de comandos através da passagem de argumentos de acesso ao servidor OSSIM. A mesma também foi testada e implementada na ESTG. A implementação foi feita num *Raspberry Pi 2*, usando o sistema operativo Raspbian, com sucesso. Foram ainda feitos testes nessa implementação, no sentido de validar carga, a funcionalidade e a estabilidade do MuSec no *Raspberry Pi 2*.

6.2 Contribuições

A principal contribuição do trabalho agora desenvolvido consiste num sistema de monitorização da rede de uma instituição com recurso a um mecanismo de sonorização de eventos identificados por SIEM. Por ser uma monitorização através de sonorização, permite-se ainda que o administrador da rede execute outras tarefas em simultâneo. Não se conhece nenhuma outra solução que disponibilize esta funcionalidade, facto que permite considerar que a contribuição, é uma contribuição inovadora.

6.3 Trabalho futuro

No decorrer do trabalho apresentado nesta tese, foram identificadas algumas limitações que podem ser assumidas como direções futuras de pesquisa e de aplicação. A solução proposta apenas funciona com o OSSIM. O OSSIM foi identificado como um dos melhores SIEM grátis, contudo a compatibilização da aplicação MuSec com outros SIEM permitirá expandir a sua utilização.

A aplicação MuSec, apesar de ser multi-plataforma, funcionando em Windows, Linux, Mac OS X, não funciona em plataforma móveis como o Android ou iOS. O funcionamento em Android e ou iOS, permitiria ao administrador continuar a monitorizar a rede, através da sonorização dos eventos gerados por um SIEM, fora do seu gabinete de trabalho. Os dispositivos móveis permitem uma portabilidade que os desktop ou um Raspberry Pi, não oferecem. O desenvolvimento de uma aplicação simples para dispositivos Android e iOS, permitindo ao administrador escutar o resultado da sonorização em qualquer lugar na sua instituição de trabalho, permitiria uma melhor portabilidade e um maior número de plataformas e dispositivos compatíveis.

A variedade musical que é produzida atualmente pela aplicação MuSec não é tão vasta quanto o desejado inicialmente. Apesar da aplicação MuSec, permitir que o administrador de rede, através dos *loops* existentes, consiga captar o essencial sobre o estado da rede, a audição e repetição de certos *loops musicais*, poderá saturar o administrador. A solução passa por adicionar mais *loops* musicais, reproduzindo

mais e diferentes géneros de música. Outra solução passa pela escolha de outra tecnologia que permita uma sonorização com uma maior variedade musical.

Bibliografia

[1] M. Afzaal, C. Di Sarno, S. Dantonio, and L. Romano. An intrusion and fault tolerant forensic storage for a siem system. In *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pages 579–586, Nov 2012.

[2] Marco Alamanni. Ossim: A careful, free and always available guardian for your network. *Linux J.*, 2014(242), June 2014.

[3] Alienvault. Alienvault ossim: The world's most widely used open source siem, https://www.alienvault.com/products/ossim. Accessed: 2015-12-15.

[4] USM AlienVault. Usm 5.1-5.2 asset management guide, rev.2. https://www.alienvault.com/doc-repo/usm/asset-management/AlienVault-USM-5.1-5.2-Asset-Management-Guide.pdf, 2015. Accessed: 2016-02-17.

[5] Giner Alor-Hernández. *Frameworks, Methodologies, and Tools for Developing Rich Internet Applications*. IGI Global, 2014.

[6] João Alves. Gestão de eventos de segurança de informação siem. *Projeto Integrado, Licenciatura em Segurança Informática em Redes de Computadores, ESTGF, Politécnico do Porto*, nov 2015.

[7] Mark Ballora, Nicklaus A Giacobe, and David L Hall. Songs of cyberspace: an update on sonifications of network traffic to support situational awareness. In *SPIE Defense, Security, and Sensing*, pages 80640P–80640P. International Society for Optics and Photonics, 2011.

[8] S. Bhatt, P. K. Manadhata, and L. Zomlot. The operational role of security information and event management systems. *IEEE Security Privacy*, 12(5):35–41, Sept 2014.

[9] Sam Cury, Bret Hartman, David P Hunter, David Martin, Dennis R Morean, Alina Oprea, Uri Rivner, and Dana Elizabeth Wolf. Mobilizing intelligent security operations for advanced persistent threat. *RSA security brief, RSA*, 2011.

[10] Debian. Download page for firmware bnx2, https://packages.debian.org/jessie/all/firmware-bnx2/download. Accessed: 2016-06-23.

[11] Debian. Download page for firmware bnx2x, https://packages.debian.org/jessie/all/firmware-bnx2x/download. Accessed: 2016-06-23.

[12] K. O. Detken, T. Rix, C. Kleiner, B. Hellmann, and L. Renners. Siem approach for a higher level of it security in enterprise networks. In *Intelligent Data Acquisition and Advanced Computing*

Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on, volume 1, pages 322–327, Sept 2015.

[13] Gartner. Gartner magic quadrant: Leaders, visionaries, niche players and challengers, http://www.gartner.com/technology/research/methodologies/magicquadrants.jsp. Accessed: 2016.10.01.

[14] Michael Gilfix and Alva L Couch. Peep (the network auralizer): Monitoring your network with sound. In *LISA*, pages 109–117, 2000.

[15] Rudi Giot and Yohan Courbe. Intention–interactive network sonification. *Georgia Institute of Technology*, 2012.

[16] Thomas Hermann. *Sonification for Exploratory Data Analysis*. PhD thesis, Bielefeld University, 2002.

[17] Thomas Hermann. Taxonomy and definitions for sonification and auditory display. *International Community for Auditory Display*, 2008.

[18] Thomas Hermann, Andy Hunt, and John G Neuhoff. *The sonification handbook*. Logos Verlag Berlin, GE, 2011.

[19] D. Hermanowski. Open source security information management system supporting it security audit. In *Cybernetics (CYBCONF), 2015 IEEE 2nd International Conference on*, pages 336–341, June 2015.

[20] T. Hildebrandt and S. Rinderle-Ma. Server sounds and network noises. In *Cognitive Infocommunications (CogInfoCom), 2015 6th IEEE International Conference on*, pages 45–50, Oct 2015.

[21] Tobias Hildebrandt, Thomas Hermann, and Stefanie Rinderle-Ma. A sonification system for process monitoring as secondary task. In *Cognitive Infocommunications (CogInfoCom), 2014 5th IEEE Conference on*, pages 191–196. IEEE, 2014.

[22] IBM. Rational unified process: Best practices for software development teams, https://www.ibm.com/developerworks/rational/library/content/03july/1000/1251/1251_bestpractices_tp026b.pdf. Accessed: 2016-10-31.

[23] Ajay Kapur. *Programming for musicians and digital artists*. Manning Publ., 2015.

[24] Kelly M. Kavanagh and Oliver Rochford. Security information and event management: Magic quadrant, http://www.gartner.com/displaydocument?doc_cd=290113. Accessed: 2016.10.01.

[25] Alan Kebert, Bikramjit Banerjee, Glover George, Juan Solano, and Wanda Solano. Detecting distributed sql injection attacks in a eucalyptus cloud environment. In *Proceedings of the 12th International Conference on Security and Management (SAM-13), Las Vegas, NV, July*, 2013.

[26] Eva Kostrecová and Helena Bínová. Research paper security information and event management. *Management*, 4(2), 2015.

[27] R. Leszczyna and M. R. Wróbel. Evaluation of open source siem for situation awareness platform in the smart grid environment. In *Factory Communication Systems (WFCS), 2015 IEEE World Conference on*, pages 1–4, May 2015.

[28] Tim Lindholm, Frank Yellin, Gilad Bracha, and Alex Buckley. *The Java Virtual Machine Specification: Java SE 8 Edition*. Pearson Education, 2015.

[29] Delfina Malandrino, Daniela Mea, Alberto Negro, Giuseppina Palmieri, and Vittoro Scarano. Nemos: Network monitoring with sound. *Georgia Institute of Technology*, 2003.

[30] Vincent F Mancuso, Eric T Greenlee, Gregory Funke, Allen Dukes, Lauren Menke, Rebecca Brown, and Brent Miller. Augmenting cyber defender performance and workload through sonified displays. *Procedia Manufacturing*, 3:5214–5221, 2015.

[31] James McCartney. Supercollider , http://supercollider.github.io/. Accessed: 2016-10-31.

[32] E. Novikova and I. Kotenko. Analytical visualization techniques for security information and event management. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 519–525, Feb 2013.

[33] Oracle. Chapter 1: Introduction to the java sound api, https://docs.oracle.com/javase/8/docs/technotes/guides/sound/programmer_guide/chapter1.html. Accessed: 2016-10-31.

[34] Oracle. Javafx: Getting started with javafx, https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm. Accessed: 2016-08-10.

[35] J. Pavlik, A. Komarek, and V. Sobeslav. Security information and event management in the cloud computing infrastructure. In *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on*, pages 209–214, Nov 2014.

[36] Dave Phillips. An introduction to osc , http://www.linuxjournal.com/content/introduction-osc. Accessed: 2015-12-17.

[37] Miller Puckette. Pure data, http://puredata.info/. Accessed: 2016-10-31.

[38] Holger Schulze. Insider threat spotlight report. *Group Founder Information Security Community on LinkedIn*, 2016.

[39] Illposed Software. Osc protocol library written in java, http://www.illposed.com/software/javaosc.html. Accessed: 2015-12-17.

[40] Carlos Soto. Advanced persistent threats (apt) 101, http://www.tomsitpro.com/articles/advanced-persistent-threats-apt-101,2-526.html. Accessed: 2016-10-01.

[41] Email Tara. Fancy bear strikes again: Tennis star nadal's medical info leaked, http://www.infosecurity-magazine.com/news/fancy-bear-strikes-again-tennis/. Accessed: 2016-10-01.

[42] Kaspersky Lab's Global Research Analysis Team. The great bank robbery: the carbanak apt, https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/. Accessed: 2016-10-01.

[43] Paul Vickers, Chris Laing, and Tom Fairfax. Sonification of a network's self-organized criticality. *arXiv preprint arXiv:1407.4705*, 2014.

[44] Ge Wang. Chuck : Language specification , http://chuck.cs.princeton.edu/doc/language/. Accessed: 2015-12-17.

[45] Ge Wang. Chuck : Strongly-timed, concurrent, and on-the-fly music programming language , http://chuck.cs.princeton.edu. Accessed: 2015-12-17.

[46] Scott Wilson, David Cottle, and Nick Collins. *The SuperCollider Book*. The MIT Press, 2011.

[47] KatieAnna E Wolf and Rebecca Fiebrink. Sonnet: A code interface for sonifying computer network data. In *NIME'13—13th International Conference on New Interfaces for Musical Expression*, pages 503–506, 2013.

[48] David Worrall. Sonipy , http://www.sonification.com.au/sonipy/index.html. Accessed: 2016-10-31.

[49] David Worrall. Realtime sonification and visualisation of network metadata. *International Conference on Auditory Display*, 2015.

[50] Matthew Wright, Adrian Freed, Ahm Lee, Tim Madden, and Ali Momeni. Managing complexity with explicit mapping of gestures to sound control with osc. In *International Computer Music Conference*, pages 314–317. Citeseer, 2001.

[51] Matthew Wright, Adrian Freed, and Ali Momeni. Opensound control: State of the art 2003. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, NIME '03, pages 153–160, Singapore, Singapore, 2003. National University of Singapore.

[52] Woon Seung Yeo, Jonathan Berger, and Zune Lee. Sonart: A framework for data sonification, visualization and networked multimedia applications. In *Proceedings of the 2004 International Computer Music Conference*, pages 180–184, 2004.

Anexos

Anexo A

Instalação OSSIM na ESTG

A.1 Preparação

Antes da instalação do servidor OSSIM na Escola Superior de Tecnologia e Gestão, procedeu-se à preparação de uma máquina, que irá conter o mesmo. A máquina que foi disponibilizada pelo Centro de Informática foi uma IBM System x3550.

Tabela A.1: Configuração IBM System x3550

Quantidade	Componente
2x	Processador Xeon E5420/2.5GHz 1333/ QC/ 12MB 80W
8x	Memória 4GB PC5300 DDR2 FBD CL5
1x	Controlador ServeRAID 8k SAS Controller Dedicated Slot
1x	Disco IBM 1TB 7200 SATA 3.5"HS HDD
1x	Placa de rede Broadcom NetXtreme II 1000 Express G Ethernet
1x	Placa de rede PRO/1000 PT Dual Port Server Adapter Intel

Criou-se um dispositivo Universal Serial Bus (USB) de arranque com a imagem do OSSIM 5.2.5 de 64 bits. Durante a instalação, existiram alguns problemas de drivers, que foram evitados criando-se uma pasta firmware no dispositivo USB de arranque, com os drivers da placa de rede Broadcom Netxtreme II [10, 11].

A.2 Inicio

Depois de arrancar o IBM System x3550 pelo dispositivo USB, com o OSSIM a primeira coisa a fazer é escolher a instalação. Neste caso escolheu-se o OSSIM 5.2.5, como é possível ver na Figura A.1.

Figura A.1: Componente a instalar.

A seguir escolher o sistema operativo, selecionar a localização e o *layout* do teclado. Depois, escolher o IP de acesso ao servidor OSSIM, Figura A.2.

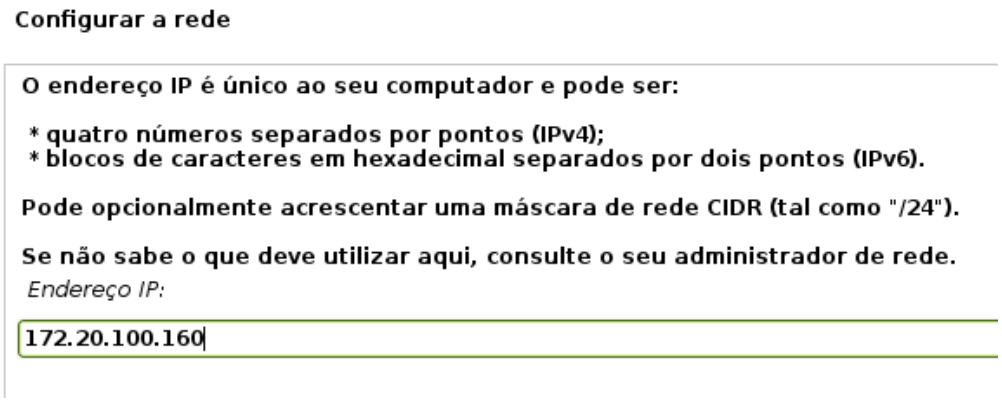

Figura A.2: Escolher o IP do OSSIM.

A seguir, escolher a máscara de rede, Figura A.3.

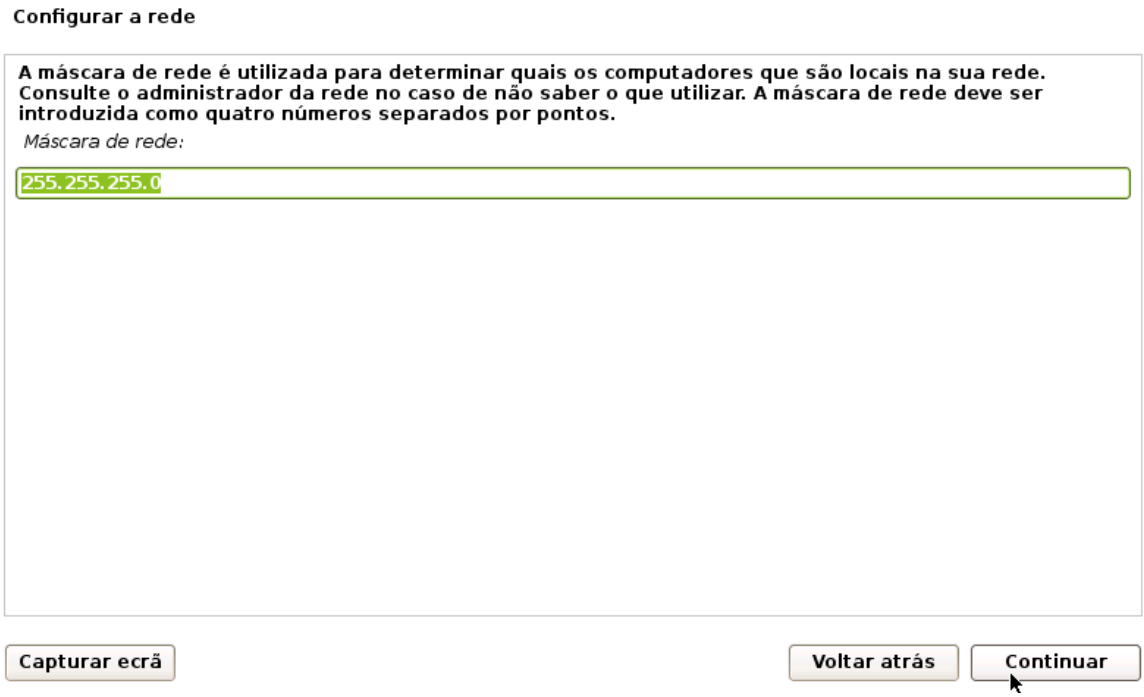

Figura A.3: Escolher a máscara de rede.

Depois, definir o gateway, Figura A.4.

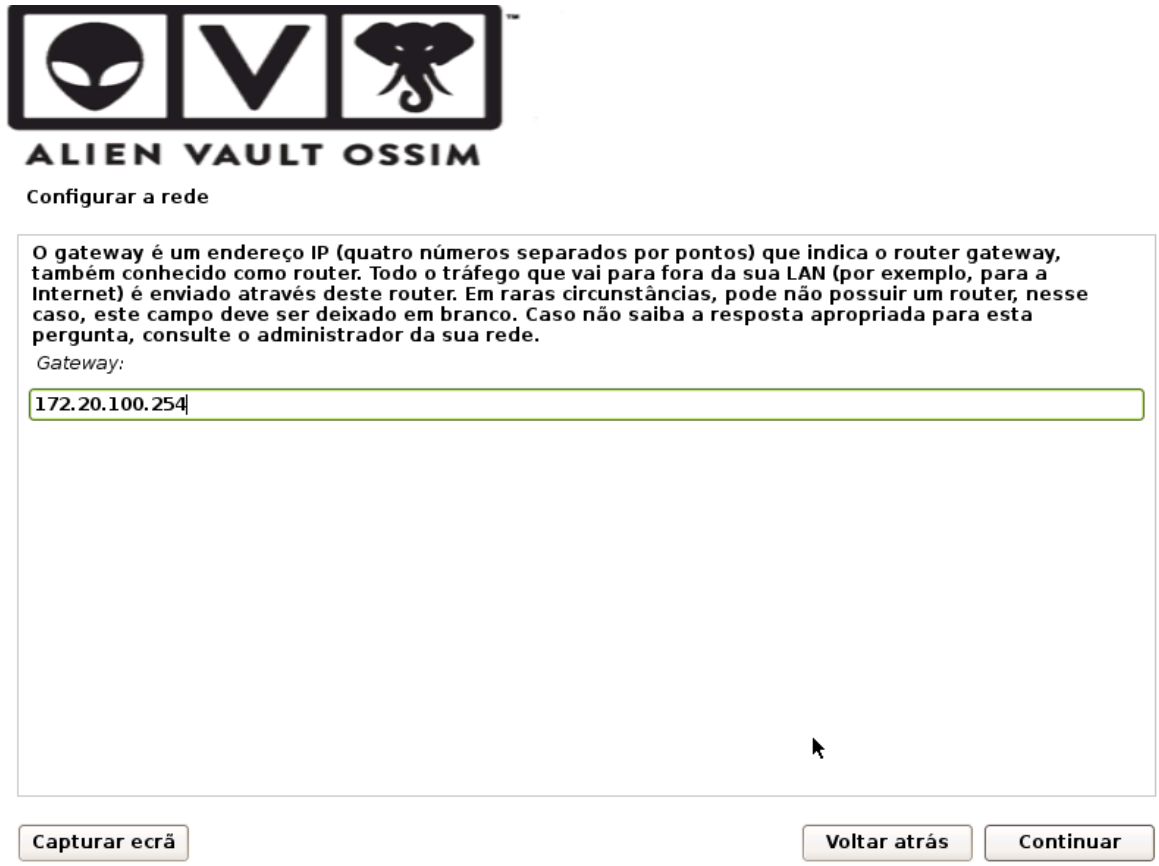

Figura A.4: Escolher o gateway.

Depois, definir o servidor de DNS da rede, Figura A.5.

Figura A.5: Escolher o servidor de DNS.

Depois, definir a password para o utilizador *root* e a seguir é iniciado o processo de instalação do servidor OSSIM na máquina, Figura A.6.

Figura A.6: A instalar o sistema.

Anexo B

Primeiras configurações

Depois de instalado o OSSIM, foram feitas umas pequenas configurações. O acesso ao servidor OSSIM, pode ser feito através de um browser, usando o IP definido na instalação, ou através da linha de comandos, como é possível ver na Figura B.1

Figura B.1: Acesso via sistema operativo.

Depois é preciso, configurar o sensor e que interfaces irão ficar em modo promíscuo, recebendo todo o tráfego da rede. Para isso basta fazer *login* com as credenciais de acesso root definidas aquando da instalação. Depois selecionar *Configure Sensor* (ver Figura B.2), *Configure Network Monitoring* (ver Figura B.3), e depois selecionar as interfaces pretendidas, como na Figura B.4

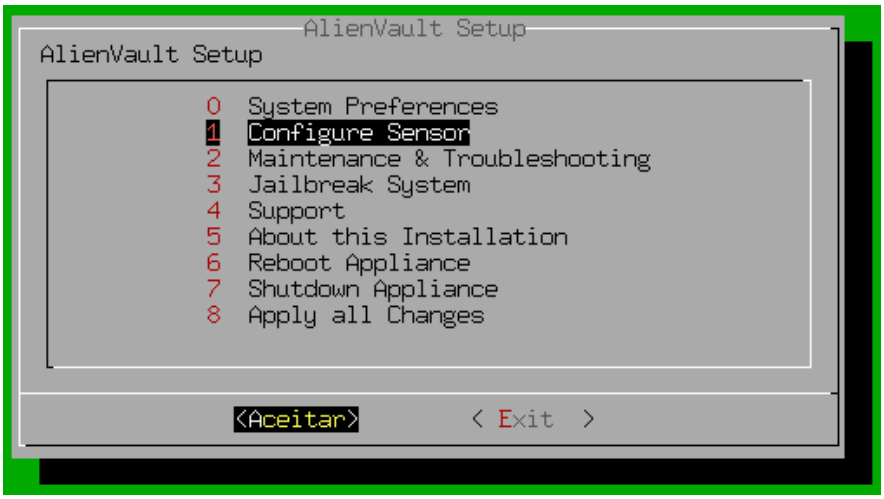

Figura B.2: Configurar o sensor.

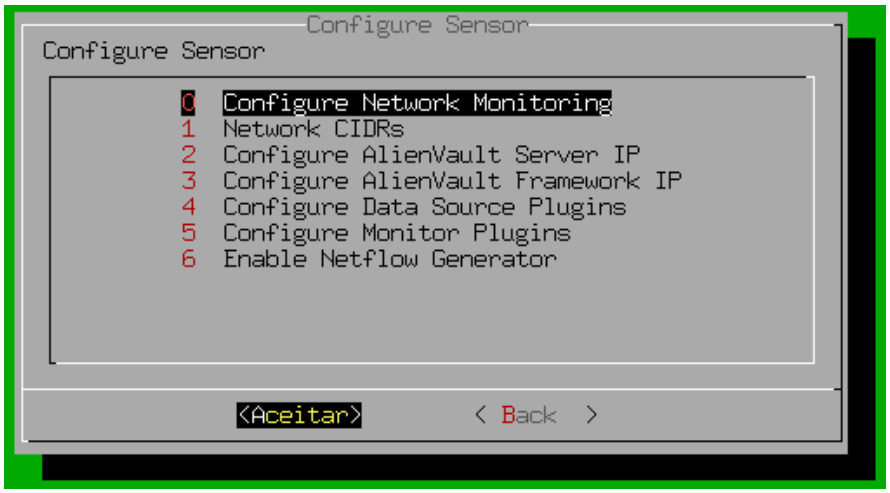

Figura B.3: Configurar a Monitorização da rede.

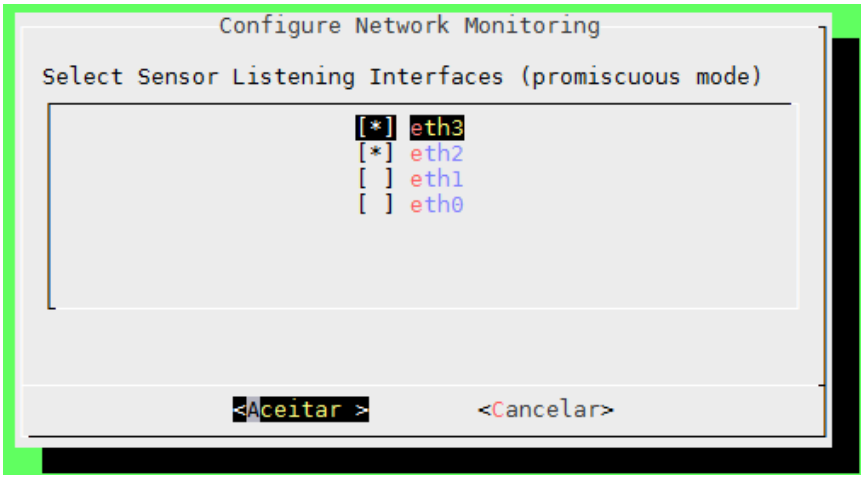

Figura B.4: Selecionar as interfaces de monitorização.

A seguir, é preciso configurar a interface de gestão usada pelo OSSIM. Para isso é necessário ir a *System Preferences* (ver Figura B.5), *Configure Network* (ver Figura B.6) e depois selecionar a interface

de gestão do servidor, como na Figura B.7.

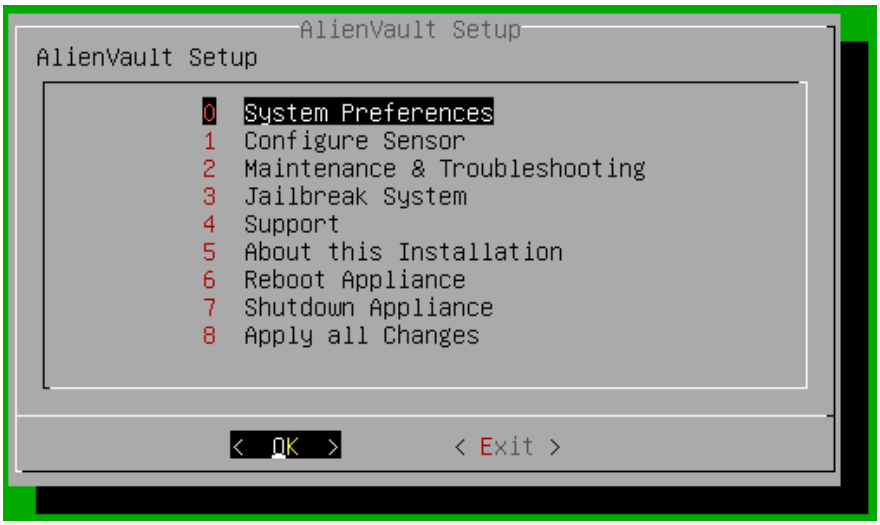

Figura B.5: Preferências de Sistema.

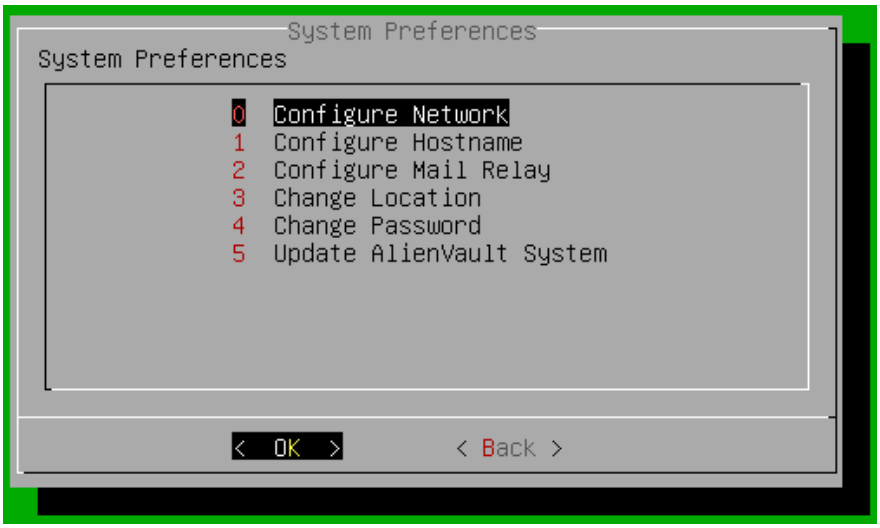

Figura B.6: Configurar a rede.

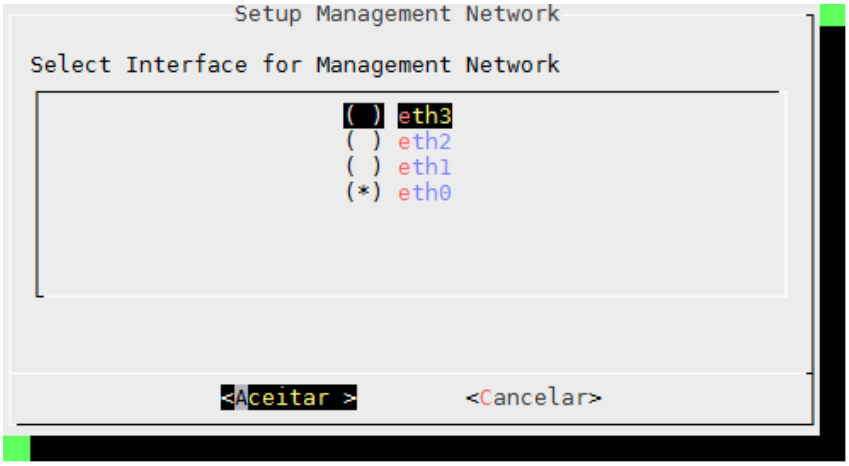

Figura B.7: Selecionar a interface de gestão.

Depois de se ter definido a interface de gestão e as interfaces de monitorização, agora é preciso aceder via web e fazer as primeiras configurações. Primeiro é preciso criar uma conta admin para aceder à plataforma do OSSIM via web browser, como é possível ver na Figura B.8. Depois é só fazer login, (ver Figura B.9) e irá ser iniciado um processo de configuração, como na Figura B.10.

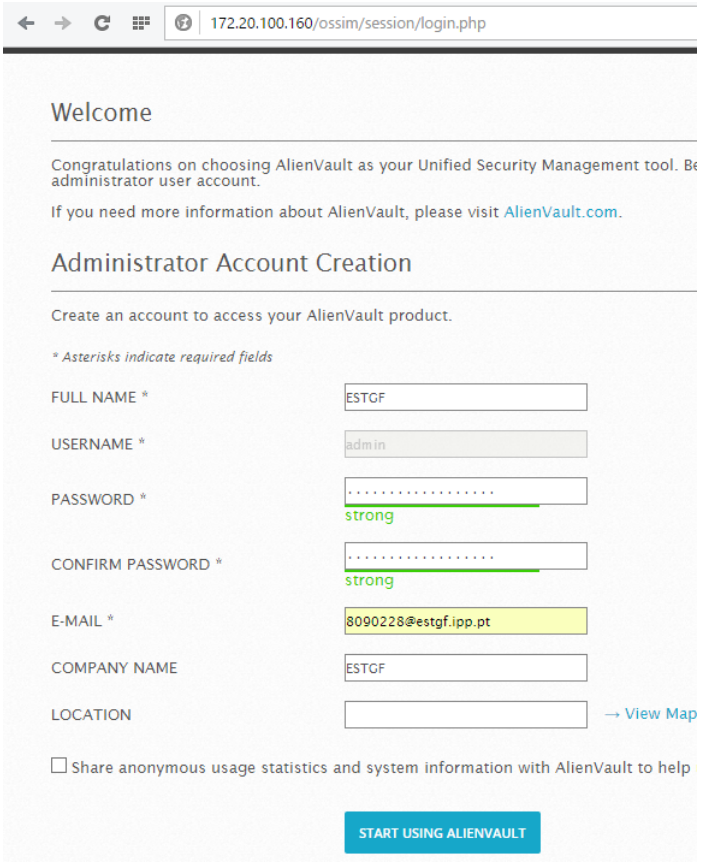

Figura B.8: Criar conta admin.

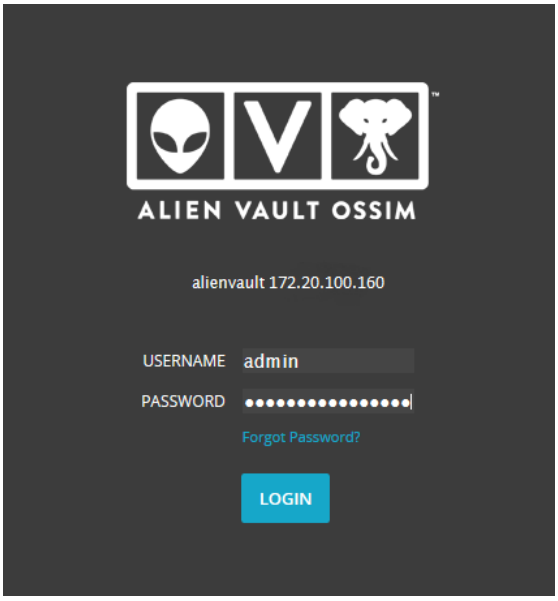

Figura B.9: Acesso via web browser.

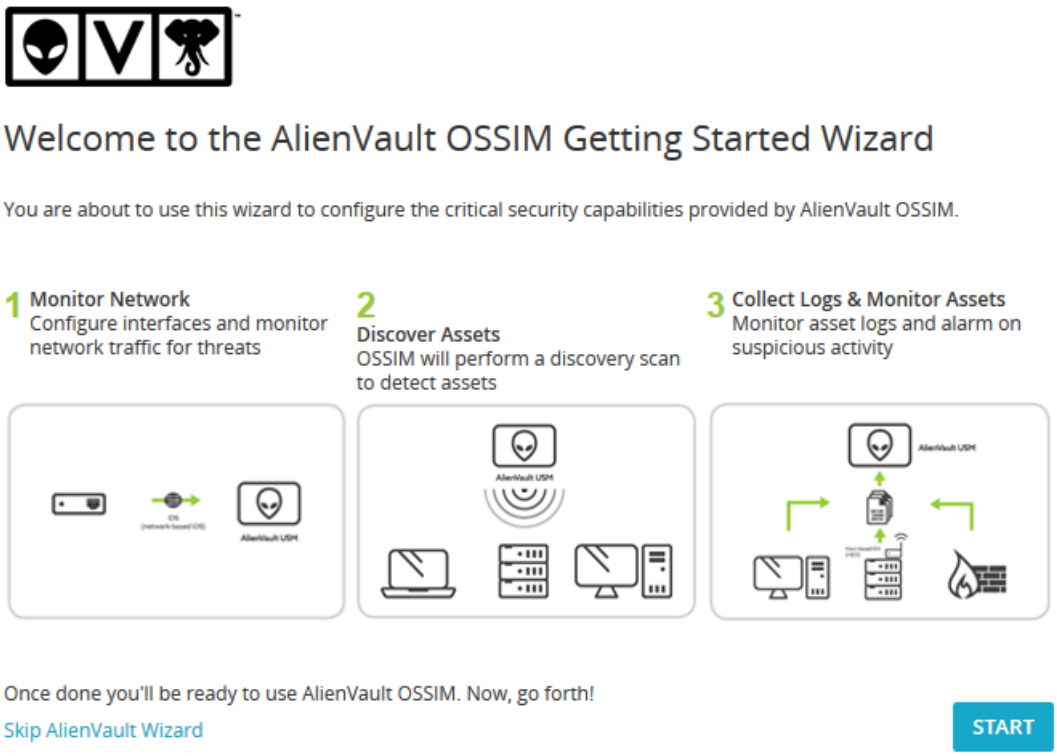

Figura B.10: Configuração do servidor OSSIM.

Primeiramente, é preciso selecionar as interfaces de gestão e de monitorização, que foram definidas anteriormente, como na Figura B.11.

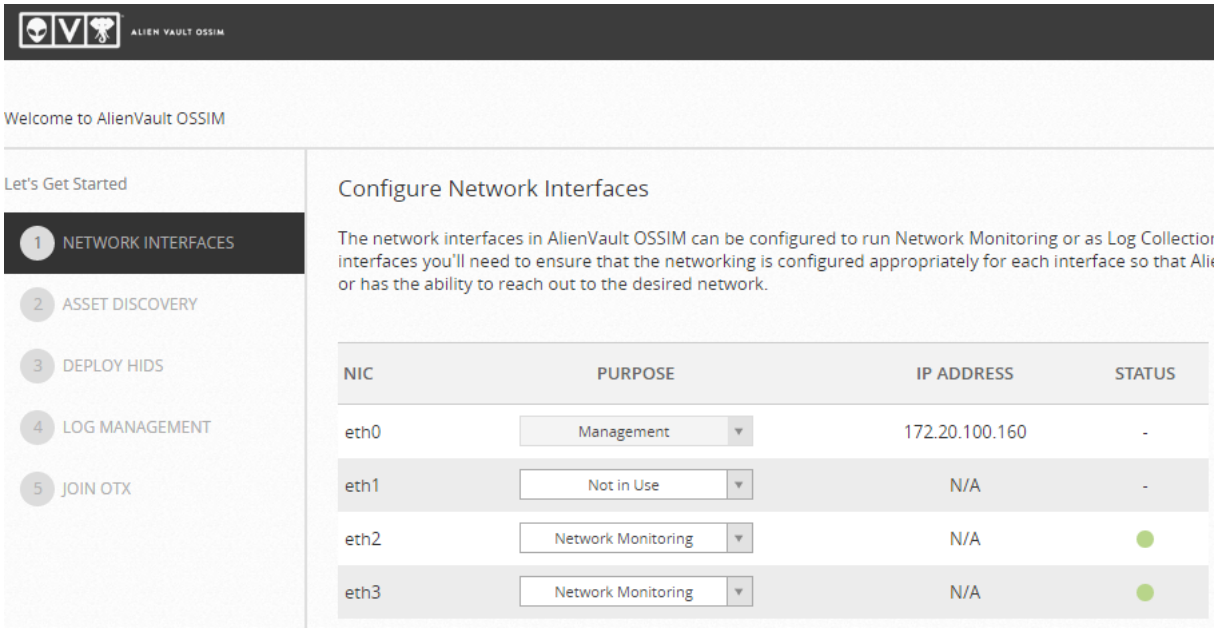

Figura B.11: Selecionar as interfaces de rede.

Depois é necessário localizar máquinas na rede, como na Figura B.12.

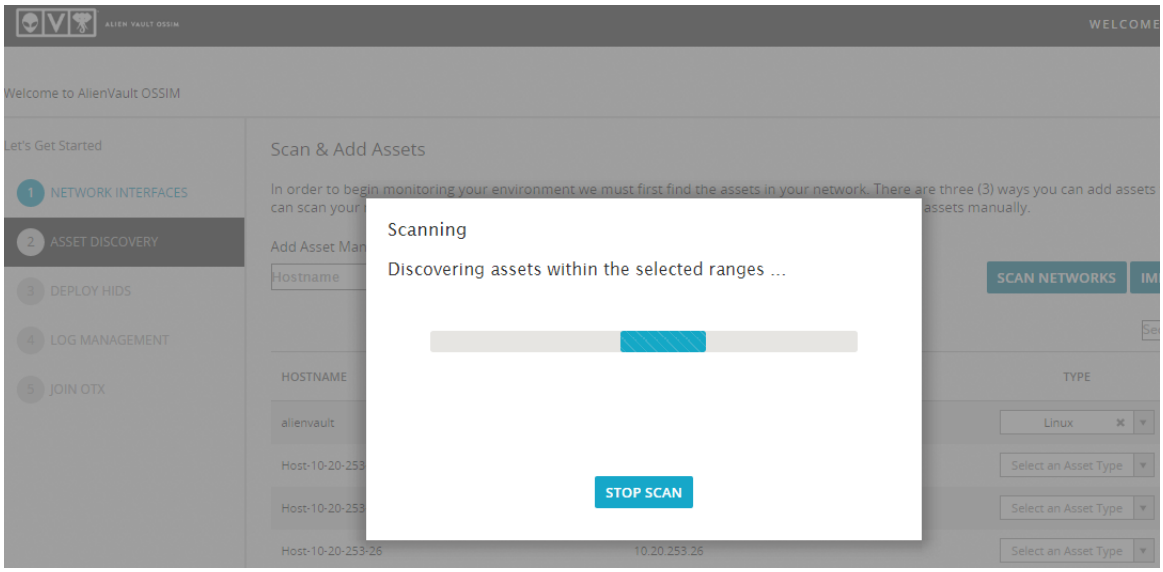

Figura B.12: Procurar máquinas na rede.

A seguir é possível aplicar agentes HIDS, para as máquinas encontradas, como na Figura B.13.

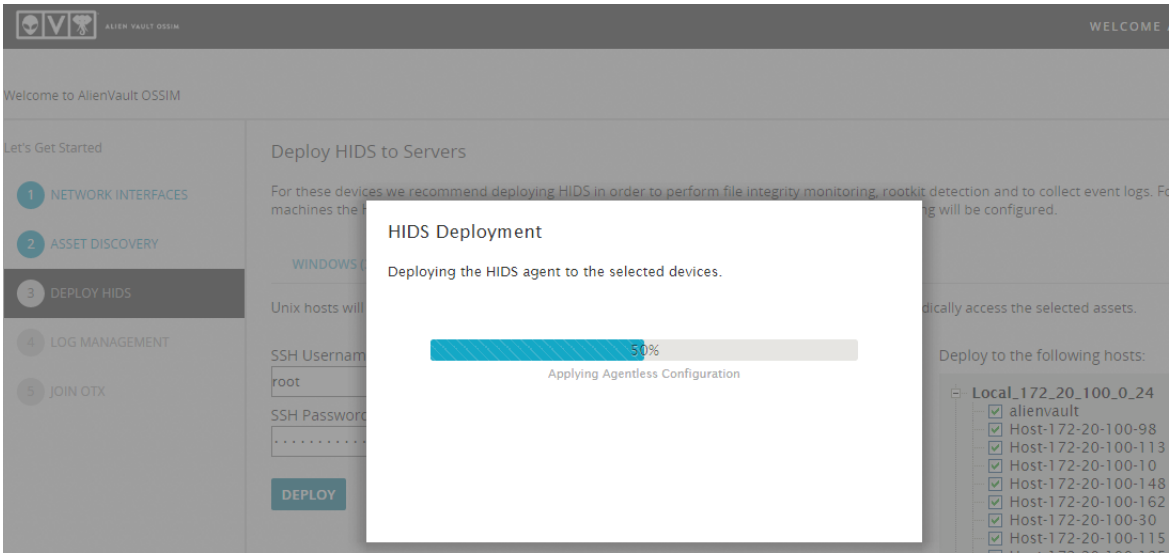

Figura B.13: Aplicar agentes HIDS.

Depois é possível definir a gestão de *logs* de determinados equipamentos e fabricantes, compatíveis com o OSSIM, como podemos ver na Figura B.14.

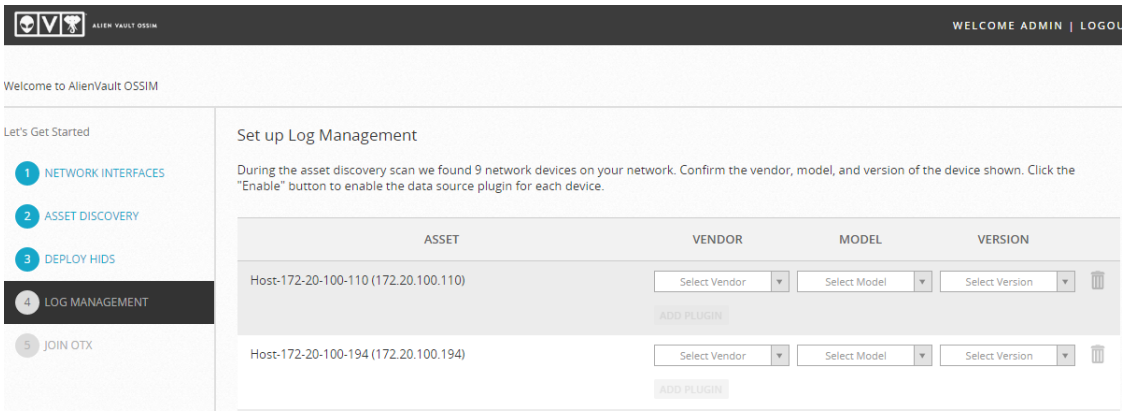

Figura B.14: Gestão de logs.

Depois é possível ligar o nosso servidor OSSIM, com a comunidade OTX da Alienvault, como é possível ver na Figura B.15. Está concluída a configuração inicial do OSSIM.

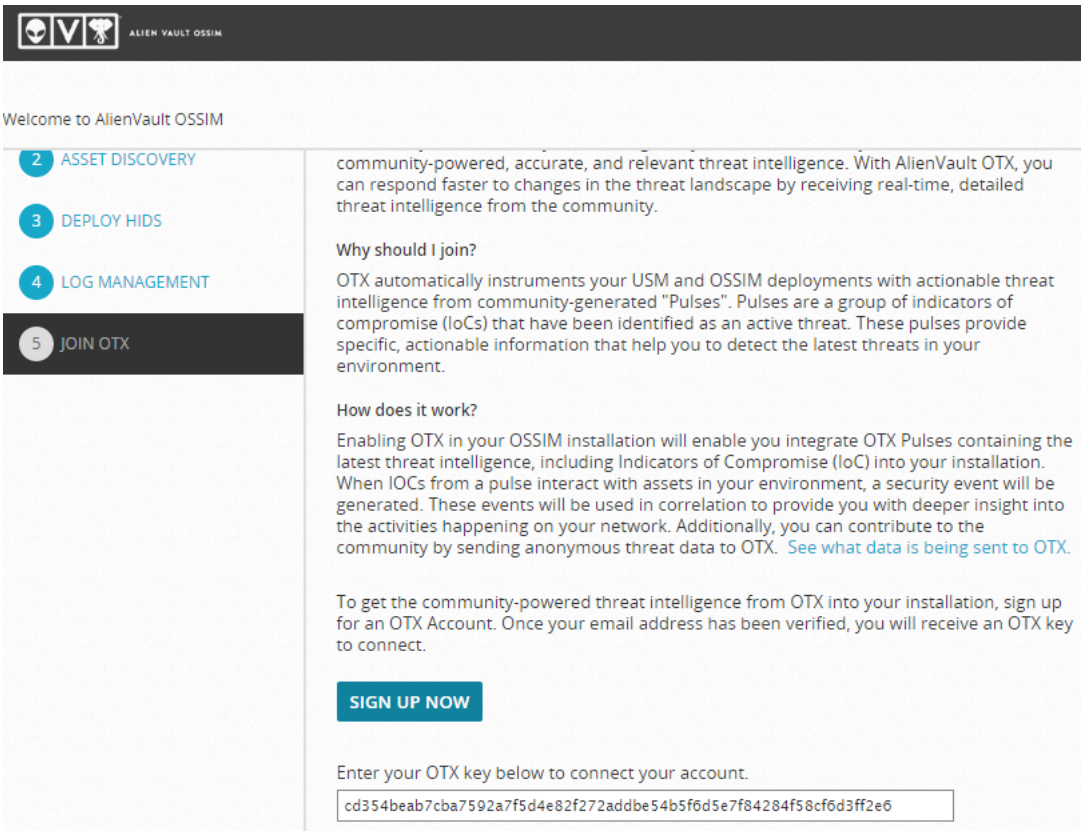

Figura B.15: Comunidade OTX da Alienvault.

Agora é possível aceder a um *dashboard* com informações sobre o que se está a passar na rede, como é apresentado na Figura B.16.

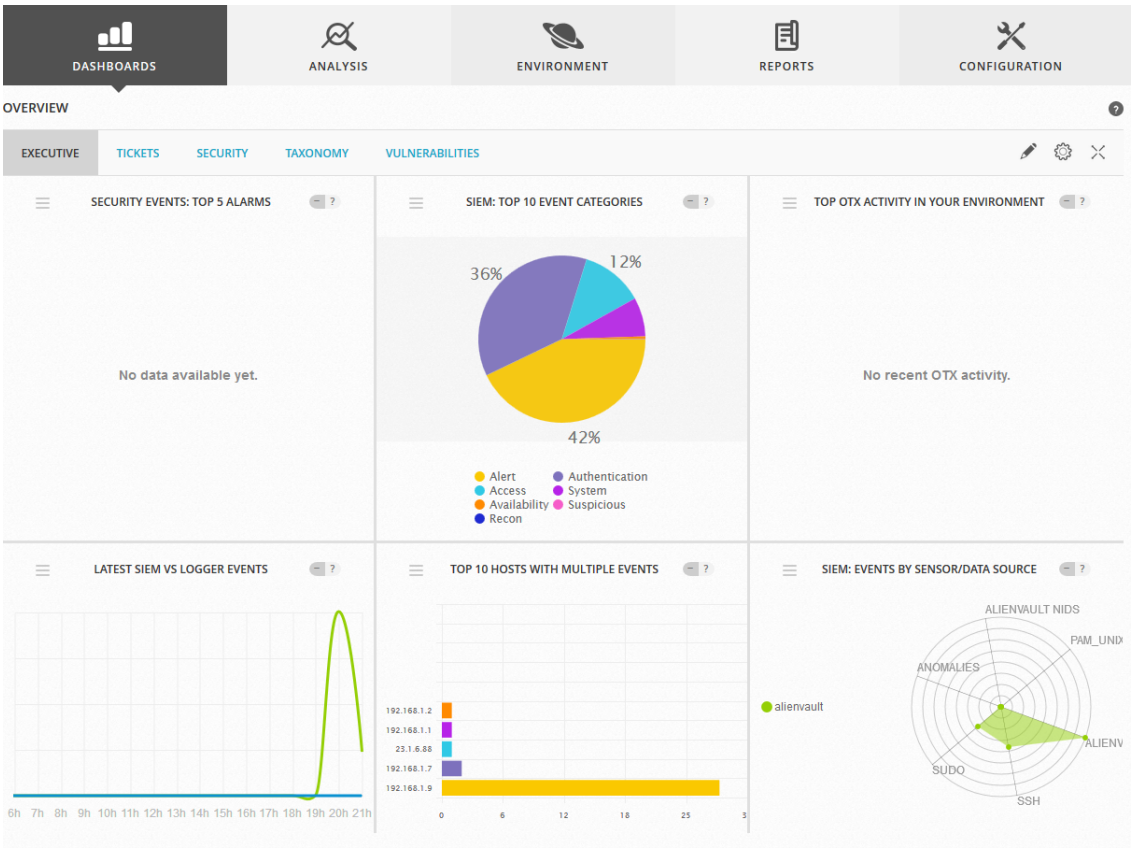

Figura B.16: Ecrã inicial do OSSIM.

Como é possível ver na Figura B.17, parece estar tudo a funcionar como o configurado e pretendido.

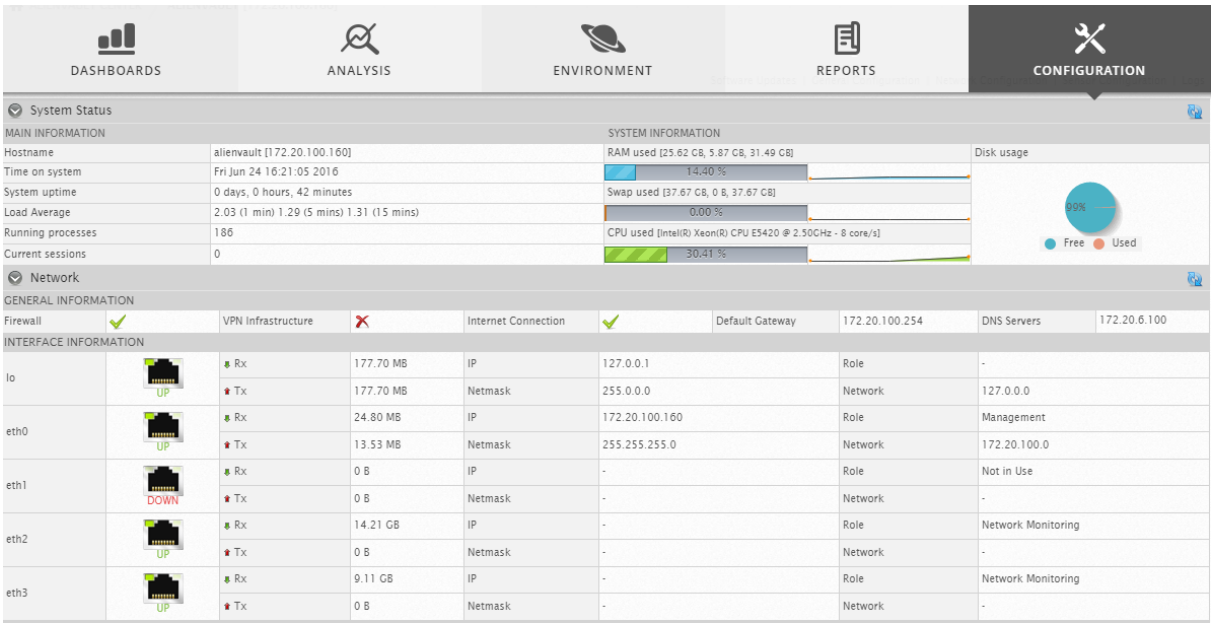

Figura B.17: Estado do servidor OSSIM.

Anexo C

Testes ao NIDS - Suricata

Na Figura C.1, é possível ver o estado do Suricata e outros componentes do OSSIM, permitindo ver que os mesmos estão ligados. Falta agora testar o desempenho deles, perante ataques, vulnerabilidades, etc. Foram feitos testes ao servidor OSSIM, principalmente ao NIDS que vem ativado por defeito, o Suricata. Para testar o seu funcionamento e o grau de deteção de intrusos na rede, foi usada a framework Pytbull. A framework Pytbull é uma framework de testes de IDS/Intrusion Prevention System (IPS) para ferramentas como o Snort, Suricata ou qualquer outro IDS/IPS que gere ficheiros de alerta. Usa uma arquitetura cliente-servidor e permite assim testar a capacidade de deteção dessas ferramentas. Permite realizar cerca de 300 testes agrupados em 11 módulos. Testes como de força bruta, nmap, DDoS, shellCodes, ficheiros pcap com tráfego duvidoso, reputações de IP, entre outros testes, são realizados.

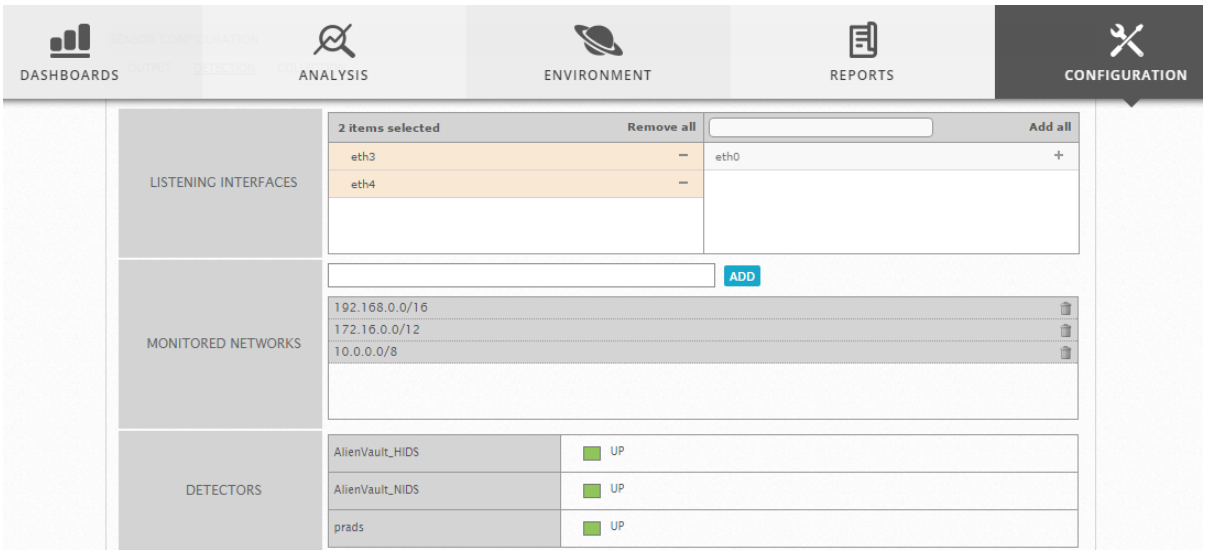

Figura C.1: Estados dos componentes.

Na Figura C.2, é possível ver o servidor e o cliente, pronto a testar o Suricata. Na Figura C.3 é possível ver alguns testes que a framework realizou. Nas Figuras C.4 e C.5 podemos ver que o Suricata detetou e gerou alertas para os testes realizados. Assim, podemos concluir que o OSSIM, está pronto a detetar intrusos na rede.

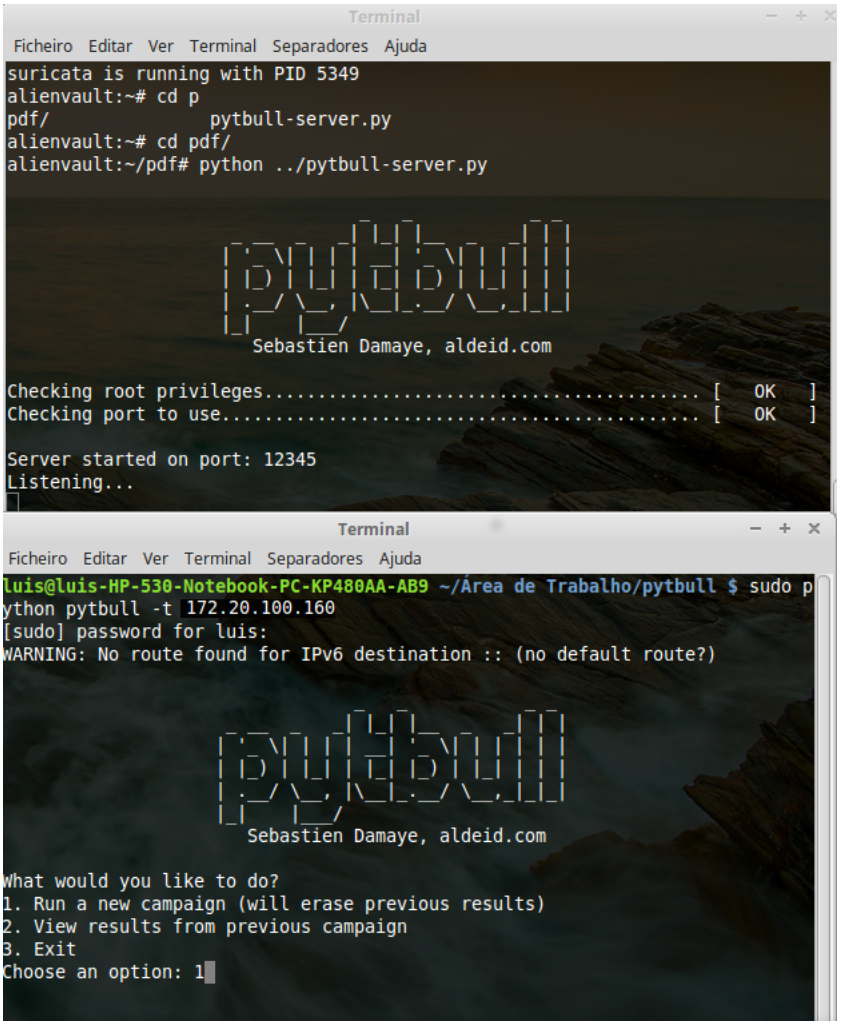

Figura C.2: Cliente e servidor Pytbull.

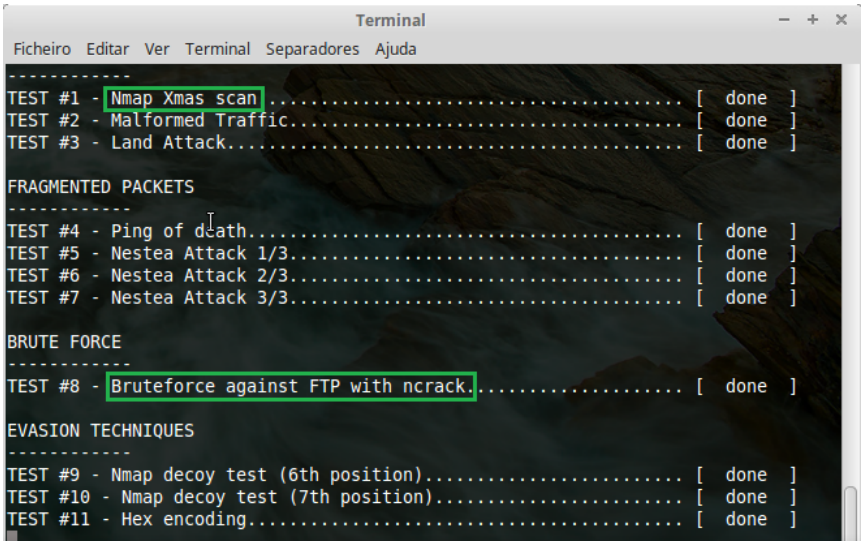

Figura C.3: Testes realizados pela Pytbull.

2016-04-11 11:24:00	AlienVault HIDS: Multiple failed logins in a small period of time.	0	AlienVault HIDS-authentication_failures	alienvault	N/A	0.0.0.0	alienvault
2016-04-11 11:24:00	directive_event: AV Bruteforce attack, login authentication attack against DST_IP	1	directive_alert	N/A	N/A	0.0.0.0	alienvault
2016-04-11 11:23:59	pam_unix: authentication failure	0	pam_unix	alienvault	N/A	computer-3	alienvault

Figura C.4: Teste de brute force.

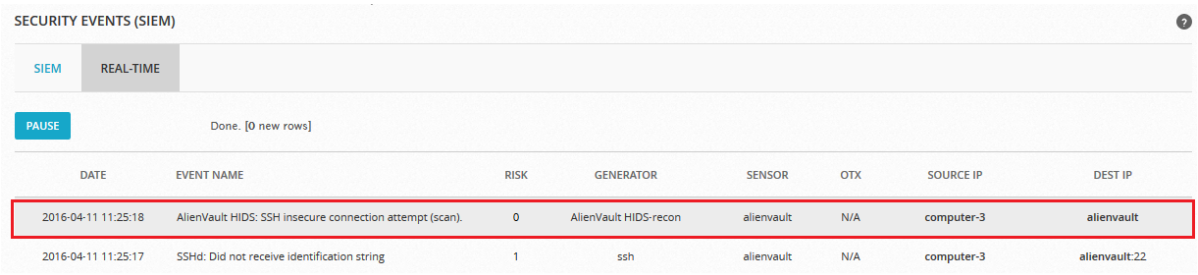

Figura C.5: Teste scan de portas.

Anexo D

Instalação de agentes HIDS

Foi instalado um agente HIDS, no servidor do moodle da ESTG, com a finalidade de: colecionar e ou monitorizar *logs*, verificar a integridade de ficheiros e ainda responder ativamente ao que se está a passar na máquina. Principais passos para a instalação:

- Aceder à máquina moodle, instalar o OSSEC HIDS 2.8.2 e indicar o IP do servidor OSSIM durante a instalação.
- Depois aceder ao servidor OSSIM, via web. Criar um agente com o IP da maquina a ser monitorizada, neste caso o IP do moodle. Para isso ir a **Environment - Detection - Agent - Add Agent**. De seguida, gerar uma chave, como é possível ver na Figura D.1. Copiar essa chave.
- De seguida aceder outra vez à máquina moodle e fazer o comando **sudo /var/ossec/bin/manage_agents**. Pressionar I e colar a chave anteriormente obtida.
- Reiniciar o ossec - **sudo /var/ossec/bin/ossec-control restart**.
- Por fim aceder ao servidor OSSIM via web e ir a **Environment - Detection - AlienVault HIDS Control - Botão Restart**.

Figura D.1: Chave gerada para o agente moodle.

Anexo E

Instalação do Raspbian

Um dos objetivos do desenvolvimento da aplicação MUSEC, é que a mesma possa funcionar num Raspberry. O Raspberry fornecido pelo centro informático da ESTG, foi o Raspberry Pi 2. A escolha do sistema operativo recaiu sobre o Raspbian Jessie de Maio de 2016. A preparação da instalação, foi feita numa máquina com Linux Mint 17.3 a 32 bits.

- Fazer o *download* do Raspbian Jessie. O ficheiro vem em formato .zip. Depois basta extrair o mesmo.
- Introduzir o cartão de memória no computador e descobrir onde foi montado, com o comando **df -h**.
- Por fim fazer usar o comando: **dd if="caminho da imagem com o Raspbian Jessie ex: /home/luis/Transferências/ 2016-05-27-jessie-raspbian.img"of=/dev/"caminho para o cartão de memória ex: sdc"bs=1M**.
- Retirar o cartão do computador e inserir o mesmo no Raspberry Pi 2.
- Ligar o Raspberry Pi 2. Pronto já está instalado o sistema operativo, como é possível ver na Figura E.1.

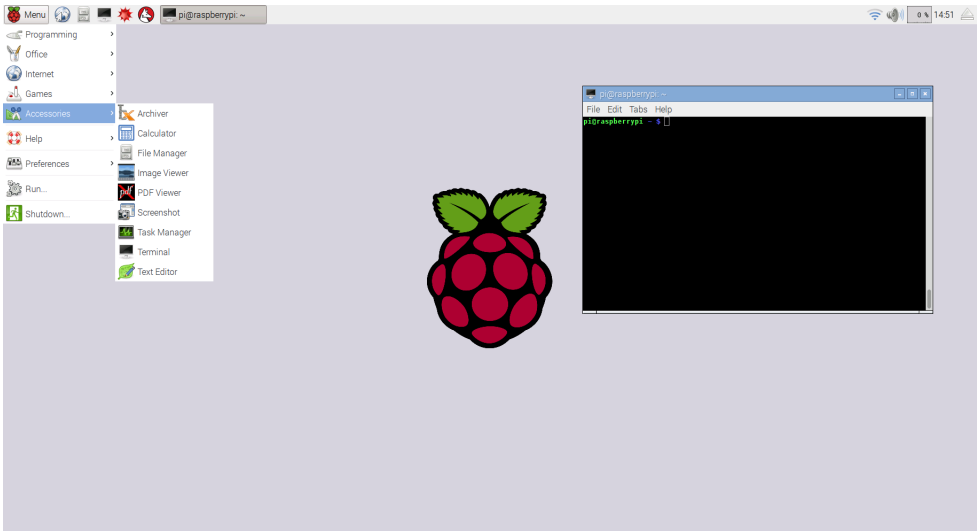

Figura E.1: Raspbian instalado no Raspberry Pi2.

Anexo F

Instalação das dependências

Como foi dito anteriormente, a aplicação pode ser usada em ambientes Windows, Mac OS e Linux. Assim, permite ao administrador de sistemas, instalar e usar a aplicação MuSec, independentemente do sistema operativo que o mesmo use. Contudo, devem ser instaladas algumas dependências:

- Oracle Java 8;
- Chuck 1.3.5.2;

A explicação seguinte, para a instalação do Java e do Chuck, foi realizada em ambiente Windows 10 de 64 bits, Linux Mint 17.3 de 32 bits e Mac OS X El Capitan de 64 bits.

F.0.1 Oracle Java 8

O Java é necessário, pois a proposta de solução, foi escrita maioritariamente em Java. O Java é fácil de instalar e é habitual o mesmo já se encontrar instalado em qualquer sistema. Recomenda-se a instalação da versão 8, pois foi a versão usada para o desenvolvimento da solução MuSec.

- Windows 10 64 bits:

  Fazer download da versão Java 8 para Windows, do site da Oracle, para a arquitetura 64 bits. Neste caso, foi feito o download do ficheiro, jdk-8u101-windows-x64.exe.

  Instalar normalmente o Java, clicando duas vezes no download realizado.

- Linux Mint 17.3 32 bits:

  Fazer download da versão Java 8 do site da Oracle, para a arquitetura 32 bits, neste caso, ficheiro jdk-8u101-linux-i586.tar.gz.

  Extrair e atualizar os caminhos para o Java. Para isso, fazer na linha de comandos:

Listagem F.1: Instalação do Java em Linux Mint

```
1 mkdir /opt/java && mv jdk-8u101-linux-i586.tar.gz /opt/java
2 cd /opt/java
3 tar -zxvf jdk-8u101-linux-x64.tar.gz
4 export JAVA_HOME=/opt/java/jdk1.8.0_101/
5 export JRE_HOME=/opt/java/jdk1.8.0._101/jre
6 export PATH=$PATH:/opt/java/jdk1.8.0._101/bin
```

- Mac OS X El Capitan de 64 bits:

  Fazer download da versão Java 8 para Mac OS X, do site da Oracle. Neste caso, foi feito o download do ficheiro, jdk-8u101-macosx-x64.dmg.

  Clicar duas vezes no ficheiro e instalar normalmente.

F.0.2 Chuck 1.3.5.2

A linguagem Chuck na versão 1.3.5.2, também tem de ser instalada, pois a proposta de solução, envia eventos para o componente Chuck os sonorizar.

- Windows 10 64 bits:

  Fazer download da instalação do Chuck 1.3.5.2 em:
  http://chuck.cs.princeton.edu/release/files/exe/chuck-1.3.5.2.msi

  Clicar duas vezes no ficheiro chuck-1.3.5.2.msi e seguir os passos de instalação do Chuck. Depois disto, o Chuck poderá ser executado, por linha de comandos, ou através de um IDE chamado MiniAudicle, como é possível ver na Figura F.1 .

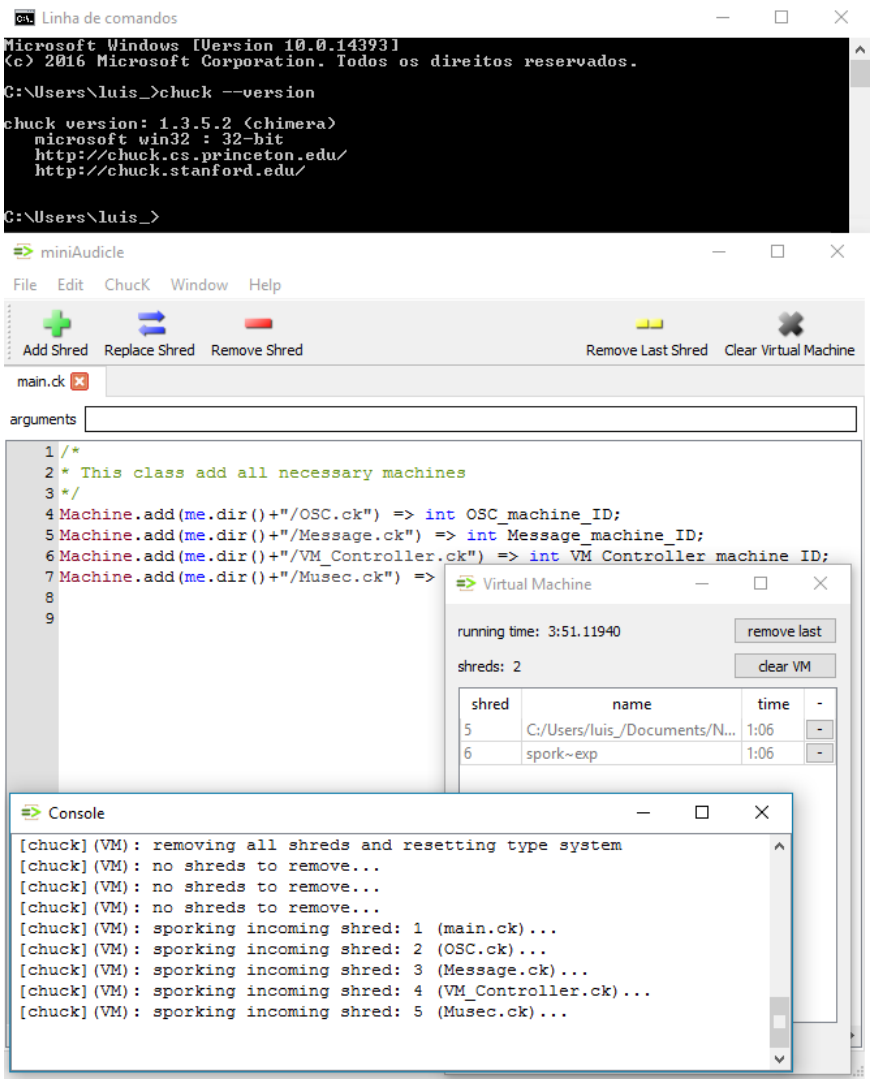

Figura F.1: Ambiente Chuck em Windows 10 x64.

- Linux Mint 17.3 32 bits:

  A instalação do Chuck em Linux, não é complicada, mas é feita através da linha de comandos.

  Fazer download do Chuck em:
http://chuck.cs.princeton.edu/release/files/chuck-1.3.5.2.tgz.

  Fazer download do IDE MiniAudicle através o endereço:
http://audicle.cs.princeton.edu/mini/release/files/miniAudicle-1.3.3.tgz. Na linha de comandos fazer os comandos explícitos na Listagem F.2.

Listagem F.2: Instalação do Chuck em Linux Mint

```
1 sudo apt-get install make gcc g++ bison flex libasound2-dev
     libsndfile1-dev libqt4-dev libqscintilla2-dev libpulse-dev
2 tar xzf chuck-1.3.5.2.tgz
3 tar miniAudicle-1.3.3.tgz
4 cd  chuck-1.3.5.2./src
5 make linux-pulse
6 sudo make install
7 cd miniAudicle-1.3.3/src
8 make linux-pulse
9 sudo make install
```

  Agora o Chuck pode ser executado, por linha de comandos, ou através de um IDE chamado MiniAudicle (ver Figura F.2).

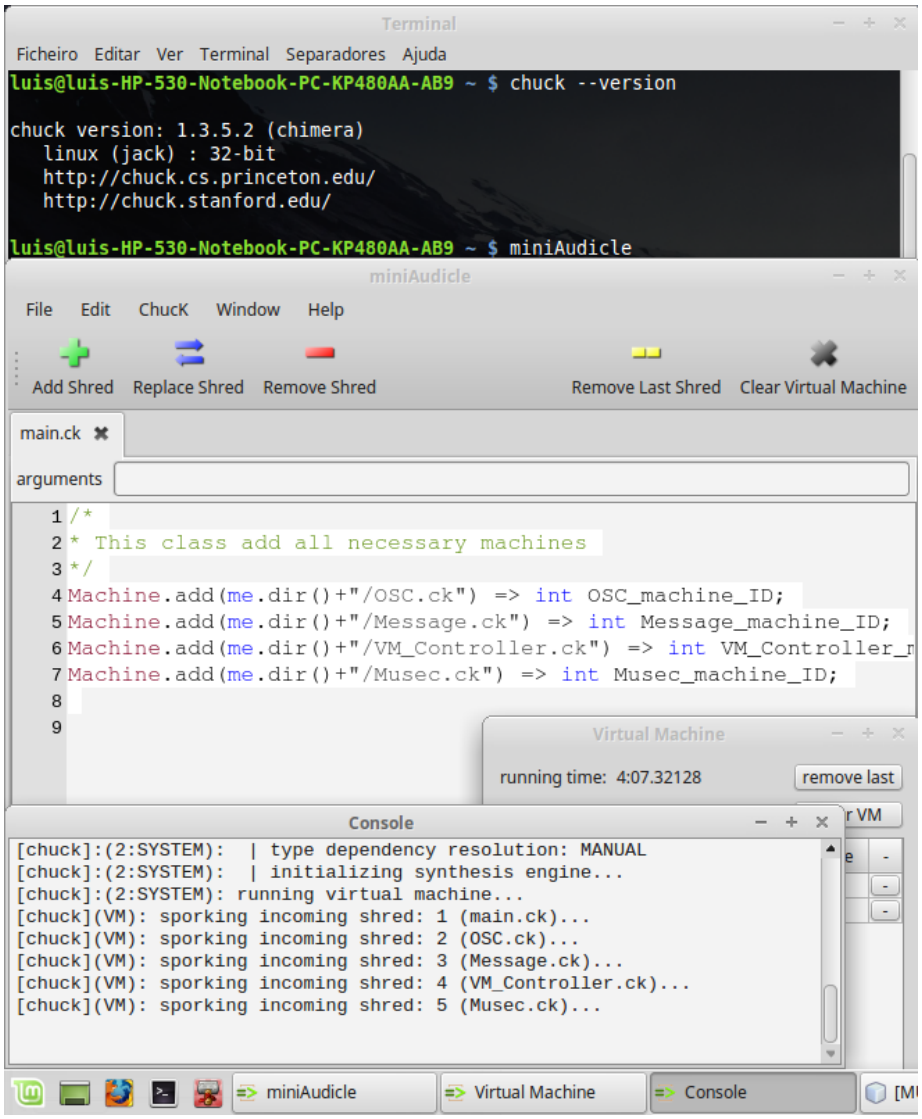

Figura F.2: Ambiente Chuck em Mint 17.3 x86.

- Mac OS X El Capitan de 64 bits:

  Ir ao web site: http://chuck.cs.princeton.edu/release/;

  Fazer download da instalação do Chuck 1.3.5.2, através do endereço: http://chuck.cs.princeton.edu/release/files/exe/chuck-1.3.5.2.pkg;

  Clicar duas vezes no ficheiro chuck-1.3.5.2.pkg, anteriormente descarregado. De seguida, instalar o Chuck normalmente. Agora o Chuck pode ser executado, por linha de comandos, ou através de um IDE chamado MiniAudicle (ver Figura F.3).

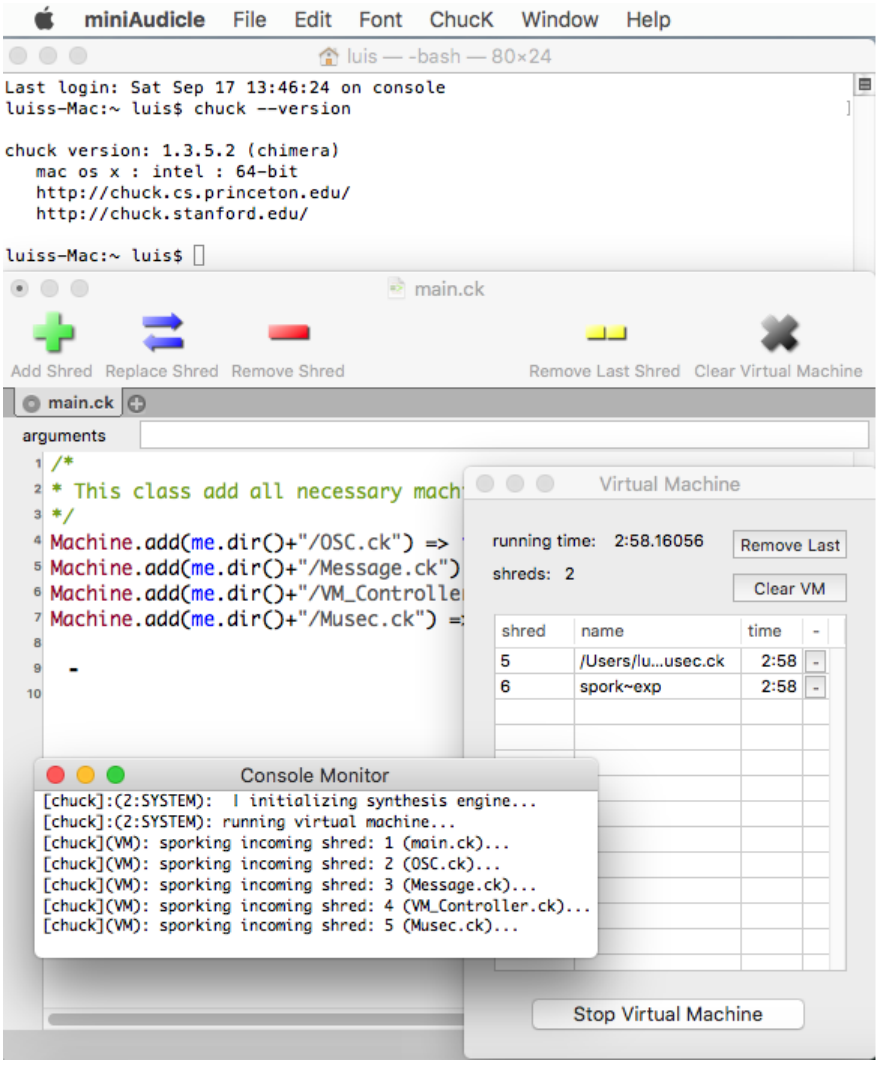

Figura F.3: Ambiente Chuck em Mac OS X El Capitan x64.

Anexo G

Diagrama de classes MuSec

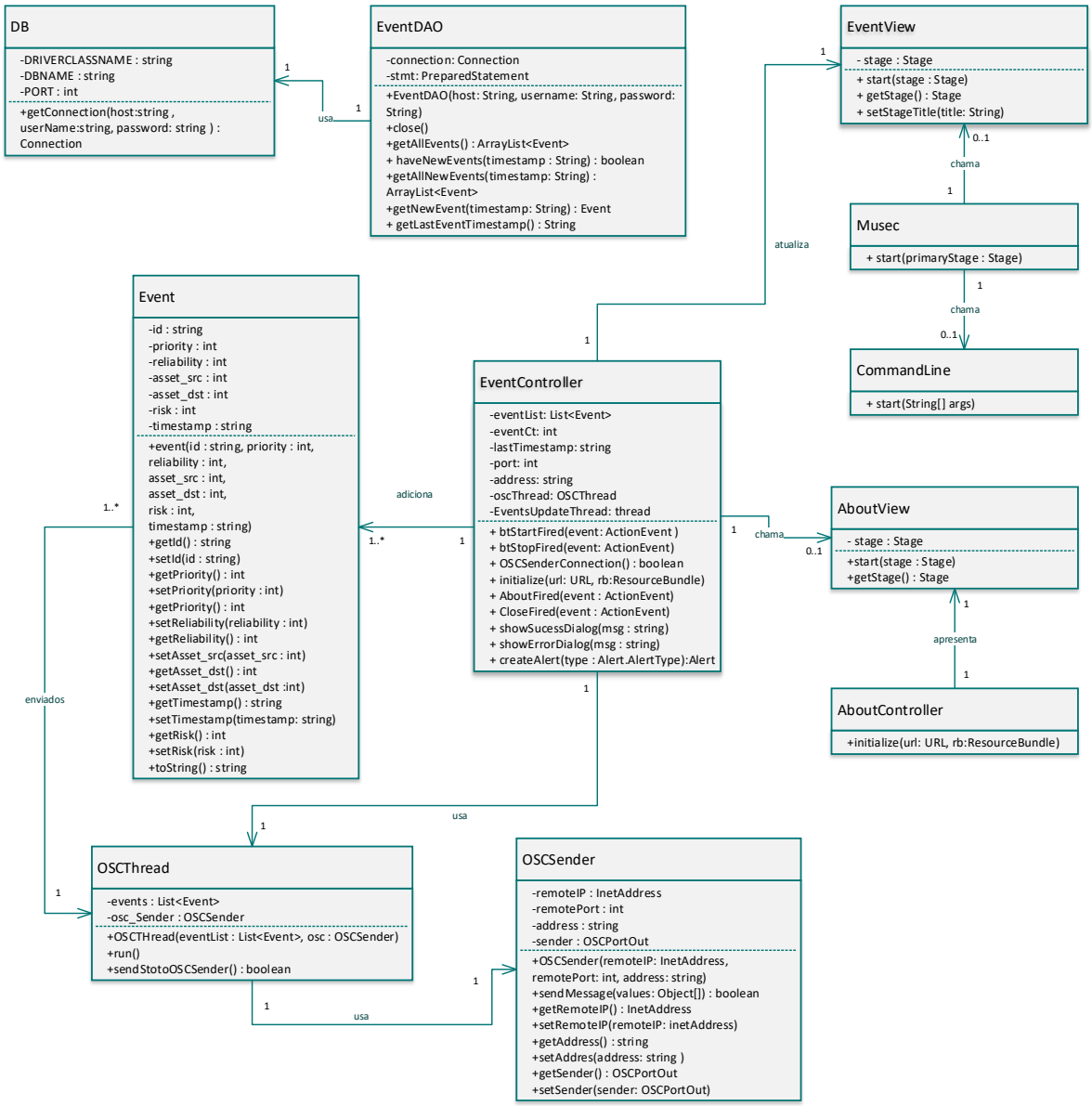

Figura G.1: Diagrama de classes completo.

Anexo H

Testes ao MuSec

H.1 Testes unitários

H.1.1 Objetivo

Testar isoladamente as várias classes e os vários métodos da componente Java MuSec.

H.1.2 Pré condições

Projeto com o código de desenvolvimento da componente Java MuSec e a *Framework* JUnit. Ligação na rede do servidor OSSIM.

H.1.3 Método utilizado

Para testar isoladamente as várias classes e métodos da componente Java MuSec, foi usada a *framework* de testes unitários JUnit. O JUnit, é uma *framework*, com suporte à criação de testes automatizados na linguagem de programação Java. Foram codificados e executados vários testes ás principais classes e métodos, importantes para o funcionamento do MuSec.

H.1.4 Classe CommandLine

Esta classe permite iniciar a componente Java MuSec, em modo linha de comandos. O objetivo desta classes de testes é testar os argumentos introduzidos pelo utilizador. O método testado foi o método Start, que recebe os argumentos introduzidos pelo utilizador (ver Listagem H.1).

Listagem H.1: Método de teste - Start

```
1 /**
2  * Test of start method, of class CommandLine.
3  */
4 @Test
5 public void testStart() {
6
7     //nenhum argumento
8     System.out.println("Nenhum argumento");
```

```
9       boolean error2 = true;
10      CommandLine commandLineMode2 = new CommandLine(new
            String[]{});
11      boolean result2 = commandLineMode2.start();
12      assertEquals(error2, result2);
13
14      //5 argumentos
15      System.out.println("Cinco argumentos");
16      boolean error3 = true;
17      CommandLine commandLineMode3 = new CommandLine(new
            String[]{"start", host, user, pass, "another_arg"});
18      boolean result3 = commandLineMode3.start();
19      assertEquals(error3, result3);
20
21      //1 argumento não existente
22      System.out.println("Um argumento nao existente");
23      boolean error4 = true;
24      CommandLine commandLineMode4 = new CommandLine(new
            String[]{"start"});
25      boolean result4 = commandLineMode4.start();
26      assertEquals(error4, result4);
27
28      //1 argumento existente
29      boolean error5 = false;
30      System.out.println("Um argumento existente - Help");
31      CommandLine commandLineMode5 = new CommandLine(new
            String[]{"-help"});
32      boolean result5 = commandLineMode5.start();
33      assertEquals(error5, result5);
34
35      //1 argumento existente
36      System.out.println("Um argumento existente - Version");
37      boolean error6 = false;
38      CommandLine commandLineMode6 = new CommandLine(new
            String[]{"-version"});
39      boolean result6 = commandLineMode6.start();
40      assertEquals(error6, result6);
41
42  }
43
44  /**
45   * Test of start method, of class CommandLine with correct login
```

```
     and args
46   * struct.
47   */
48  @Test(timeout = 5000)
49  public void testStart2() {
50      System.out.println("Start Command Line Mode");
51      boolean error = false;
52      CommandLine commandLineMode = new CommandLine(new
           String[]{"start", host, user, pass});
53      boolean result = commandLineMode.start();
54      assertEquals(error, result);
55  }
56
57  /**
58   * Test of start method, of class CommandLine with incorrect
       login.
59   */
60  @Test
61  public void testStart3() {
62      System.out.println("Start Command Line Mode");
63      boolean error = true;
64      CommandLine commandLineMode = new CommandLine(new
           String[]{"start", "host", "root1", pass"});
65      boolean result = commandLineMode.start();
66      System.out.println(result);
67      assertEquals(error, result);
68  }
69  }
```

Na Figura H.1, é possível ver os resultados dos testes, a possíveis argumentos introduzidos por um utilizador.

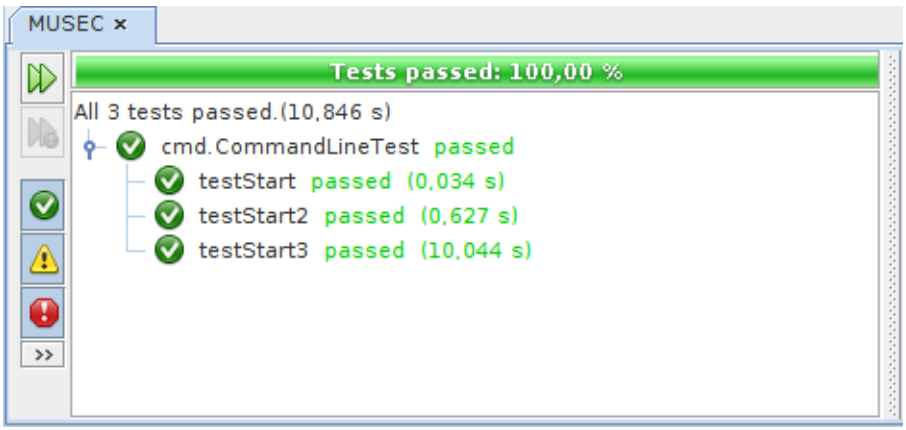

Figura H.1: Teste à classe CommandLine.

H.1.5 Classe DBTest

Nesta classe foi testado, o método GetConnection, que permite criar uma ligação SSL à base de dados MySQL do OSSIM. Na Listagem H.2 é possível ver o código de teste realizado ao método.

Listagem H.2: Método de teste - GetConnection

```
1  /**
2   * Test of getConnection method, of class DB.
3   */
4  @Test
5  public void testGetConnection() {
6      System.out.println("getConnection");
7
8      String host = "172.20.100.160";
9      String user = "root";
10     String pass = "password";
11
12     Connection expResult0 = null;
13     Connection result0 = DB.getConnection("", "", "");
14     assertEquals(expResult0, result0);
15
16     Connection expResult = null;
17     Connection result = DB.getConnection(host, user, "");
18     assertEquals(expResult, result);
19
20     Connection expResult1 = null;
21     Connection result1 = DB.getConnection(host, "", "");
22     assertEquals(expResult1, result1);
23
24     Connection expResult2 = null;
25     Connection result2 = DB.getConnection(host, "usernotexist",
           pass);
26     assertEquals(expResult2, result2);
27
28     Connection expResult3 = null;
29     Connection result3 = DB.getConnection(host, user, pass);
30     assertNotEquals(expResult3, result3);
31
32 }
```

Na Figura H.2, é possível ver os resultados a várias tentativas de criação de ligações, com parâmetros diferentes.

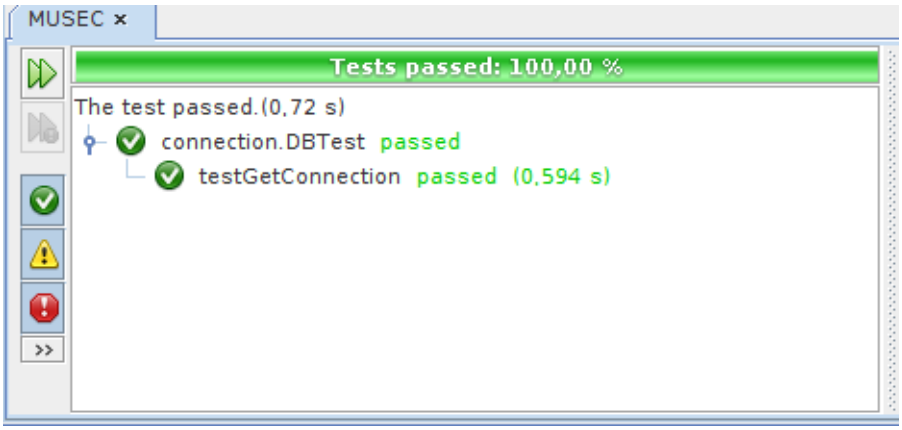

Figura H.2: Método GetConnection.

H.1.6 Classe EventDAO

Esta classe é responsável por se ligar à base de dados do OSSIM e obter eventos. O objetivo deste teste é verificar se a obtenção de eventos na base de dados do OSSIM acontece, sem problemas. Foi realizado um *dump* à base de dados do OSSIM, porque a base de dados OSSIM está em constante modificação e como tal era impossível trabalhar sobre ela, para a realização de testes unitários. O *dump*, foi assim utilizado nestes testes. Nesta classe foram testados vários métodos, entre os quais o método HaveNewEvents e o método GetNewEvent. O método HaveNewEvents permite verificar se existe novos eventos a partir de um *timestamp*. Na Listagem H.3 é possível ver o código do teste realizado ao método.

Listagem H.3: Método de teste - HaveNewEvent

```
1  /**
2   * Test of haveNewEvents method, of class EventDAO.
3   */
4  @Test
5  public void testHaveNewEvents() {
6      System.out.println("haveNewEvents");
7      EventDAO eventdao = new EventDAO(host, user, pass);
8      boolean result = eventdao.haveNewEvents("2015-11-09
           12:14:58");
9      assertEquals(false, result);
10
11     EventDAO eventdao2 = new EventDAO(host, user, pass);
12     boolean result2 = eventdao2.haveNewEvents("2015-11-09
           12:14:56");
13     assertEquals(true, result2);
14 }
```

Foi ainda testado o método GetNewEvent que permite obter um evento da base de dados do OSSIM a partir de uma determinado *timestamp*. Na Listagem H.4 é possível ver o código do teste realizado ao método.

Listagem H.4: Método de teste - GetNewEvent

```
1  /**
2   * Test of getNewEvent method, of class EventDAO.
3   */
4  @Test
5  public void testGetNewEvent() {
6      System.out.println("getNewEvent");
7      EventDAO eventdao = new EventDAO(host, user, pass);
8      Event evento = eventdao.getNewEvent("2015-11-09 12:14:58");
9      assertEquals(null, evento);
10
11     EventDAO eventdao2 = new EventDAO(host, user, pass);
12     Event evento2 = eventdao2.getNewEvent("2015-11-09 12:14:56");
13     ArrayList<Event> lista = new ArrayList<>();
14     lista.add(evento2);
15     assertNotEquals(null, evento2);
16     assertEquals(1, lista.size());
17 }
```

Na Figura H.3, é demonstrado o resultado dos testes, aos métodos da classe EventDAO.

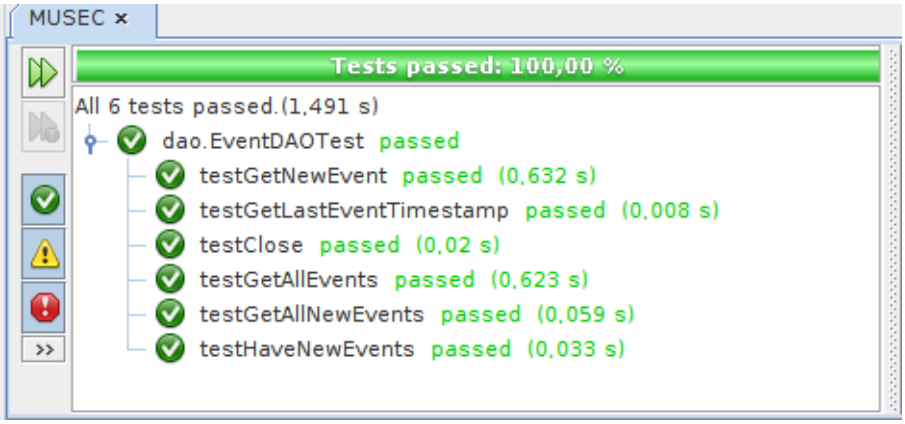

Figura H.3: Testes classe EventDAO.

H.1.7 Classe OSCSender

Esta classe é responsável pelo envio de mensagens OSC com os eventos para o componente Chuck. Nesta classe foi testado, o método SendMessage, que envia mensagens via OSC para o Chuck, neste caso com os eventos obtidos na base de dados do OSSIM. Na Listagem H.5 é possível ver o código do teste realizado ao método.

Listagem H.5: Método de teste - SendMessage

```
1  /**
2   * Test of sendMessage method, of class OSCSender.
3   *
```

```
4   * @throws java.net.UnknownHostException
5   */
6   @Test
7       public void testSendMessage() throws UnknownHostException {
8           System.out.println("Test sendMessage");
9           int port = 57111;
10          String address = "/chuck/events";
11          OSCSender osc = new
                OSCSender(InetAddress.getByName("localhost"), port,
                address);
12          Object[] values = {"event_id", 1, "event_timestamp"};
13          // message
14          boolean send = true;
15          boolean send_result = osc.sendMessage(values);
16          assertEquals(send, send_result);
17      }
```

Na Figura H.4, é possível ver que a mensagem foi enviada para o componente Chuck com sucesso, passando sem problemas no teste executado.

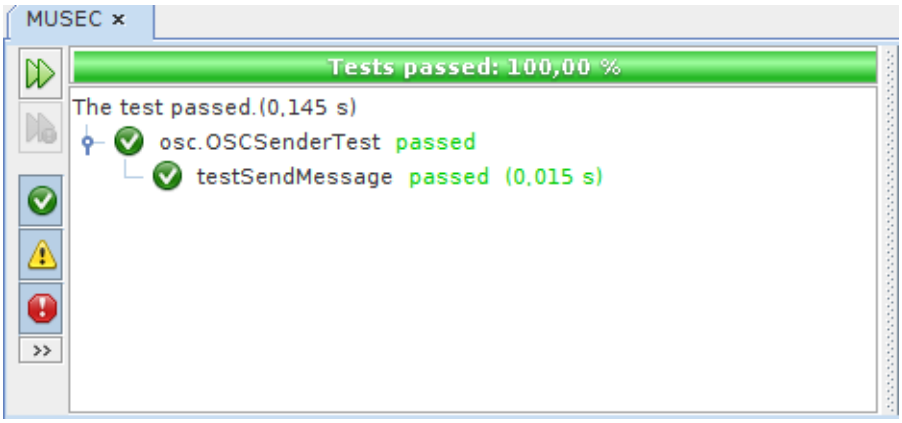

Figura H.4: Método OSC Sender.

H.1.8 Resultados Obtidos

Durante a execução dos testes foram encontrados alguns pequenos pormenores e erros que foram imediatamente reparados. Os erros mais comuns encontrados foram: a validação de certos *inputs* introduzidos pelo utilizador, mensagens de aviso geradas, e ainda alguns problemas com o envio de mensagens OSC para o componente Chuck, que por vezes não enviavam os eventos obtidos na base de dados do OSSIM. Com os testes unitários, foi possível corrigir esses erros.

H.2 Testes funcionais

H.2.1 Objetivo

Testar os requisitos funcionais e os casos de uso do MuSec. Verificar se todos os recursos oferecidos pela proposta de solução, estão aplicados e a funcionar corretamente.

H.2.2 Pré condições

Java 1.8 e Chuck 1.3.5.2 instalado. Executar o jar da componente Java MuSec, bem como o componente Chuck MuSec. Ligação na rede do servidor OSSIM.

H.2.3 Método utilizado

Verificar o fluxo dos dados de entrada para cada teste. Verificar ainda se a resposta da componente Java MuSec, era adequada às entradas introduzidas pelo utilizador. Foram testados dados válidos e inválidos, com a finalidade de abranger o numero máximo de casos possíveis. De seguida, são apresentados os vários casos de teste.

H.2.4 Acesso à base de dados

Este caso de teste, permite validar o acesso à base de dados MySQL, do OSSIM.

Tabela H.1: Caso de teste 1

[CT01] Acesso à BD				
Entradas			Resultados esperados	
Nº da ntrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	MuSec e acesso ao servidor OSSIM	IP OSSIM válido e utilizador e password válidos	Acesso ao servidor OSSIM	Mensagem de sucesso
2	MuSec e acesso ao servidor OSSIM	IP OSSIM inválido e ou utilizador e ou password inválidos	Acesso negado ao servidor OSSIM	Mensagem de insucesso

H.2.5 Obter Eventos

Este caso de teste, permite testar a obtenção de eventos, da base de dados do OSSIM.

Tabela H.2: Caso de teste 2

[CT02] Obter Eventos				
Entradas			Resultados esperados	
Nº da ntrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	MuSec e acesso ao servidor OSSIM	IP OSSIM válido e utilizador e password válidos	Acesso ao servidor OSSIM com os eventos apresentados no ecrã	Mensagem de sucesso e eventos apresentados no ecrã
2	MuSec e acesso ao servidor OSSIM	IP OSSIM inválido e ou utilizador e ou password inválidos	Acesso negado ao servidor OSSIM sem eventos apresentados no ecrã	Mensagem de insucesso e sem eventos apresentados no ecrã

H.2.6 Iniciar a sonorização

Este caso de teste, permite verificar se a sonorização é iniciada, perante os eventos que são enviados da componente Java MuSec, para o componente Chuck MuSec.

Tabela H.3: Caso de teste 3

[CT03] Iniciar sonorização				
Entradas			Resultados esperados	
Nº da ntrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	MuSec e acesso ao servidor OSSIM	IP OSSIM válido e utilizador e password válidos	Acesso ao servidor OSSIM e sonorização dos eventos	Mensagem de sucesso e os eventos são sonorizados
2	MuSec e acesso ao servidor OSSIM	IP OSSIM inválido e ou utilizador e ou password inválidos	Acesso negado ao servidor OSSIM e sem sonorização de eventos	Mensagem de insucesso e não existe sonorização

H.2.7 Parar sonorização

Este caso de teste, permite verificar se a sonorização é interrompida, quando um utilizador, assim o pretenda.

Tabela H.4: Caso de teste 4

[CT04] Parar sonorização				
Entradas			Resultados esperados	
Nº da ntrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	MuSec e acesso ao servidor OSSIM	Ordem de paragem efetuada pelo utilizador (via botão stop ou via Ctrl C)	Sem sonorização e sem novos eventos apresentados	Sem sonorização e sem eventos

H.2.8 Resultados Obtidos

Durante a execução dos testes foram encontrados alguns erros de validação de dados introduzidos pelo utilizador e mais alguns pormenores, que não comprometiam o MuSec. Todos os erros e ou pormenores foram corrigidos. Depois da correção, todos os requisitos propostos, funcionam perfeitamente.

H.3 Testes de execução

H.3.1 Objetivo

Verificar e testar se a componente Java MuSec, executa como planeado, em diferente hardware e sistemas operativos.

H.3.2 Pré condições

Sistema operativo Linux, Windows e Mac OS, com Java 1.8 instalado e chuck 1.3.5.2. Ligação na rede do servidor OSSIM.

H.3.3 Método utilizado

A componente Java MuSec, foi testada em sistema operativo Linux Mint 17.3 de 32 bits, Mac OS X El capitan de 64 bits, Windows 10 de 64 bits e ainda em sistema Rasbian Jessie de 32 bits.

H.3.4 Resultados Obtidos

Funcionou em todos os sistemas operativos testados. Apenas foram corrigidos, pequenos pormenores, para funcionar no sistema operativo Mac OS X (problema de caminhos). Depois de aplicadas as correções, a componente Java MuSec, ficou a funcionar perfeitamente em todos os sistemas.

H.4 Testes de Carga

H.4.1 Objetivo

Testar o MuSec, sob condições anormais de uso e testar a sonorização, em todos os níveis de riscos possíveis.

H.4.2 Pré condições

Sistema operativo Linux, Windows e Mac OS, com Java 1.8 instalado e Chuck 1.3.5.2. Ligação na rede do servidor OSSIM.

H.4.3 Método utilizado

Foram criadas regras (directives) no OSSIM, com o objetivo de detetar tentativas de ligações *SSH* a uma máquina da rede. Nessas regras, ficou definido que, perante o número de tentativas de ligações efetuadas a essa máquina, o risco iria aumentando de forma gradual. Depois para testar a sonorização de vários riscos, foi escrito um *script* em bash, que segundo a segundo, fazia automaticamente tentativas de ligações *SSH* à maquina definida anteriormente, nas regras do OSSIM. Esse *script*, foi executado durante 15 minutos, para se tirar as devidas ilações.

H.4.4 Resultados Obtidos

A aplicação foi testada e monitorizada durante 15 minutos. Durante esse tempo, foi possível ver a subida gradual do risco no OSSIM e também foi possível ouvir e ver que o MuSec, acompanhou essa subida, sonorizando cada nível de risco, conforme o esperado. Assinalou assim, as tentativas de *SSH*, durante todo o tempo do teste, sem problemas.

H.5 Testes de estabilidade

H.5.1 Objetivo

Testar o MuSec sob as condições normais e também testar se o sistema funciona de maneira satisfatória, durante um longo período de uso.

H.5.2 Pré condições

Sistema operativo Linux, Windows e Mac OS, com Java 1.8 instalado e Chuck 1.3.5.2. Ligação na rede do servidor OSSIM.

H.5.3 Método utilizado

Foram criados ficheiros de *logs*, que continham informações sobre o estado do MuSec e sobre os recursos ocupados pela aplicação no sistema operativo. Esses ficheiros de *logs*, eram enviados periodicamente (uma vez por dia), para um email. Era assim, possível tirar ilações sobre o comportamento do mesmo. Deste modo, era exequível monitorizar o estado do MuSec e que recursos ocupava no sistema operativo.

H.5.4 Resultados Obtidos

O MuSec foi testado e monitorizado 24 horas por dia, durante um mês, no centro informático da ESTG, instalado num *Raspberry Pi 2*, com o sistema operativo Raspbian Jessie de 32 bits. Funcionou perfeitamente sem falhas, sendo possível ouvir a sonorização do OSSIM, durante todo o tempo de teste, sem problemas.