Instituto Superior de
Engenharia de Coimbra

Departamento
de Engenharia Informática e de Sistemas

dgs.SGA-IDS

Dissertação apresentada para a obtenção do grau de Mestre em Informática e Sistemas

Autor
Ricardo Jorge da Fonseca Marques Ferreira

Orientadores
Professor Doutor Carlos Pereira
ISEC
Mestre Rui Gonçalo Amaro
Dognædis

Coimbra, Dezembro, 2013

Resumo

O presente documento representa todo o trabalho realizado por Ricardo Jorge da Fonseca Marques Ferreira, no âmbito da sua Dissertação de Mestrado em Informática e Sistemas, no Departamento de Engenharia Informática e Sistemas do Instituto Superior de Engenharia de Coimbra, do Instituto Politécnico de Coimbra. A Dissertação é composta pela especificação, projecto e implementação de uma aplicação da empresa Dognædis, denominada dgs.SGA-IDS. Esta trata-se de um módulo de uma *appliance* de segurança (SGA - Security Gateway Agent), capaz de detectar e gerir intrusões e/ou tráfego não autorizado, sobre a infraestrutura de rede que monitoriza e os segmentos que a compõem. Para tal utiliza um conjunto de IDS (Intrusion Detection System) já existentes, integrando-os de forma modular, para que possam ser facilmente substituídos no futuro.

É ainda trabalho do autor facilitar a integração desta aplicação com o *core* do SGA, outros módulos implementados sobre este e os IDS utilizados para monitorização da rede. A Dognædis, entidade acolhedora que suporta este projecto, é uma empresa que actua na área da segurança da informação e que ambiciona disponibilizar serviços inovadores com ênfase na excelência, na área em que se insere. Apresenta-se detalhadamente o estado da arte da temática abordada, a especificação de requisitos do sistema informático, a arquitectura do mesmo e a implementação do projecto.

Palavras chave: IDS, SIEM, IDMEF

Abstract

The present document represents all the work done by Ricardo Jorge da Fonseca Marques Ferreira, in the scope of his Masters Thesis on *Informática e Sistemas*, in the *Departamento de Engenharia Informática e Sistemas* of the *Instituto Superior de Engenharia de Coimbra*, of the *Instituto Politécnico de Coimbra*. The thesis is composed by the specification, planning and implementation of a project of the Dognædis company, named dgs.SGA-IDS. It consists of a module for a security appliance (SGA - Security Gateway Agent), capable of detecting and managing intrusion attempts and malicious traffic in the monitored network infra-structure. For that it uses a set of preexisting IDS, integrating them in a modular fashion so that they can be replaced in the future.

It's also the work of the author to facilitate the integration of the application into the core of SGA, other modules implemented using it and the IDS used for monitoring of the network. Dognædis is a company that provides services in the information security market, aiming to differentiate itself by the inovation and excellence of its services.

Each phase of the project is described in detail: the state of the art of the market, the software requirements specification for the proposed system, its architecture, and implementation.

Keywords: IDS, SIEM, IDMEF

Agradecimentos

Agradeço todo o apoio prestado pelo Professor Doutor Carlos Pereira e Mestres Rui Gonçalo Amaro e Sérgio Alves.
Agradeço também a todos os meus colegas de trabalho na Dognædis, o apoio e ambiente descontraído e produtivo que me proporcionaram.

Dedicatória

À minha família que sempre me apoiou em tudo. Em especial ao meu pai que não pode estar presente neste momento.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

SIMBOLOGIA E ABREVIATURAS

BSD **Berkeley Software Distribution**

CPU **Central Processing Unit**

CSIRT **Computer Security Incident Response Team**

CSS **Cascading Style Sheets**

CSS **Cascading Style Sheets**

DHCP **Dynamic Host Configuration Protocol**

DNS **Domain Name Service**

FIFO **First In First Out**

HTML **Hypertext Markup Language**

HTTP **Hypertext Transfer Protocol**

IDMEF **Intrusion Detection Message Exchange Format**

IP **Internet Protocol**

IPSEC **IP Security Protocol**

JSON **Javascript Object Notation**

NAT **Network Address Translation**

NIDS **Network Intrusion Detection System**

PDF **Portable Document Format**

QoS **Quality of Service**

RAM **Random Access Memory**

RFC **Request For Comments**

RPC **Remote Procedure Calls**

RSA **Algoritmo de criptografia de chaves publicas**

SGA **Security Gateway Agent**

SGBD **Sistema de Gestão de Base de Dados**

SIEM **Security Information and Event Management**

SSD **Solid State Drive**

TCP **Transmission Control Protocol**

TLS **Transport Layer Security**

UML **Unified Modelling Language**

VPN **Private Virtual Network**

XML **Extensible Markup Language**

Capítulo 1

INTRODUÇÃO

Este relatório foi redigido no âmbito de um estágio do Mestrado de Informática e Sistemas do Instituto Superior de Engenharia de Coimbra. O estágio foi realizado de Janeiro a Dezembro de 2013 nas instalações da empresa acolhedora. É objectivo deste documento reunir todo o trabalho realizado durante a dissertação.

1.1 Âmbito

A grande maioria das empresas depende de redes de comunicação informáticas para potenciar a agilidade da comunicação com clientes, fornecedores e do próprio processo de negócio, para ter a capacidade de resposta necessária ao sucesso nos mercados contemporâneos. Consequentemente, a segurança e estabilidade destas redes é essencial, pelo que a sua monitorização é um dos requisitos essenciais para o garantir. A aplicação de gestão de informação e eventos de segurança (SIEM - *Security Information and Event Management*)[1] desenvolvida durante este estágio, procura ajudar na monitorização da rede e dos activos que fazem parte desta, centralizando o armazenamento de eventos de segurança[1] detectados por várias ferramentas de monitorização já existentes.

O dgs.SGA-IDS enquanto módulo integrado no SGA (*Security Gateway Agent*) da Dognædis, um sistema desenvolvido pela empresa que ambiciona ser um ponto de passagem de tráfego de uma rede (*gateway*) e fornecedor de segurança à infraestrutura dos seus clientes, procura ser um ponto central de monitorização de intrusões e de tráfego não autorizado pelas politicas de utilização estabelecidas, auxiliando assim o administrador da rede na detecção de comportamentos estranhos ou inadequados na rede que administra.

Numa segunda dimensão de aplicação, existem hoje normas dedicadas a qualificar a segurança da informação das empresas, procurando garantir que esta é planeada, implementada, monitorizada e aperfeiçoada num processo contínuo e holístico. Estas certificações têm vindo a ser encaradas por organizações dos mais diversos sectores, como um valor cada vez mais fundamental para que obtenham um diferencial competitivo frente aos seus concorrentes. A norma que mais se destaca neste cenário de procura de excelência organizacional estandardizada a nível de segurança, é a ISO / IEC 27001[2], sobre a qual uma empresa se pode certificar, recorrendo para tal, fundamentalmente às normas ISO / IEC 27002[3] e ISO / IEC 27035[4], guias de implementação para obtenção da primeira.

Um dos requisitos desta norma é precisamente a monitorização contínua da rede de comunicação da empresa, requisito que o dgs.SGA-IDS preenche na sua totalidade.

[1] Entende-se por eventos de segurança qualquer acontecimento numa rede ou máquina, que possa ter implicações em termos da segurança da mesma.

1.2 O Projecto dgs.SGA-IDS

A aplicação desenvolvida pretende centralizar a monitorização de eventos de segurança de uma rede suportando para tal a recepção de eventos de várias ferramentas de análise já existentes. Estes eventos são armazenados pela aplicação e apresentados ao administrador da rede para que este possa tomar as acções que se revelem necessárias através de uma interface web. O presente projecto assenta na teoria que a monitorização eficaz e continua dos incidentes de segurança numa rede de comunicação, conduz a uma melhor e mais rápida reacção a estes, o que por sua vez será conducente a um maior nível de segurança e fiabilidade. Para tal, implementa um conjunto de funcionalidades enumeradas de seguida:

- Receber eventos de segurança provenientes de ferramentas de análise preexistentes e armazená-los num SGBD (Sistema de Gestão de Base de Dados) seguindo um modelo de dados criado pelo autor, que permite que a plataforma se torne independente face às ferramentas de detecção utilizadas;
- Associar estes eventos aos activos de rede a que dizem respeito e permitir a organização destes activos por grupos de negócio personalizáveis;
- Consultar os eventos recebidos filtrando-os por grupos de activos ou por outros critérios relevantes tais como o IDS que o reportou, nome do evento, endereços de origem e destino, severidade, entre outros;
- Criar relatórios sobre os eventos recebidos direccionados a equipas de resolução de incidentes do cliente;
- Criação de incidentes num sistema integrado de resolução de incidentes;
- Gerir os dados presentes no SGBD de modo a manter a eficiência da mesma. Estas operações consistem em eliminar eventos relativos a activos que já não estão presentes na rede ou eventos de determinada severidade, assegurando sempre uma cópia de segurança antes da eliminação dos mesmos.

Os requisitos a implementar de modo que as funcionalidades expostas sejam consideradas implementadas podem ser consultados no Capítulo 4.

O projecto dgs.SGA-IDS enquadra-se assim na atitude pro-activa que uma empresa deve ter em relação à segurança da rede e seus activos, permitindo que os eventos de segurança sejam detectados atempadamente e que se dê inicio ao processo de resolução dos mesmos.

1.3 Plataforma Existente

Nesta secção é descrito o trabalho já existente e no qual este projecto se baseia esclarecendo o que se trata de trabalho do autor e o que constitui trabalho de terceiros.

1.3.1 *Appliance* de Segurança (SGA)

Conforme referido a aplicação desenvolvida será integrada numa *appliance* de segurança que permite, até ao ano de 2013:

- Detecção e mapeamento de activos de rede;
- Detecção e gestão de vulnerabilidades;

- Serviços de túneis VPN (Virtual Private Network) e IPSEC (IP Security Protocol);
- A implementação de politicas de segurança (filtragem, condicionamento e priorização de tráfego);
- Serviços de rede que lhe permitem funcionar como *gateway* (DHCP, DNS, NAT, QoS, Encaminhamento de tráfego);

Cada uma destas funcionalidades actua de forma modular, sendo possível activar ou desactivar facilmente cada um destes módulos, ou acrescentar novas aplicações, independentemente do trabalho que tiver sido realizado no passado.

Relativamente à aplicação a desenvolver neste estágio, a empresa já oferecia serviços deste tipo aos clientes mas utilizando para o efeito aplicações já existentes e com fraca integração no SGA. Outro problema destas aplicações é a sua fraca prioridade no plano de negócio dos seus criadores e por consequência podem apresentar deficiências que persistem já que os seus autores não as actualizam.

Dado que as aplicações utilizadas para este efeito anteriormente eram externas à empresa e a modificação das mesmas era proibitiva devido às licenças utilizadas e à sua complexidade, a base para este estágio foi somente a plataforma que permite a integração com os outros módulos da *appliance* de segurança. Toda a implementação da aplicação, bem como todas as decisões arquitecturais couberam ao autor, sendo-lhe apenas requerido que respeitasse as seguintes restrições impostas pela *appliance*:

- Utilizar o mesmo sistema operativo sobre o qual se encontra implementada a *appliance*;
- Respeitar a arquitectura modular da SGA de modo que o módulo desenvolvido possa ser facilmente activado ou desactivado;
- Disponibilizar um conjunto de funcionalidades próprias da aplicação para que possam ser acedidas por outras aplicações e/ou *appliances* utilizando o modelo de comunicação do SGA, descrito na secção 5.1, do capítulo 5.

1.3.2 Interface *Web*

Como a aplicação a desenvolver será integrada na já referida *appliance* de segurança, existia já uma *template* base utilizada pelo SGA. Esta *template* define uma área exterior igual em todos os módulos e uma zona interior que pode ser definida pelo próprio módulo. Define também um padrão ao nível de CSS (Cascading Style Sheets) e Javascript de modo que todos os módulos tenham um aspecto similar ao nível de botões, menus, formulários, etc. Todo o trabalho de implementação do *layout* da zona interior dedicada ao módulo em si foi desenvolvido pelo autor. O aspecto da referida *template* pode ser consultado no apêndice B.

1.4 Entidade Acolhedora

A Dognædis é uma *Spin-off* do CSIRT[5] integrado no CERT-IPN[2] durante cinco anos. Após esses cinco anos de actividade, e com o aumento da procura dos seus serviços e produtos, constituiu-se como sociedade por quotas, em 2010. A Dognædis enquanto CERT-IPN operou no mercado Português de duas formas: disponibilizando serviços altamente especializados na área de segurança de informação e criando projectos de investigação aplicada, com uma forte componente de consciencialização social para diversas

[2]Equipa de Resposta a Incidentes de Segurança de Computadores do Instituto Pedro Nunes

problemáticas dessa área. O mais famoso e mediatizado destes projectos é o Vigilis (antigo Nonius), um sistema que produz, periodicamente, indicadores do nível de segurança na Internet portuguesa.

Empresa privada cuja visão é estar na vanguarda das tecnologias de segurança, a Dognædis define a sua missão actual como a vontade de levar, através de soluções inovadoras com ênfase na excelência, e a segurança da informação.

A empresa foi fundada por quatro engenheiros da Universidade de Coimbra: Francisco Rente, Hugo Trovão, Mário Zenha Rela e Sérgio Alves, mas teve também na sua génese, uma equipa de investigadores do Laboratório de Sistemas Confiáveis da UC, sendo que a simbiose de objectivos e conhecimentos destas duas equipas deu à Dognaedis um novo leque de capacidades.

Para além destes elementos, a Dognaedis conta ainda com uma equipa altamente especializada, bem como de diversos colaboradores na UC e na Indústria (nacional e internacional) para consultoria e colaboração, de acordo com as necessidades especificas de cada cliente.

Com três anos de existência, a Dognaedis tem actualmente, negócios em Portugal, França, Angola, Brasil, Índia e Reino Unido. Lançou um produto inovador, o CodeV, vencedor do prémio BPI Inovação no ano de 2012, um produto singular, não pela falta de oportunidade de negócio, mas sim pela complexidade que acarreta. A Dognaedis assume, acima de tudo, uma postura de luta face à actual conjuntura de crise, vendo o futuro da economia portuguesa além fronteiras.

1.5 Tecnologias e Ferramentas Utilizadas

As tecnologias utilizadas durante o estágio foram as seguintes:

- **Python[6]:** Linguagem de programação utilizada para a implementação.
- **Django[7]:** *Framework* web baseada em Python, utilizada para implementar a interface com o utilizador.
- **Twisted[8]:** Biblioteca baseada em Python, utilizada para implementar os servidores da solução.
- **LXML[9]:** Biblioteca Python para criação e manipulação de XML.
- **ReportLab[10]:** Biblioteca baseada em Python, utilizada para criar ficheiros PDF (*Portable Document Format*) programaticamente.
- **PostgreSQL[11]:** Este foi o SGBD utilizado para implementar o armazenamento de dado da aplicação.

As ferramentas utilizadas foram as seguintes:

- **Debian Linux[12]:** Sistema operativo em que foi desenvolvida a aplicação.
- **OpenBSD[13]:** Sistema operativo em que a instalação final da aplicação foi feita.
- **PyCharm Community Version[14]:** Ambiente de desenvolvimento utilizado pelo autor durante a implementação.
- **Subversion[15]:** Sistema de controlo de versões utilizado durante o estágio para gerir as versões do código fonte e dos demais documentos criados, em que se inclui a presente dissertação.
- **LyX[16]:** Editor de LaTeX utilizado para escrever o presente documento.

1.6 Estrutura do Documento

O documento encontra-se dividido em 8 capítulos e 7 apêndices. No Capítulo 1 introduz-se toda a envolvente do projecto e distingue-se entre as bases já existentes e o que foi efectivamente trabalho do autor. O Capítulo 2 descreve o planeamento inicial, o planeamento efectivamente cumprido e também a metodologia de desenvolvimento utilizada. O Capítulo 3 pretende apresentar os conceitos base necessários para compreender o que se pretende implementar, o *software* já existente que implementa ou usa estes conceitos e também as soluções concorrentes já existentes no mercado. Os capítulos 5 ao Capítulo 7 centram-se sobre o processo de desenvolvimento de software propriamente dito, nomeadamente e respectivamente, a análise de requisitos funcionais e não funcionais, os casos de uso, a arquitectura proposta, a implementação e os testes a que a solução implementada foi sujeita. Finalmente, o Capítulo 8 apresenta as conclusões que o autor tirou de todo o processo e os possíveis melhoramentos ou modificações que considera úteis de serem implementadas no futuro. No apêndice A apresentam-se as regras *ASVS* do *OWASP* que foram seguidas na implementação, seguidamente o apêndice B apresenta a *template* com o aspecto base, fornecida pelo SGA para uso pelo autor. O Apêndice C inclui alguns *screenshots* da aplicação. O apêndice D especifica detalhadamente os casos de uso implementados pela aplicação. O apêndice E apresenta alguns exemplos de código. Os apêndices F e G apresentam, respectivamente, o processo de investigação do protocolo Prelude e a documentação da informação daí resultante.

Capítulo 2

PLANEAMENTO

Neste Capítulo é apresentado o planeamento proposto e metodologia de desenvolvimento seguida pelo autor durante o estágio.

2.1 Planeamento Inicial

O autor iniciou o estágio na empresa acolhedora no dia 2 de Janeiro de 2013. Na Tabela 2.1 apresenta-se o planeamento inicial como consta na proposta de estágio.

Nr.	Tarefa	Inicio	Duração
1	Elaboração de um estudo das ferramentas de detecção de intrusão existentes, para que se seleccionem aquelas que permitirão alertar a rede de comunicação para ataques, de forma mais eficaz e eficiente.	2 de Janeiro	2 Semanas
2	Estudo do problema e escolha das tecnologias a utilizar para execução da dissertação.	17 de Janeiro	2 Semanas
3	Análise de Requisitos funcionais e não funcionais para o projecto a desenvolver.	1 de Fevereiro	1 mês
4	Definição e especificação da arquitectura para implementação dos requisitos propostos.	1 de Março	1 mês
5	Implementação da arquitectura especificada.	1 de Abril	2 meses e 2 semanas
6	Testes a funcionalidades implementadas e aos atributos de qualidade definidos.	16 de Junho	2 Semanas
7	Elaboração do documento final de dissertação.	1 de Julho	1 mês

Tabela 2.1: Planeamento inicial

2.2 Erro de Estimativa

Na estimativa inicial do tempo necessário para a implementação da solução, o autor assumiu que a fonte utilizada para a obtenção dos eventos estaria documentada e como tal a sua implementação não seria problemática. No entanto, depois de feito o estudo inicial e tomada a decisão sobre o protocolo a utilizar, constatou que o mesmo, não só não era documentado, como utilizava estruturas de dados binárias para codificar documentos XML.

Desta forma, para que a implementação fosse bem sucedida, seria necessário o estudo da estrutura interna do protocolo, de forma a que se criassem as condições para a transformação do binário recebido no formato final.

Dado que este passo de implementação não constava no plano inicial e a sua implementação requeria um esforço adicional considerável, face ao inicialmente previsto, o autor viu-se confrontado com a escolha entre diminuir funcionalidades a implementar, ou aumentar o tempo utilizado.

Assim, o autor procurou a implementação de todas as funcionalidades previstas inicialmente, através da requisição de um adiamento de entrega, proposto por sí e aceite pela Dognaedis e pelo ISEC.

A aceitação do adiamento traduziu-se numa melhoria efectiva do trabalho realizado, quer a nível de implementação, quer a nível documental.

De seguida apresenta-se a tabela 2.2 que contém o planeamento na forma como foi executado já com a fase de Implementação dividida entre os módulos implementados. Os módulos e as funcionalidades que implementam são apresentados detalhadamente nos capítulos 5 e 6.

Nr.	Tarefa	Inicio	Duração
1	Elaboração de um estudo das ferramentas de detecção de intrusão existentes, para que se seleccionem aquelas que permitirão alertar a rede de comunicação para ataques, de forma mais eficaz e eficiente.	2 de Janeiro	2 semanas
2	Estudo do problema e escolha das tecnologias a utilizar para execução da dissertação.	17 de Janeiro	2 semanas
3	Análise de Requisitos funcionais e não funcionais para o projecto a desenvolver.	1 de Fevereiro	1 mês
4	Definição e especificação da arquitectura para implementação dos requisitos propostos.	1 de Março	1 mês
5	Implementação da arquitectura especificada.	1 de Abril	4 meses e 2 semanas
	Alert Receiver	1 de Abril	1 mês
	Alert Manager	1 de Maio	1 mês
	Web Console	1 de Junho	2 meses e 2 semanas
6	Testes a funcionalidades implementadas e aos atributos de qualidade definidos.	1 de Agosto	2 semanas
7	Elaboração do documento final de dissertação.	16 de Agosto	4 meses

Tabela 2.2: Planeamento final

2.3 Metodologia de Desenvolvimento

Uma vez que a aplicação seria desenvolvida unicamente pelo autor e a entidade acolhedora não requeria entregas periódicas, a metodologia utilizada foi baseada no modelo em cascata[17] mas com algumas vantagens das metodologias ágeis já que a disponibilidade do cliente para esclarecimentos era total, uma vez que se tratava da própria Dognædis. Na tabela apresenta-se um mapeamento do planeamento já apresentado no ponto 2.2 às fases do modelo cascata. De notar que a fase de Manutenção foi omitida já que esta está fora do âmbito deste estágio.

Fase Cascata	**Fases Planeamento**	**Inicio**	**Duração**
Requisitos	1,2,3	2 de Janeiro	2 meses
Desenho	4	1 de Março	1 mês
Implementação	5	1 de Abril	4 meses e 2 semanas
Alert Receiver		1 de Abril	1 mês
Alert Manager		1 de Maio	1 mês
Web Console		1 de Junho	2 meses e 2 semanas
Verificação	6	1 de Agosto	2 semanas

Tabela 2.3: Metodologia de desenvolvimento

Na fase de implementação, foi dada prioridade aos requisitos funcionais mais importantes de acordo com a classificação apresentada no Capítulo 4.

Capítulo 3

ESTADO DA ARTE

Neste Capítulo será descrito o estado da arte em relação aos sensores e gestores de informação e eventos de segurança e as soluções já existentes com funcionalidades similares ao dgs.SGA-IDS. São também descritas as tecnologias e ferramentas que poderão fazer parte da implementação do trabalho.

Primeiramente, é descrito o que se entende por evento de segurança e as razões para a sua detecção ser importante. Seguidamente, são caracterizadas as ferramentas de detecção e os seus métodos de funcionamento. Finalmente, são apresentadas as soluções equivalentes já disponíveis.

3.1 Eventos de Segurança

Um evento de segurança descreve um acontecimento numa rede ou máquina, que pode ter implicações em termos de segurança.

Estes acontecimentos podem ser de vários tipos, entre os quais:

- Tentativas falhadas de autenticação numa máquina, o que pode indicar que um atacante está a tentar obter acesso a essa máquina sem autorização;
- Mudanças em ficheiros, o que pode indicar que algum ficheiro foi substituído por uma versão alterada que poderá conter código malicioso;
- Um protocolo que está a ser utilizado na rede da empresa que vai contra a politica de utilização;
- Uma tentativa de explorar um bug conhecido num software servidor através da rede.

Esta lista não é exaustiva. Qualquer acontecimento numa rede ou máquina pode ser detectado e um evento de segurança gerado para o reportar, bastando para isso existir sensores que o detectem ou que permitam a sua configuração para tal.

Actualmente, grande parte das empresas possui uma rede interna da qual depende para comunicar com os clientes e assegurar o seu negócio internamente. Qualquer falha de rede pode ter custos elevados em termos monetários e de imagem. Para garantir que esta se comporta de forma robusta e segura, é essencial aceder a toda a informação que transita nessa rede e activos que dela fazem parte, já que muitos dos eventos detectados podem indicar problemas futuros que podem assim ser evitados.

3.2 Sensores

Os sensores são também designados por sistemas de detecção de intrusões ou analisadores. Nos pontos seguintes o autor utiliza o nome que permita a classificação mais clara. É de ressalvar que para o sistema a implementar, todos são sensores. Um sensor tem como objectivo detectar eventos de segurança, informar o administrador e, possivelmente, activar medidas automáticas de mitigação dessa tentativa de intrusão (se tem esta capacidade torna-se também num sistema de prevenção de intrusões). Estes sistemas são usados por muitas empresas para detectar potenciais problemas de segurança e comportamentos de risco por parte de utilizadores e classificam-se em dois tipos que se detalham nos pontos seguintes.

3.2.1 Sistemas de Detecção de Intrusões de Rede (NIDS)

São programas que identificam eventos de segurança analisando o tráfego de rede. Apesar de poderem ser usados individualmente, em redes de maior dimensão são normalmente usados como sensores remotos, posicionados em locais em que podem analisar todo o tráfego de rede necessário.[18] Neste modo, os eventos detectados são reportados para um servidor central que serve de consola onde o administrador pode consultar toda a informação centralmente.

Estes sensores podem empregar dois métodos de detecção:

- **Detecção baseada em assinaturas**
  O sistema analisa o tráfego interceptado comparando-o com padrões de eventos previamente conhecidos que estão definidos em regras. Este método tem o problema de existir sempre um atraso entre a utilização de um método no mundo real e a criação do padrão para este poder ser detectado.

- **Detecção baseada em anomalias**
  Neste método são definidas características normais das redes analisadas (protocolos usados, largura de banda consumida, máquinas presentes, etc) e qualquer desvio desse padrão pode ser usado para inferir da existência de um evento de segurança.

O método de detecção por assinaturas é altamente dependente das regras utilizadas. Um conjunto incompleto de regras tem como consequência que nem todos os eventos conhecidos serão detectados. Existem dois conjuntos de regras para sistemas de detecção de intrusões de rede disponíveis de forma gratuita e que são actualizados pela comunidade de segurança:

- **SourceFire Vulnerability Research Team (VRT)** [19]
  É o conjunto de regras oficial do *Snort*, um NIDS que será descrito mais à frente nesta secção. Para obter este conjunto de regras o utilizador tem que se registar no site do *Snort* e obter um *oinkcode*, que se trata de um código com o qual se pode descarregar as regras.

- **Emerging Threats** [20]
  Este conjunto de regras surgiu como uma forma de complementar o conjunto anterior mas evoluiu para um conjunto de regras independente da plataforma de modo a ser suportado por qualquer NIDS.

Estes dois conjuntos podem, e normalmente são, usados simultaneamente para obter uma cobertura abrangente do maior número de ataques conhecidos.

Nos pontos seguintes, iremos introduzir e caracterizar em termos de funcionalidades, os sistemas de detecção de intrusões de rede *Open-Source* e gratuitos mais utilizados. Para além de uma descrição

textual será apresentada no final, uma tabela comparativa de funcionalidades que poderá ser uma base para comparação entre eles.

3.2.1.1 Snort

O *Snort* [21] foi criado por Martin Roesch em 1998 e é um sistema de detecção e prevenção de intrusões de rede. A licença de software utilizada é a GPL 2.0 [22]. Neste momento é mantido pela empresa Sourcefire da qual Martin Roesch é um dos fundadores. Pode ser usado em 3 modos: modo *sniffer*[1] como o *tcpdump*[23], modo *logger* de pacotes em que escreve os pacotes capturados para disco e modo de detecção de intrusões. É este último modo que tem interesse para esta dissertação.

No modo de detecção de intrusões utiliza detecção baseada em regras. Estas regras são, na sua maioria, criadas pela Sourcefire, empresa que mantém o software, contudo, estas podem também ser criadas pela comunidade, ou pelo próprio utilizador. Qualquer tráfego que seja detectado por uma regra constitui um evento de segurança. Em redes maiores pode ser configurado com limites de modo a que algumas regras não gerem demasiados eventos. Esta configuração ajuda o sistema a lidar com redes de maior dimensão.

3.2.1.2 Suricata

O *Suricata*[24] é um NIDS desenvolvido pela *Open Information Security Foundation* (OISF) que faz parte do *Navy's Space and Naval Warfare Systems Command (SPAWAR)* e do Departamento de Segurança Interna dos EUA. A primeira versão beta foi lançada em Dezembro de 2009 e a primeira versão estável em Julho de 2010. A licença usada é a GPL 2.0 [22]. Este sistema foi criado para implementar algumas novas ideias e tecnologias no campo de detecção de intrusões, sendo a principal vantagem um motor de detecção *multi-threaded*[2].

A sintaxe das regras utilizadas é compatível com o *Snort* de modo que este sistema pode usar regras de várias fontes, tais como as *Emerging Threats* e VRT.

O Suricata tem um funcionamento análogo ao de outros NIDS, monitorizando o tráfego de rede e emitindo eventos quando alguma das regras configuradas detecta tráfego malicioso. Analogamente, o Suricata suporta vários destinos para os eventos. Permite ainda a ligação com outro *software* através de bibliotecas que disponibiliza para o efeito.

O Suricata tenta ser mais eficiente e rápido na análise de tráfego através do seu motor de análise *multi-threaded.* O facto de ser *multi-threaded* permite-lhe utilizar mais eficientemente os novos processadores com múltiplos *cores*. Promete também no futuro usar o poder de processamento paralelo dos GPUs modernos, através da OpenCL[25] ou CUDA[26], para acelerar o processamento de pacotes.

3.2.1.3 Bro Network Security Monitor

O Bro[27] foi criado por Vern Paxson em 1998 e consiste num sistema de detecção de intrusão de rede open-source que corre sobre sistemas operativos UNIX. A licença que segue é a BSD[28]. O sistema monitoriza a rede à procura de tráfego malicioso. Detecta intrusões comparando o tráfego interceptado com padrões definidos como maliciosos. Estes padrões podem ser assinaturas de ataques ou comportamentos e características anómalas. Como os demais NIDS é normalmente colocado em pontos de intercepção de redes.

[1] *Programa que captura trafego de rede.*

[2] Técnica que usa múltiplos processos que partilham a memória entre si, de modo a acelerar a alternância entre os mesmos.

Tem a capacidade de fornecer uma análise detalhada de vários protocolos através de eventos que descrevem a actividade observada, o que é útil no processo de análise posterior a um ataque. Inclui um conjunto de *scripts*[3] desenhados para detectar os ataques mais comuns limitando o número de falsos positivos. Estes *scripts* são escritos numa linguagem própria do Bro e contém regras para identificar tráfego malicioso.

Ao analisar a actividade na rede, inicia acções definidas em *scripts* Bro para esse tipo de actividade. Por exemplo, verificar se um ficheiro deve ou não ser acessível por HTTP. Os *scripts* Bro podem usar a funcionalidade de reconhecimento de assinaturas que o sistema inclui. As assinaturas são expressões regulares e o contexto que o Bro mantém da actividade na rede permite-lhe diminuir a quantidade de falsos positivos. Para além das assinaturas, pode também analisar parâmetros do protocolo, tamanho do pacote, nome de ficheiros, etc.

Os *scripts* podem gerar ficheiros com a actividade na rede e alertas de ataques para sistemas de *logging*, incluindo o sistema de *logs* do sistema operativo. Podem também executar programas que podem por sua vez tomar medidas preventivas ou de alerta.

[3] *Ficheiros que contém comandos que devem ser executados em conjunto.*

3.2.1.4 Análise Comparativa

Na Tabela 3.1 apresentam-se as características dos vários NIDS descritos nos pontos anteriores.

Caracteristica	**Snort**	**Suricata**	**Bro**
Sistemas Operativos	Linux, Free e OpenBSD, Windows, Mac OS X	Linux, Free e OpenBSD, Windows, Mac OS X	Linux, Free e OpenBSD
Licença	GPLv2	GPLv2	BSD
Ultima versão	2.9.2	1.4.5	2.1
Regras disponíveis	VRT, Emerging Threats	VRT, Emerging Threats, Scripts Lua[29]	Bro (linguagem própria)
Motor de detecção multi-threaded	✗	✓	✗
Suporte IPv6	✓	✓	✓
Detecção por assinaturas	✓	✓	✓
Detecção por anomalia	✗	✗	✗
Prevenção de intrusões	✓	✓	✓
Envio para ficheiros Unified2	✓	✓	✗
Envio para Prelude-IDS	✓	✓	✗
Envio para SGBD	✓	✗	✗
Envio para servidor de *Logs*	✓	✓	✗
Envio para *logs* de sistema	✓	✓	✗

Tabela 3.1: Tabela comparativa NIDS

As funcionalidades em que são classificados são as seguintes:

Regras disponíveis Regras de que pode fazer uso para o seu motor de detecção.

Motor de detecção multi-threaded Se o motor de detecção suporta *multi-threading* numa tentativa de tomar partido de múltiplos processadores ou processadores de múltiplos *cores*.

Suporte IPv6 Se o NIDS suporta analisar tráfego IPv6.

Detecção por assinaturas Se suporta o modo de detecção por assinaturas.

Prevenção de intrusões Se suporta o modo de detecção por anomalia.

Envio para ficheiros Unified2 Se suporta enviar os eventos para ficheiros no formato *Unified2*.

Envio para Prelude-IDS Se suporta enviar os eventos para o *Prelude-IDS*.

Envio para SGBD Se suporta enviar os eventos directamente para SGBD.

Envio para servidor de Logs Se suporta enviar os eventos para um servidor de *Logs*.

Envio para logs de sistema Se suporta enviar os eventos para os *logs* do sistema.

Da análise da tabela pode-se concluir que o *Snort* e *Suricata* parecem ser as melhores opções, uma vez que estão equilibrados em termos de características. O *Bro* perde em comparação com estes últimos, devido à necessidade de escrever regras especificamente para ele e pelo menor número de métodos de comunicação dos eventos.

3.2.2 Sistemas de Detecção de Intrusões Locais (HIDS)

São sensores locais, que detectam eventos de segurança relativamente à máquina que se enontram a monitorizar. Apenas podem analisar informação a que a máquina tem acesso como os seus *logs*, tráfego que tem a máquina como destino, entre outros.[30]

Podem fazer uso de vários métodos:

- **Análise de *logs***
  Estes sistemas analisam os *logs[31]* do sistema para padrões que indiciam eventos que devem gerar alertas. Exemplo: Um utilizador autenticou-se como *root* fora das horas de trabalho.

- **Verificação de integridade de ficheiros**
  É criada uma base de dados com o estado dos ficheiros utilizando uma função de *hash*[4]*[32]* criptográfica. Periodicamente, são verificados os ficheiros e comparadas as *hashes* actuais com as guardadas anteriormente, detectando se foram alteradas e emitindo eventos em caso positivo.

- **Detecção de *rootkits***
  Verifica a existência de *rootkits*[5]*[33]* conhecidos no sistema.

Apesar de locais, estes sensores fornecem informação sobre eventos, aos quais um NIDS não teria acesso, visto que a informação é restrita à máquina sobre a qual o HIDS se encontra instalado.

Nos pontos seguintes serão descritos os HIDS *open-source*. Para além de uma descrição textual será apresentada no final uma tabela comparativa de funcionalidades que poderá ser uma base para comparação entre eles.

3.2.2.1 OSSEC

O OSSEC[34] é um HIDS *open-source* que utiliza os métodos de análise de *logs*, verificação de integridade de ficheiros, monitorização de politicas e detecção de *rootkits.*

Apesar de ser um HIDS, o OSSEC é organizado de modo distribuido. Uma máquina corre um *manager* ao qual todos os *agents* enviam informação do estado actual da máquina onde estão instalados. O protocolo usado é próprio do OSSEC. O *manager* usa a informação que recebe para proceder à verificação de ficheiros, monitorização de *logs*, etc. Para além de *agents*, o OSSEC também suporta monitorização através do uso de SSH para obter informação de máquinas em que não é possível ou desejável instalar aplicações adicionais.

[4] *String de bits gerado a partir de um conjunto de dados utilizando um algoritmo que garante que, aquando de alguma modificação desses mesmos dados, o string resultante será diferente.*

[5] *Ferramentas que permitem a um atacante controlar um computador sem ser detectado.*

3.2.2.2 Samhain

O Samhain[35] é um HIDS que fornece verificação de modificações de ficheiros, monitorização de *logs*, detecção de *rootkits* e monitorização de portos de rede. Tal como o OSSEC, apesar de o *agent* poder ser corrido individualmente, normalmente o sistema está organizado num modelo cliente-servidor em que as máquinas a monitorizar correm o *agent* que envia a informação para o *manager*. Adicionalmente, está disponível uma interface web para consultar os alertas e forçar a actualização das assinaturas de ficheiros remotamente.

3.2.2.3 Prelude-LML

O *Prelude-LML*[36] é um sistema de monitorização de *logs* incluido com o *Prelude-IDS*, que será apresentado na secção 3.4.5. Suporta unicamente a análise de *logs*. Possui um *plugin* de expressões regulares[37] que inclui regras para vários sistemas e servidores.

3.2.2.4 Sagan

O *Sagan*[38] é um sistema de monitorização de *logs* criado pela *Quadrant Security* que utiliza regras com uma sintaxe similar ao *Snort*. Através destas regras analisa os *logs* à procura de comportamento suspeito e, se encontrado, emite eventos. Adicionalmente, suporta o envio de eventos para uma base de dados *Snort* e nesse caso tenta correlacionar os eventos com os eventos do *Snort*. Esta funcionalidade de correlação é suportada unicamente neste modo.

3.2.2.5 AIDE

O *AIDE[39]* é um verificador de integridade de ficheiros. Cria uma base de dados com os *hashes* dos ficheiros que depois utiliza para verificar a integridade dos mesmos, emitindo eventos no caso de alterações. Os eventos são enviados unicamente para o sistema de *logs* da máquina.

3.2.2.6 Análise Comparativa

Na Tabela 3.2 apresentam-se as características dos vários HIDS descritos nos pontos anteriores.

Funcionalidade	**OSSEC**	**Samhain**	**Prelude-LML**	**Sagan**	**AIDE**
Sistemas operativos	Linux, Free e OpenBSD, Mac OS X, Solaris, Windows	Linux, Free e OpenBSD, Mac OS X, Solaris, Windows (só o *agent*)	Linux, Free e OpenBSD	Linux, Free e OpenBSD	Linux, Free e OpenBSD
Licença	GPLv3[40]	GPLv3	GPLv2	GPLv3	GPLv2
Ultima versão	2.7	3.0.13	1.0.1	0.3.0	0.15.1
Monitorização de *logs*	✓	✓	✓	✓	✗
Monitorização de integridade de ficheiros	✓	✓	✗	✗	✓
Verificação de presença de *rootkits*	✓	✗	✗	✗	✗
Prevenção de intrusões	✓	✓	✗	✗	✗
Envio para ficheiro Unified2	✗	✗	✗	✓	✗
Envio para Prelude-IDS	✓	✓	✓	✗	✗
Envio para SGBD	✓	✓	✗	✓	✗
Envio para servidor de *Logs*	✓	✓	✗	✓	✗
Envio para *logs* de sistema	✓	✓	✓	✓	✓

Tabela 3.2: Tabela comparativa HIDS

As funcionalidades em que são classificados são as seguintes:

Monitorização de logs Possui ou não a funcionalidade de monitorizar os *logs* da máquina onde está a correr por sinais de eventos de segurança.

Monitorização de integridade de ficheiros Possui ou não a capacidade de monitorizar ficheiros para detectar mudanças não reportadas.

Verificação de presença de *rootkits* Possui ou não a capacidade de procurar pela presença de *rootkits* na máquina.

Prevenção de intrusões Possui ou não a capacidade de tomar medidas preventivas ou reactivas a eventos de segurança.

Envio para ficheiro Unified2 Tem ou não a possibilidade de enviar os eventos que detecta para ficheiros no formato Unified2.

Envio para Prelude-IDS Tem ou não a capacidade de enviar os eventos que detecta para o Prelude-IDS através do protocolo deste.

Envio para SGBD Tem ou não a capacidade de enviar os eventos detectados para uma SGBD directamente.

Envio para servidor de *logs* Possui ou não a capacidade de enviar eventos para um servidor de *logs* remoto.

Envio para *logs* do sistema Possui ou não a capacidade de envio de eventos para o *log* do sistema.

A opção mais completa em termos de HIDS parece ser o *OSSEC* já que é o mais completo em termos de características. O *Samhain* é uma boa alternativa já que só perde a funcionalidade de detecção de *rootkits* em relação ao *OSSEC*.

3.3 Comunicação de Eventos

Os vários sensores estudados até aqui, quer NIDS, quer HIDS, suportam múltiplos tipos de comunicação para publicar os eventos que detectam. Nesta secção iremos descrever estes modos de comunicação.

3.3.1 *Logs* do Sistema Locais e Remotos

O sensor envia uma mensagem com a informação do evento para o *Log* do sistema. Os sistemas de logs podem ser configurados para enviar as mensagens para um servidor de *logs* central permitindo deste modo a centralização do armazenamento e gestão destas mensagens [41]. Alguns sensores suportam também a comunicação directa com o servidor de *logs* remoto.

3.3.2 Sistemas de Gestão de Base de Dados

Neste caso o sensor insere os eventos e toda a informação relativa aos mesmos numa base de dados criada num SGBD seguindo um modelo de dados próprio. A grande maioria dos sensores suporta vários SGBDs, sendo os mais suportados o *MySQL* e *PostgreSQL*. Este modo tem a desvantagem que cada sensor tem o seu modelo de dados diferente, o que faz com que uma ferramenta que queira receber eventos por esta via tenha que implementar cada um deles.

3.3.3 Prelude IDS

Este modo suporta o envio de eventos para o sistema *Prelude IDS* que será descrito em detalhe na secção 3.4.5. Este modo comunica transformando os eventos e seus detalhes num documento XML que segue o RFC 4765[42] enviando-os de seguida para o sistema Prelude IDS. O protocolo utilizado para este envio é um protocolo especifico do *Prelude IDS* utilizado para encapsular e codificar o XML. Este protocolo está documentado no apêndice G, documentação essa fruto de um esforço de investigação utilizando capturas de tráfego de rede e sua análise utilizando a ferramenta *Wireshark*[43], que se encontra também ele descrito no apêndice F. O RFC 4765 é uma tentativa de estandardizar um formato de comunicação que seria comum a todos os sensores e gestores de eventos [42], utilizando para isso um esquema XML de nome IDMEF. Esta estandardização teria a grande vantagem de permitir a recepção de informação de todos os sensores simplesmente implementando um módulo de recepção IDMEF em vez dos vários necessários actualmente.

3.3.4 Formato *Unified* e *Unified2*

Este método escreve os eventos para um ficheiro binário no formato *unified* ou *unified2* definido pelo *Snort*. A escrita para um ficheiro em formato binário permite acelerar o processo de armazenamento de eventos, diminuindo assim a hipótese de perda de tráfego. Dificulta também a adulteração dos eventos em disco já que o formato é binário. A comunicação posterior destes eventos para os sistemas de gestão de eventos pode ser feita pela ferramenta *Barnyard2*, descrita seguidamente, que lê o ficheiro binário e envia os eventos nele presentes para o sistema referido.

3.3.5 *Barnyard2*

Apesar de se tratar de um ferramenta e não um modo de comunicação, o seu propósito é fazer a ligação entre sensores e as ferramentas que pretendem ter acesso aos eventos detectados e por isso é apresentada neste ponto. Esta ferramenta implementada em C e sob a licença GPL 2.0[22] tem como objectivo aliviar os sensores do peso de escrever *logs* e outros possíveis tipos de *output* mais lentos permitindo assim que se concentrem em interceptar pacotes e detectar ataques que é a sua principal função. Para cumprir este objectivo, esta ferramenta lê o formato binário *unified2* e envia os dados para vários tipos de saída, transferindo dados entre elas.

Na Tabela 3.3 apresentam-se as opções de saída suportadas por esta ferramenta. Como se pode ver, esta ferramenta para além de libertar o sensor de operações potencialmente lentas (espera pela rede, espera por *locks* na base de dados), permite multiplexar os alertas e *logs* de pacotes para vários destinos. Finalmente, como se trata de uma ferramenta independente, permite substituir o sensor por outro bastando para isso que este tenha um módulo de saída *unified2*.

Nome	**Função**
sguil	Os alertas são enviados para o sistema Sguil.
database	Envia os alertas para uma base de dados no esquema snortdb. O motor de base de dados pode ser PostgreSQL, MySQL,MS SQL Server, Oracle ou qualquer motor suportado pelo unixODBC.
alert_bro	Envia alertas para o Bro IDS usando o protocolo de comunicação Bro. Estes alertas são recebidos pelo Bro como eventos.
alert_syslog	Envia alertas para o serviço de logs do sistema.
alert_csv	Escreve os alertas num ficheiro CSV no formato do Snort.
alert_fast	Os alertas são escritos num ficheiro no formato de alertas rápidos do Snort.
alert_full	Os alertas são gravados num ficheiro no formato de alertas completos do Snort.
alert_fwasm	Este modulo de *output* envia os alertas para uma máquina onde está a correr o SnortSam que irá bloquear os IPs originários dos ataques.
alert_prelude	Envia os alertas para o sistema Prelude IDS.
alert_echidna	Envia os alertas para o Echidna. O Echidna trata-se de um substituto para o Sguil.
log_ascii	Guarda os pacotes em formato ASCII.
log_null	Desliga a funcionalidade de armazenamento de pacotes.
log_tcpdump	Guarda os pacotes em formato binário compativel com a ferramenta *tcpdump*.
alert_syslog_full	Envia os alertas para o syslog local ou um servidor de logs remoto.
log_syslog_full	Envia os pacotes para um syslog local ou um servidor de logs remoto.

Tabela 3.3: Módulos de saída do Barnyard2

3.4 Gestores de Informação e Eventos de Segurança (SIEM)

Um gestor de informação e eventos de segurança agrega, correlaciona e disponibiliza informação sobre alertas emitidos por sensores (NIDS ou HIDS) que correm em vários pontos de uma rede. Estes sistemas centralizam a informação de alertas num só ponto de modo a facilitar a avaliação do nível de segurança da rede por parte do administrador. Estes sistemas normalmente pretendem aumentar a visibilidade dos alertas permitindo efectuar procuras e consultar estatísticas e/ou relatórios. Nos pontos seguintes serão apresentados os SIEM disponíveis no mercado actualmente, caracterizando-os pela presença ou não das seguintes funcionalidades:

Sistemas operativos Sistemas operativos em que pode ser instalado e corrido.

Licença Licença de software que governa o seu uso.

Última versão Última versão disponível à data.

Recepção de logs remotos Possui ou não a funcionalidade de receber *logs* remotos pelo protocolo *syslog*.

Recepção de IDMEF Possui ou não a capacidade de receber eventos em formato IDMEF utilizando o protocolo *Prelude IDS*.

Recepção SGBD Snort Possui ou não a capacidade de receber eventos através de um SGBD com o esquema do *Snort*.

Procura de eventos Possui ou não a capacidade de efectuar procuras nos eventos que recebe.

Correlação de eventos Possui ou não a capacidade de correlacionar eventos, isto é, detectar que vários eventos se referem ao mesmo acontecimento ou acontecimentos relacionados.

Gráficos de estatisticas Possui ou não a capacidade de apresentar gráficos que permitam obter uma visão global sobre o que se passa na rede.

Relatórios Possui ou não a capacidade de criar relatórios que possam ser impressos e entregues a terceiros.

Note que uma tabela comparativa relativa a cada uma destas, de SIEM para SIEM, pode ser encontrada no final desta secção na Tabela 3.4.

3.4.1 Enterprise Log Search And Archive (ELSA)

O *ELSA*[44] é uma plataforma de centralização de *logs* baseada no servidor de *logs* Syslog-NG[45], SGBD *MySQL* e *Sphinx*[46]. Recebe e armazena os *logs* na base de dados, permitindo desta forma procuras completamente assíncronas sobre os *logs* de modo análogo a procurar na web. Permite também definir procuras que são automaticamente corridas quando novas mensagens são recebidas. Permite gerir o tamanho da base de dados eliminando mensagens mais antigas que um valor de tempo configurável. Inclui também ferramentas para gerir permissões de acesso às mensagens, configurar o envio de alertas por email, gerir procuras agendadas e geração de gráficos.

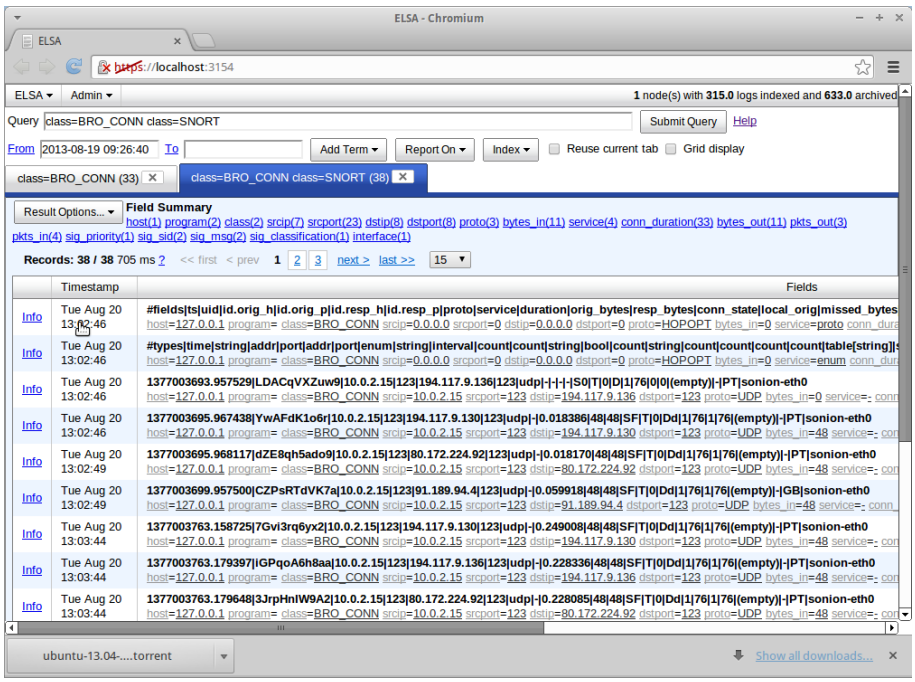

Figura 3.1: *Screenshot* do ELSA

3.4.2 Sguil

O *Sguil*[47] é um software de Gestão de Segurança de Redes escrito usando a linguagem TCL/Tk[48] e como consequência disso corre em qualquer sistema operativo que suporte a linguagem (e.g. Linux, Free e OpenBSD, Solaris, MacOS e Windows). O componente principal do sistema é uma interface gráfica que permite a consulta de eventos/alertas em tempo real, dados de sessão e capturas de pacotes de rede.

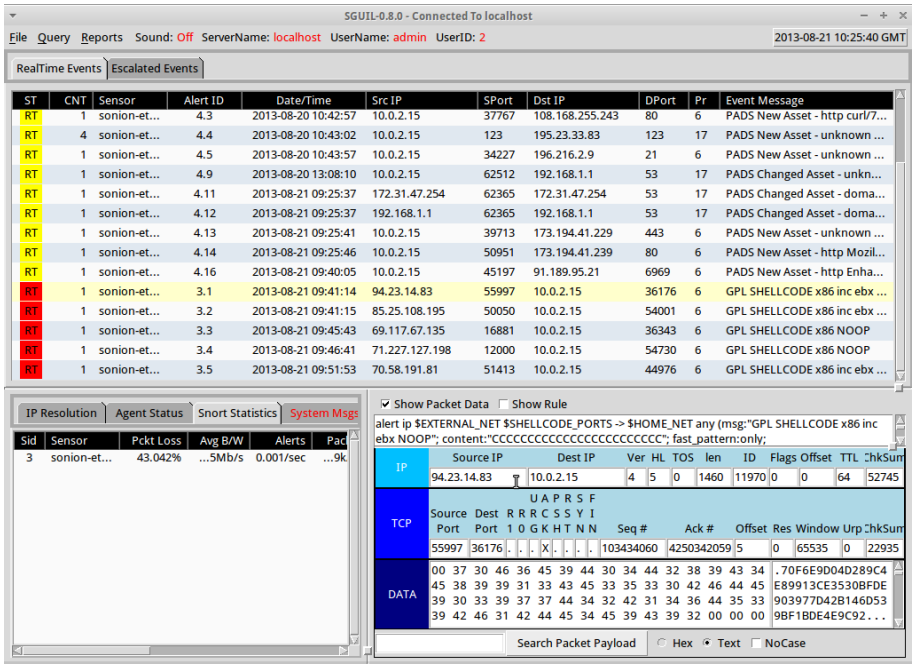

Figura 3.2: *Screenshot* do Sguil

3.4.3 Squert

O *Squert*[49] é uma aplicação web usada para explorar e consultar os dados de eventos/alertas numa base de dados do *Sguil*. É uma ferramenta visual que tenta fornecer contexto sobre os eventos através de meta-dados, séries de dados estatísticos e resultados agrupados de forma lógica e ponderada. Desta forma espera que esta representação dos dados dê origem a perguntas que não surgiriam de outra forma.

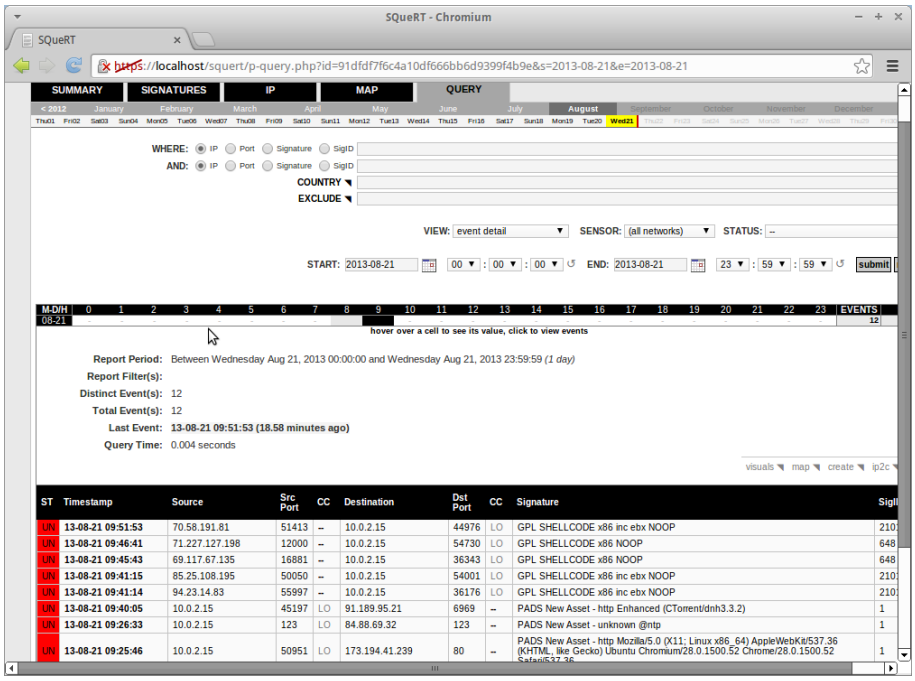

Figura 3.3: Screenshot do Squert

3.4.4 ACARM-ng

O *ACARM-ng*[50] é uma aplicação de correlação de alertas que pode facilitar significativamente a análise de tráfego numa rede. Armazena e correlaciona alertas enviados por sensores de rede e locais. O processo de correlação tem o objectivo de diminuir o numero de mensagens que o administrador tem que consultar agrupando eventos referentes à mesma actividade maliciosa. Inclui uma interface web que pode ser usada para consultar os alertas recolhidos.

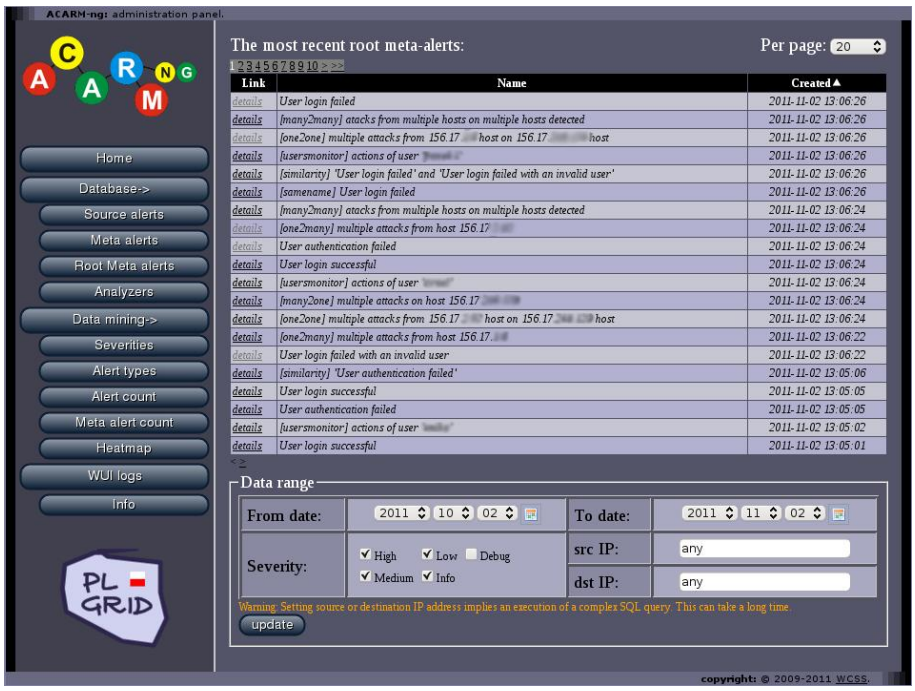

Figura 3.4: *Screenshot* do ACARM-ng

3.4.5 Prelude IDS

O *Prelude*[51] recolhe, normaliza, agrega e correlaciona eventos relacionados com segurança independentemente da marca ou licença do produto que deu origem ao evento. Para além de ter a capacidade de recolher qualquer tipo de *log*, beneficia de suporte nativo nalgumas aplicações o que lhe permite enriquecer ainda mais a informação. Os eventos de segurança são normalizados graças a uma norma internacional de nome *Intrusion Detection Message Exchange Format* (IDMEF) que foi criada por iniciativa da *Internet Enginneering Task Force* (IETF)[52] para que as várias ferramentas de segurança no mercado pudessem interagir. Tem também uma interface gráfica de nome *Prewikka*. O *Prelude IDS* possui uma versão *open-source* e outra comercial de nome *PreludePro*. A versão *open-source,* que é descrita aqui, permite consultar e efectuar procuras sobre os eventos recebidos, correlacionar os eventos e consultar gráficos com as estatísticas dos eventos recebidos. A versão comercial para além das funcionalidades da versão *open-source* inclui também um modelo de dados mais eficiente, sistema de tickets, criação de relatórios PDF e a possibilidade de suporte técnico. Recentemente, a Agosto de 2011, a *Prelude Technologies* entrou em processo de falência e foi comprada pela empresa francesa *C-S*[53]. A *C-S* tem mantido até aqui a versão *open-source* do *Prelude IDS*, contudo o seu suporte limita-se quase totalmente à disponibilização do software e da documentação criada anteriormente à aquisição. Todo o desenvolvimento está a ser direccionado para a versão comercial, resultando na estagnação da versão *open-source* em termos de funcionalidades e resolução de *bugs*. Outra desvantagem do Prelude-IDS é o seu modelo de dados pouco eficiente e complexo na versão *open-source* e que é utilizado como ferramenta para desencorajar o uso dessa versão pela C-S. Este modelo de dados torna tarefas simples de manutenção, como apagar falsos positivos, uma tarefa proibitivamente lenta.

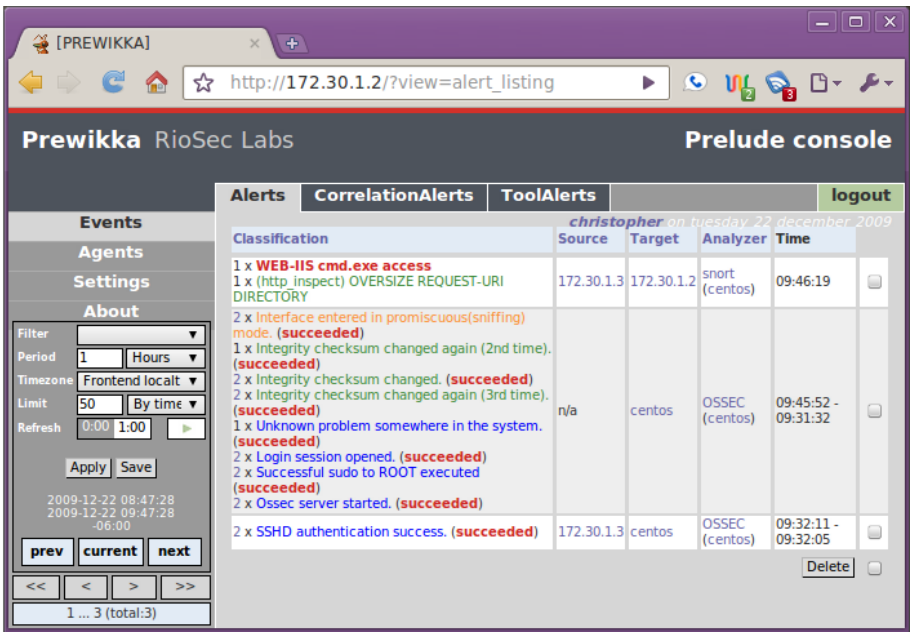

Figura 3.5: *Screenshot* do Prewikka

3.4.6 Snorby

O *Snorby*[54] é uma aplicação web programada em *Ruby*[55] que permite receber eventos de segurança gerados por vários sensores. A recepção de eventos é feita lendo directamente de um SGBD com o esquema do *Snort* com algumas tabelas adicionais especificas ao *Snorby*, o que limita desde já o leque de sensores de que pode receber eventos aos que tenham suporte directo para o *Snort* ou formato *Unified2*. O facto de ser implementado em *Ruby* implica também a dependência sobre um grande numero de *gems*[6] de *Ruby* o que se revela uma desvantagem em termos de instalação e manutenção. Uma funcionalidade muito útil é permitir ao utilizador classificar eventos por tipo, que posteriormente pode ser usado para filtragem e procuras.

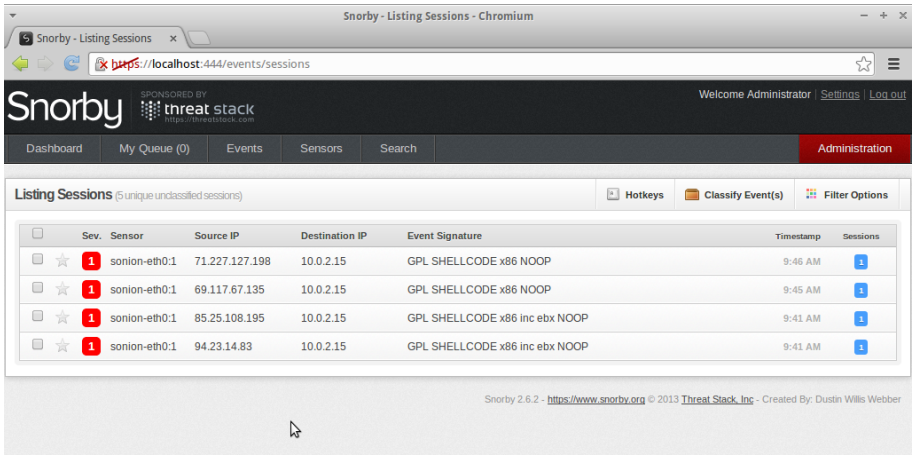

Figura 3.6: *Screenshot* do Snorby

[6] *Gems tratam-se de módulos Ruby que implementam funcionalidades não presentes na linguagem.*

3.4.7 Aanval

O *Aanval*[56] é um SIEM comercial desenvolvido pela *Tactical FLEX* desde 2003. Tem a capacidade de receber eventos do *Snort*, *Suricata* e mensagens do sistema de *logs*.

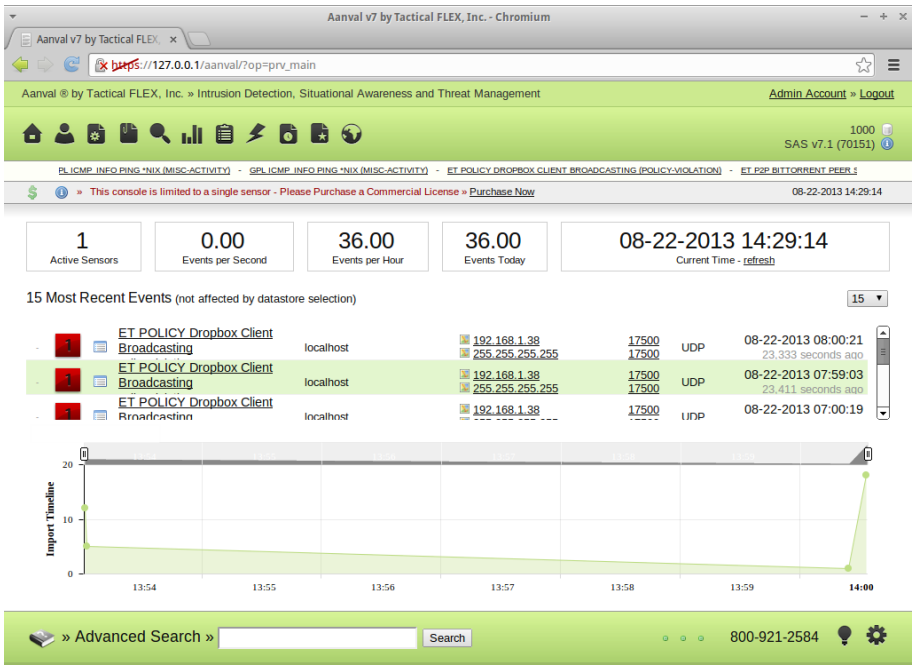

Figura 3.7: Screenshot do Aanval

3.4.8 Análise Comparativa

Na Tabela 3.4 apresentam-se as características dos SIEMs em análise, e acresenta-se uma comparação com a aplicação desenvolvida neste projecto, o SGA-IDS.

Funcionalidade	**ELSA**	**Sguil**	**Squert**	**ACARM-ng**	**Prelude-IDS**	**Snorby**	**Aanval**	**SGA-IDS**
Sistemas Operativos	Linux, FreeBSD	Linux, Free e OpenBSD, Solaris, MacOS e Windows	Linux, Free e OpenBSD, Solaris, MacOS e Windows	Linux	Linux, Free e OpenBSD	Linux, Free e OpenBSD	Linux, Free e OpenBSD, Mac OS X	Linux, Free e OpenBSD
Licença	GPLv2	GPLv2	GPLv3	GPLv2	GPLv2	GPLv3	Comercial	Comercial
Ultima versão	1.5	0.8.0	1.0	1.1.1	1.0.1	2.5.6	7.1	1.0
Recepção de *logs* remotos	✓	✗	✗	✗	✗	✗	✓	✗
Recepção IDMEF	✗	✗	✗	✓	✓	✗	✗	✓
Recepção SGBD Snort	✗	✓	✓	✗	✗	✓	✓	✗
Procura de eventos	✓	✓	✓	✓	✓	✓	✓	✓
Correlação de eventos	✗	✗	✗	✓	✓	✗	✓	✗
Gráficos de estatísticas	✓	✗	✓	✓	✓	✓	✓	✓
Relatórios	✓	✗	✗	✗	✗	✗	✓	✓

Tabela 3.4: Tabela comparativa SIEM

A principal conclusão a tirar é que não há uma opção claramente superior, mas o *ACARM-ng*, *Prelude-IDS* e *Aanval* revelam-se as mais capazes entre os SIEMs já existentes. O *Aanval* tem a desvantagem de ser uma solução comercial.

Em relação aos SIEMs já existentes, o SGA-IDS implementa a funcionalidade de recepção de IDMEF o que lhe permite receber eventos de um grande número de sensores directamente e da quase totalidade dos sensores existentes utilizando a ferramenta *Barnyard2*. Uma vantagem do SGA-IDS que não está reflectida na tabela é o seu modelo de dados simplificado que permite que a manutenção da base de dados seja efectuada em segundos e não em horas como se verificava com o *Prelude-IDS,* atingindo assim um dos objectivos mais importantes que a Dognædis ambicionava com este projecto. Implementa também a criação de incidentes no RTIR que é uma funcionalidade que só está presente na versão comercial do *Prelude-IDS*.

3.5 Implementação Típica

Um sistema para que se considere completo, deverá ter um ou vários sensores, a reportar eventos de segurança a um SIEM capaz de os gerir. Na figura 3.8 é apresentado um esquema de um sistema que incorpora sensores (locais e de rede) com um sistema de gestão de segurança de rede para formar um sistema completo de detecção e gestão de segurança de rede.

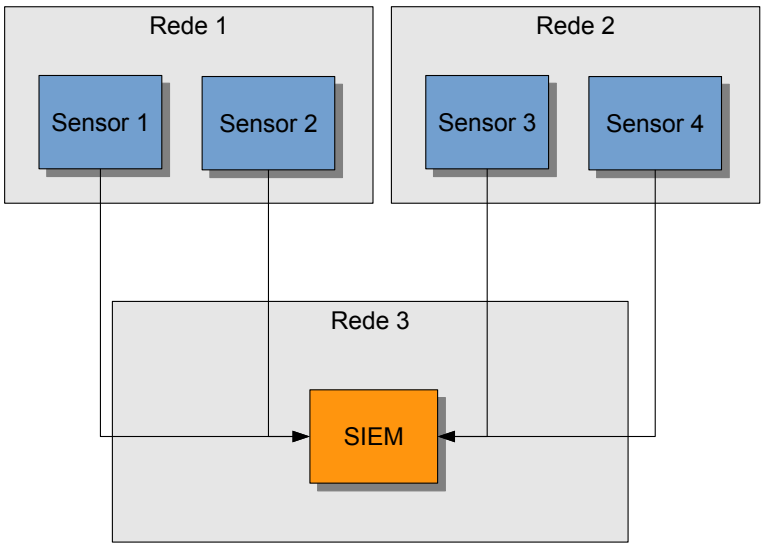

Figura 3.8: Implementação típica de um SIEM

De notar que as redes apresentadas servem simplesmente para reforçar o ponto que todos os componentes podem estar em redes distintas. Todas as outras combinações são possíveis.

Capítulo 4

ESPECIFICAÇÃO DE REQUISITOS

Neste Capítulo apresenta-se o problema proposto ao autor, anexando as decisões adoptadas por este, de forma a dar resposta aos objectivos do projecto. São então descritos os requisitos funcionais, casos de uso, requisitos não funcionais, dependências, restrições e riscos a que a implementação está sujeita.

4.1 O Problema

Nos tempos que correm qualquer rede que esteja ligada à Internet é alvo de ataques múltiplos. Estes ataques podem originar de simples amadores curiosos que tiveram acesso ao código de *exploit[57]* de uma vulnerabilidade (vulgarmente chamados de *script kiddies*) ou de organizações criminosas que se dedicam à venda de *exploits*, de exércitos de máquinas para correr esses *exploits* e até mesmo da informação obtida ilegalmente pelo uso dos *exploits*. Mesmo que não seja possível aceder a dados confidenciais através destes ataques, o simples facto de serem tentados pode causar quebras na qualidade de serviço das máquinas ou redes alvo.

Para detectar estes ataques existem sensores que emitem alertas com informação sobre o ataque detectado. No entanto, sem uma ferramenta que permita a consulta, procura e classificação desses alertas de uma forma intuitiva, eficaz e centralizada, torna-se difícil ao administrador de redes manter-se ao corrente da situação.

A Dognædis pretende oferecer uma solução integrada na applicance SGA que permita receber, armazenar e gerir eventos de segurança, de modo a permitir aos seus clientes obter uma melhor visão do que acontece nas suas redes e desta forma permitir-lhes tomar medidas para melhorar a segurança das mesmas.

4.2 Requisitos Funcionais

Neste ponto enumeram-se e descrevem-se os requisitos funcionais que a solução terá que implementar para que possua as funcionalidades que se pretende. Os requisitos funcionais encontram-se organizados por funcionalidade, funcionalidade essas que foram classificadas de acordo com a sua importância, obtida em reunião com o orientador da empresa.

- **Prioridade 1:** É uma funcionalidade central sem a qual a aplicação perde todo o valor para o cliente e para a entidade acolhedora;
- **Prioridade 2:** Uma funcionalidade com esta prioridade não é fulcral para o funcionamento da aplicação, mas a sua não implementação retira valor significativo à solução;

- **Prioridade 3:** Funcionalidade que optimizaria o uso da aplicação e adicionaria valor extra. A arquitectura não deve limitar a possível implementação embora esta não seja essencial.

Na Tabela 4.1 estão apresentadas as funcionalidades a implementar e a sua prioridade.

Identificador	Funcionalidade	Prioridade
ID-FU-001	Recepção de eventos de sensores preexistentes	1
ID-FU-002	Painel com situação actual/histórica (*Dashboard*)	1
ID-FU-003	Painel de eventos	1
ID-FU-004	Histórico de eventos	1
ID-FU-005	Geração de relatórios	2
ID-FU-006	Integração com módulo de resolução de incidentes	2
ID-FU-007	Manutenção da Base de dados	1
ID-FU-008	Obter a captura do tráfego que gerou um evento	3
ID-FU-009	Gerir grupos de activos	2

Tabela 4.1: Funcionalidades

4.2.1 ID-FU-001: Recepção de alertas de sensores

A aplicação deve ser capaz de receber e armazenar eventos emitidos pelos sensores que realizam a detecção propriamente dita. Esta deve permitir a recepção de mais que um sensor simultaneamente. Devem ser suportados os sensores mais utilizados evitando modificações desnecessárias sobre os mesmos. Os sensores mais utilizados são:

- Snort
- Suricata
- OSSEC

4.2.1.1 ID-RF-001: Receber alertas de sensores

A solução deve suportar receber eventos do maior número possível de sensores com o menor número de modificações sobre os mesmos.

4.2.1.2 ID-RF-002: Adicionar e remover sensores

A solução deve permitir adicionar e remover sensores da lista de sensores de que recebe eventos.

4.2.1.3 ID-RF-003: Permissões de sensores

Aos sensores que não foram adicionados, ou que foram removidos, não pode ser permitido enviar eventos para o sistema.

4.2.2 ID-FU-002: Painel com situação actual (*Dashboard*)

A aplicação deverá implementar um ecrã que o utilizador pode consultar para obter rapidamente o estado actual da rede por que é responsável. Este ecrã deve apresentar estatísticas num formato que permita a sua rápida percepção, das quais se destacam:

- 10 tipos de ataques mais detectados;

- 10 activos mais atacados;
- 10 endereços que mais realizam ataques;
- Outras estatísticas consideradas de interesse.

4.2.2.1 ID-RF-004: Estatísticas

A solução deve apresentar no mínimo as seguintes estatísticas sobre os alertas recebidos:

- Número de eventos nas últimas 24 horas;
- 10 tipos de eventos mais detectados;
- 10 máquinas mais atacadas;
- 10 máquinas que mais originam eventos.

4.2.3 ID-FU-003: Painel de alertas

A aplicação deverá possuir um ecrã que permita a consulta dos alertas à medida que estes são recebidos dos sensores. Este ecrã deve actualizar-se automaticamente. Deve também permitir que esta actualização automática seja desligada e ligada novamente sempre que o utilizador pretender. O ecrã deve suportar,no mínimo, a filtragem dos eventos pelos seguintes campos:

- Fonte do tráfego que originou o evento;
- Destino do tráfego que originou o evento;
- Grupo de activos.

4.2.3.1 ID-RF-005: Actualização automática

O sistema deve actualizar automaticamente a lista de eventos periodicamente (nunca maior de 10 segundos).

4.2.3.2 ID-RF-006: Alternar a actualização automática

O sistema deve permitir ao utilizador alternar entre a actualização automática e manual.

4.2.3.3 ID-RF-007: Filtragem por grupo de activos

Os alertas devem poder ser limitadas a um grupo de activos definidos previamente.

4.2.4 ID-FU-004: Histórico de alertas

A aplicação terá que manter um histórico de alertas recebidos. A aplicação deve permitir a procura nesse histórico com base em combinações de (no mínimo):

- Fonte do tráfego que originou o evento;
- Destino do tráfego que originou o evento;
- Grupo de activos;
- Intervalo de tempo.

4.2.4.1 ID-RF-008: Filtragem por tempo

O sistema deve permitir ao utilizador especificar um intervalo de tempo (granularidade de dia) ao qual os alertas mostrados serão limitados.

4.2.4.2 ID-RF-009: Filtragem por grupo de activos

Os alertas devem poder ser limitados a um grupo de activos (que serão definidos previamente).

4.2.5 ID-FU-005: Geração de relatórios

A geração de relatórios com a informação de um determinado intervalo temporal é uma funcionalidade muito útil na relação com os elementos de gestão do cliente, uma vez que facilita a comunicação de informação. A aplicação deve permitir gerar relatórios com estatísticas relevantes a um intervalo temporal seleccionado pelo utilizador, entre as quais:

- Tipos de ataques mais detectados;
- Activos mais atacados;
- Endereços que mais realizam ataques;
- Outras estatísticas consideradas de interesse.

4.2.5.1 ID-RF-010: Filtragem por tempo

O sistema deve permitir ao utilizador especificar um intervalo de tempo (granularidade de dia) ao qual os eventos e estatísticas incluídas no relatório serão limitados.

4.2.5.2 ID-RF-011: Filtragem por grupo de activos

Os relatórios devem poder ser limitados a um grupo de activos definidos previamente.

4.2.6 ID-FU-006: Integração com módulo de resolução de incidentes

A aplicação deve permitir ao utilizador criar um incidente para o(s) evento(s) seleccionados no sistema de resolução de incidentes RTIR[58] já utilizado pela entidade acolhedora. Esta integração permitirá ao utilizador dar inicio ao processo de mitigação do ataque.

4.2.6.1 ID-RF-012: Criação de incidente

O sistema deve permitir ao utilizador criar um incidente no sistema RTIR relativo a um evento.

4.2.7 ID-FU-007: Manutenção da Base de Dados

Uma aplicação deste tipo terá sempre uma base de dados com muitos registos. Para manter níveis de performance aceitáveis deverá ser possível definir um intervalo de tempo fora do qual se considera que os alertas perdem o valor e devem ser eliminados.

As constantes mudanças nos activos de uma rede também devem ser previstas. Para isso deve ser possível remover todos os alertas relativos a um activo. Adicionalmente deve ser possível remover alertas abaixo de determinado nível de severidade.

4.2.7.1 ID-RF-013: Eliminação de activos

O sistema deve permitir a eliminação de activos e, consequentemente, todos os eventos referentes ao activo.

4.2.7.2 ID-RF-014: Rotação da base de dados

O sistema deve permitir ao utilizador fazer uma cópia da base de dados e passar a utilizar uma base de dados limpa a partir desse momento.

4.2.7.3 ID-RF-015: Eliminação de eventos por severidade

O sistema deve permitir a eliminação de eventos com uma severidade igual, ou menor ou igual, a um valor especificado pelo utilizador.

4.2.8 ID-FU-008: Obter a captura do tráfego que gerou um alerta

Sempre que há um alerta que tem que ser investigado, ter acesso à captura completa de tráfego é muito útil para a investigação. A aplicação deverá fornecer um método para a obtenção da captura de tráfego que deu origem a um alerta recebido pela mesma.

Este requisito requer suporte por parte do sensor que emitiu o alerta e não foi julgado essencial para o valor da aplicação, por isso está classificado com prioridade 3. A arquitectura especificada deve, no entanto, procurar facilitar a sua implementação futura.

4.2.8.1 ID-RF-016: Obtenção da captura de tráfego

O sistema deve permitir ao utilizador requisitar o ficheiro com a captura do tráfego que deu origem a um alerta. O tráfego é identificado pelo intervalo de tempo, endereços de origem e destino e sensor onde foi capturado.

4.2.9 ID-FU-009: Gerir grupos de activos

Numa rede de maior dimensão torna-se impraticável o utilizador ver os alertas para toda a rede. A aplicação deve possuir a funcionalidade de gerir grupos de activos de modo a organizar as várias máquinas presentes na rede. Visto que a *appliance* tem um módulo de grupos de activos em desenvolvimento, a aplicação deve integrar-se com este se possível. Estes grupos serão posteriormente utilizados para filtrar informação.

4.2.9.1 ID-RF-017: Criar e apagar grupos de activos

O sistema deve permitir ao utilizador criar e apagar grupos de activos. Um grupo de activos possui um nome e os activos que engloba.

4.2.9.2 ID-RF-018: Editar grupo de activos

O sistema deve permitir ao utilizador editar o nome do grupo de activos.

4.2.9.3 ID-RF-019: Adicionar e remover activos de um grupo

O sistema deve permitir ao utilizador adicionar e remover activos de um grupo de activos.

4.3 Casos de Uso

Nesta secção são detalhados os casos de uso que descrevem a interacção do utilizador com a aplicação e implementam as funcionalidades descritas neste capítulo. Os casos de uso que envolvem a gestão de utilizadores, permissões de utilizadores e a sua autenticação não são apresentados, uma vez que essas funcionalidades são fornecidas pela SGA.

4.3.1 Diagrama de Casos de Uso

O diagrama apresentado na Figura 4.1 representa todos os casos de uso da aplicação e sua relação com o actor e outros casos de uso, se aplicável.

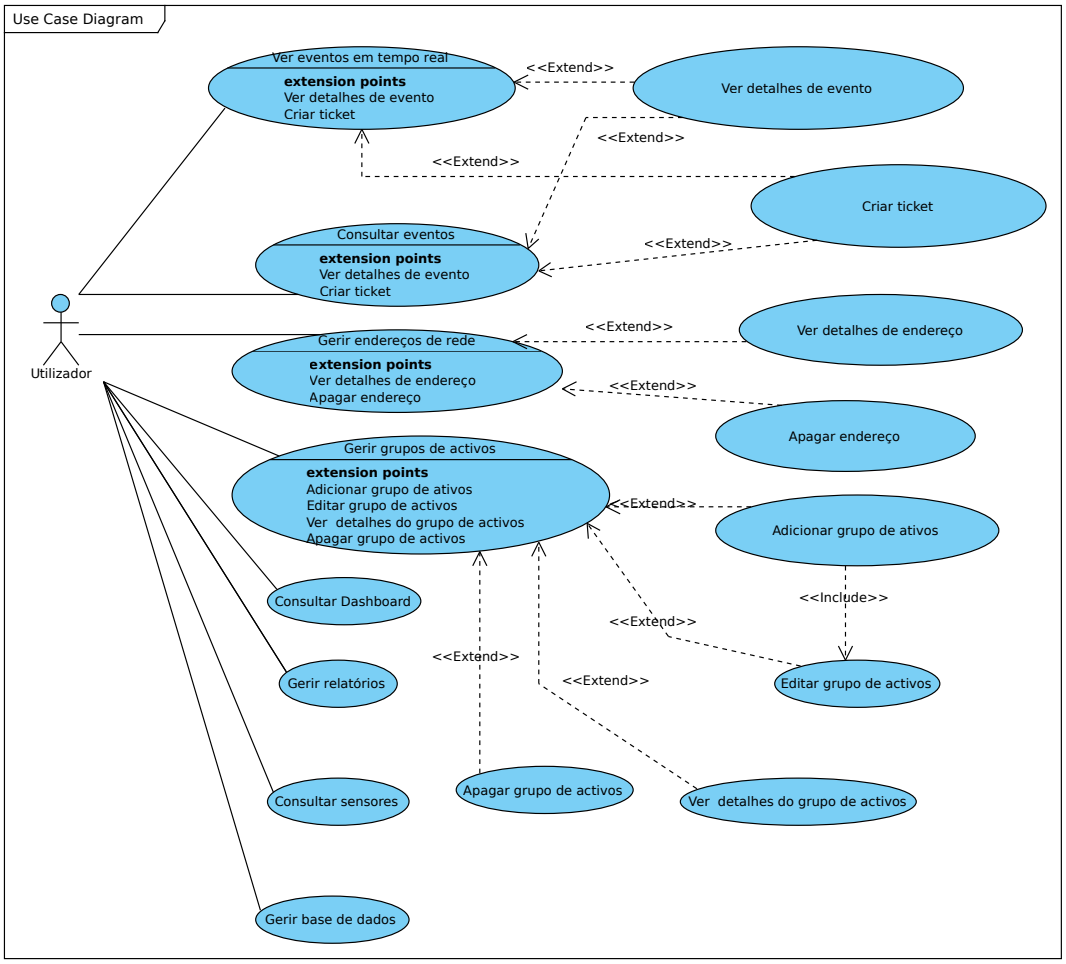

Figura 4.1: Diagrama de Casos de Uso

4.3.2 Casos de Uso

Na Tabela 4.2 apresenta-se uma pequena descrição de cada caso de uso que a aplicação deve implementar. A especificação detalhada de cada caso de uso pode ser consultada no apêndice D.

Tabela 4.2: Sumário dos casos de uso

Identificador	**Descrição**
ID-UC-001	O utilizador escolhe o menu "*Dashboard*" e visualiza as tabelas e gráficos de estatísticas: • 10 eventos mais detectados; • 10 activos mais atacados; • 10 activos que mais atacam. E os gráficos: • Distribuição de severidade dos eventos recebidos; • Distribuição de sensores que enviaram eventos; • Eventos recebidos nas ultimas 24 horas; • Eventos recebidos nos últimos 7 dias.
ID-UC-002	O utilizador escolhe o menu "*Realtime alerts*" e visualiza os últimos 10 eventos recebidos ordenados pela ordem de recepção. O utilizador pode também: • Mudar o número de eventos que vê (10, 25, 50 ou 100); • Filtrar os eventos por sensor, nome, origem, destino e grupo de activos; • Ligar e desligar a actualização automática da lista. • Ver os detalhes do evento; • Criar um *ticket* no RTIR com a informação do evento.
ID-UC-003	O utilizador pode ver todos os detalhes armazenados acerca do evento. Pode também ver o XML original do evento.
ID-UC-004	O utilizador pode criar um *ticket* no sistema RTIR com a informação do evento e do sensor que o enviou.

Identificador	Descrição
ID-UC-005	O utilizador escolhe o menu "*Browse alerts*" e pode consultar todos os eventos recebidos pelo sistema. O utilizador pode também: • Mudar o número de eventos que vê de uma vez (10, 25, 50 ou 100); • Filtrar os eventos por numero, sensor, nome, origem, destino, severidade, grupo de activos e todos os critérios simultaneamente; • Filtrar os eventos por um intervalo temporal com a granularidade de dia; • Ver os detalhes do evento; • Criar um *ticket* no RTIR com a informação do evento.
ID-UC-006	O utilizador pode consultar todos os endereços de rede do sistema. O utilizador pode também: • Mudar o número de endereços que vê de uma vez (10, 25, 50 ou 100); • Filtrar os eventos por numero, sensor, nome, origem, destino, severidade, grupo de activos e todos os critérios simultaneamente; • Ver os detalhes do endereço; • Apagar o endereço e os eventos com origem ou destino nele.
ID-UC-007	O utilizador consulta todos os detalhes sobre um endereço. Pode também apagar o endereço e todos os eventos com origem ou destino nele.
ID-UC-008	O utilizador pode apagar um endereço e todos os eventos com origem ou destino neste.
ID-UC-009	O utilizador pode consultar os grupos de activos definidos. Pode também criar, editar ou apagar um grupo de activos.
ID-UC-010	O utilizador pode criar um grupo de activos definindo o seu nome e posteriormente editando o mesmo para adicionar activos.

Identificador	Descrição
ID-UC-011	O utilizador pode consultar todos os detalhes do grupo de activos: • Nome; • Uma lista de todos os activos que lhe pertencem. O utilizador pode também editar o grupo de activos.
ID-UC-012	O utilizador pode apagar um grupo de activos.
ID-UC-013	O utilizador pode editar um grupo de activos. Pode mudar o nome e/ou adicionar e remover activos.
ID-UC-014	O utilizador pode gerar relatórios relativos a um intervalo de tempo e um grupo de activos. O utilizador pode descarregar relatórios gerados previamente.
ID-UC-015	O utilizador pode consultar os sensores que já comunicaram com o sistema. Pode também consultar todos os detalhes de um sensor.
ID-UC-016	O utilizador pode: • Eliminar todos os eventos com severidade igual à especificada; • Eliminar todos os eventos com uma severidade menor ou igual à especificada; • Criar uma cópia de segurança da base de dados e esvaziar a base de dados actual; • Descarregar uma cópia de segurança gerada anteriormente.

4.4 Requisitos Não Funcionais

Os requisitos não funcionais consistem em atributos de qualidade que serão especificados nesta secção. No Capítulo 7, Validação e Verificação, o autor procurará confirmar que a aplicação implementa cada um deles satisfatoriamente.

4.4.1 Modularidade

Como exposto no levantamento de requisitos, a aplicação depende de sensores que detectam os eventos e os reportam. A aplicação deve garantir que não está limitada a receber alertas apenas de um tipo de sensor, evitando assim a perda de valor caso este sensor passe a estar indisponível para uso por mudança de licença ou outra razão. A arquitectura da solução deverá dar ênfase ao isolamento dos vários módulos que a compõe, de modo a que uma alteração em um deles não dê origem a alterações em outros módulos. Deverá também procurar integrar-se de forma modular ao SGA, de forma a que alguma alteração sobre esta, não implique alterações sobre a base desenvolvida para a *appliance* de segurança.

4.4.2 Escalabilidade e Fiabilidade

Em engenharia de software entende-se por escalabilidade a capacidade de um programa para manipular uma quantidade crescente de trabalho de forma uniforme, estando preparado para crescer no serviço que pretende disponibilizar. No caso da solução implementada, este requisito implica que a mesma não imponha limitações na distribuição dos sensores e dos servidores receptores de eventos de modo a que o trabalho possa ser dividido por várias máquinas possibilitando assim o crescimento da capacidade do sistema. Caso ocorram falhas, o sistema deve dar resposta às mesmas evitando perder eventos.

4.4.3 Performance

A solução deve ser o mais rápida possível na recepção, processamento e armazenamento dos eventos. Observando um artigo sobre os desafios de manter um IDS numa empresa de grandes dimensões[59], onde foi estudada uma empresa na área da saúde com 8000 máquinas na sua rede, constatou-se que eram gerados aproximadamente 400 eventos por minuto. Com base nestes dados, e com o intuito de garantir que a aplicação pode ser escalada até essa dimensão, especifica-se que a solução deve ser capaz de receber e armazenar no mínimo 400 eventos por minuto, considerando um sensor, um *Alert Receiver* e um *Alert Manager* a correrem na mesma máquina.

4.4.4 Usabilidade

Usabilidade no contexto deste projecto entende-se como um atributo de qualidade de software que avalia se este é fácil de usar ou não. A solução implementada deve ser de fácil utilização e requerer treino mínimo para ser utilizada por um utilizador com noções básicas sobre o domínio.

4.4.5 Segurança

Sendo este projecto uma aplicação que pretende monitorizar redes quanto a ataques contra as mesmas, é importante que a solução implementada não se possa tornar ela própria um vector de ataque sobre as redes que monitoriza. O *software* a desenvolver deverá ser seguro, não permeável a ataques internos ou externos à aplicação e caso esses ataques aconteçam, o sistema deverá ter informação suficiente para os detectar correctamente.

4.5 Dependências

Nesta secção especificam-se os sistemas dos quais a solução depende para a sua implementação.

4.5.1 Sistema de Autenticação, Identificação e Permissões

Estes subsistema já se encontra implementado pela *appliance* SGA e consequentemente a solução a implementar limitar-se-á a integrar-se com esta.

4.5.2 Sistema de RPC

A SGA utiliza um sistema de RPC de modo a separar a interface web do código que implementa a lógica interna dos módulos e permitir a estes a utilização de serviços implementados por outros módulos. A aplicação irá utilizar esta solução como parte da sua integração na SGA.

4.6 Restrições

Nesta secção são apresentadas algumas escolhas impostas por necessidades de integração com a infraestrutura já existente e pelos objectivos da entidade acolhedora.

4.6.1 Licenciamento

As licenças de software normalmente têm o objectivo de limitar o que pode ser feito em termos de modificações e distribuição dessas modificações ao *software*, com o objectivo de impedir o uso contra os interesses do autor. Como é intenção da entidade acolhedora fornecer a solução implementada como produto no mercado, é necessário garantir que as licenças de todo o software e bibliotecas utilizadas o permitem sem necessidade de divulgação de informação proprietária da Dognædis.

4.6.2 Sistema Operativo

O sistema operativo onde a aplicação terá que correr é o *OpenBSD*[13] por este ser o sistema sobre o qual corre a SGA. O desenvolvimento poderá ser feito em outro sistema operativo, tendo o autor apenas que assegurar que a aplicação funcionará sem problemas em *OpenBSD*. Uma análise mais detalhada deste sistema operativo pode ser consultada no Capítulo 5.

4.6.3 Linguagem de Programação

A linguagem de programação utilizada será a *Python* versão 2.7. Uma análise mais detalhada sobre esta linguagem pode ser consultada no Capítulo 5.

4.6.4 Framework Web

A *framework web* que será utilizada para implementar a interface com o utilizador será a *Django* 1.5. uma vez que a interface do SGA foi implementada usando esta mesma *framework*, no entanto, uma análise mais detalhada sobre esta linguagem está disponível no Capítulo 5.

4.7 Riscos

Após a definição dos objectivos e requisitos do projecto, constatou-se que a implementação levantava alguns riscos já que dependia de outros projectos internos da entidade acolhedora, nomeadamente a SGA com a qual teria que ser integrado. Assim, foi realizada uma análise de riscos ao projecto com o objectivo de encontrar os problemas que poderiam surgir, o seu impacto no projecto e o que poderia ser feito para mitigar este impacto.

Neste ponto expoêm-se os riscos previsíveis para a implementação da solução proposta, em conjunto com possíveis estratégias de mitigação dos mesmos. Os riscos foram classificados quanto ao seu impacto numa escala de 1 a 3, que têm como significado:

- **Impacto 1**: Risco que não põe em causa o valor da solução e cuja mitigação é de fácil execução.
- **Impacto 2**: Risco que comporta alguma perda de valor para a solução, caso a estratégia de mitigação não seja seguida.
- **Impacto 3**: Risco que diminui significativamente o valor da solução caso não seja mitigado.

4.7.1 Risco ID-RI-001: *Core* do SGA em desenvolvimento concorrente

Descrição O *core* do SGA está em processo de desenvolvimento da segunda versão e pode não estar suficientemente maturo para a integração do módulo.

Implicações Dificuldade na implementação do módulo já que a infraestrutura de suporte fornecida pelo *core* da SGA não será estável. Dificuldade na integração do módulo com a SGA.

Impacto 2

Mitigação Deverá ser utilizada uma versão estável e mínima do core que permita o desenvolvimento e funcionamento do módulo separadamente. Deste modo assegura-se que o módulo é implementado seguindo as convenções e práticas do SGA e que a sua posterior integração aquando da estabilização do *core* é facilitada.

4.7.2 Risco ID-RI-002: Módulo de activos da SGA em desenvolvimento concorrente

Descrição O projecto SGA tem em desenvolvimento um módulo de gestão de grupos de activos com que o módulo a desenvolver pelo autor deveria interagir. Este módulo pode não estar nas condições mínimas necessárias para a integração ser bem sucedida.

Implicações Impossibilidade de implementação do requisito funcional referente aos grupos de activos.

Impacto 2

Mitigação O autor deverá desenvolver um sistema de grupos de activos interno ao módulo. Desta forma mantêm-se todas as funcionalidades do módulo com a desvantagem de uma pequena duplicação de esforço. A integração com o módulo externo poderá ser executada numa versão posterior.

Capítulo 5

ARQUITECTURA

Neste Capítulo, apresenta-se a arquitectura proposta para a solução a implementar, de modo a que esta cumpra com os requisitos apresentados no Capítulo 4.

Como resultado do estudo realizado pelo autor, foi decidido em reunião com o orientador da empresa que o sistema receberia eventos no formato IDMEF definido pelo RFC 4765. Deste modo favorece-se a modularidade e escalabilidade do projecto, ao assegurar que o maior número de sensores poderão comunicar com o sistema. Todo o *software* que suporta envio e recepção de IDMEF utiliza o protocolo Prelude, fazendo uso da biblioteca *libprelude* para o implementar. O protocolo *Prelude*, como já abordado na secção 3.4.5 do Capítulo 3, consiste num protocolo de rede criado pela *Prelude Technologies,* para envio de mensagens IDMEF dos sensores para o seu sistema *Prelude IDS*. Do anterior deriva que o presente projecto poderia também utilizar a biblioteca referida, no entanto, devido à venda da *Prelude Technologies* à empresa *C-S* e à notória falta de actualizações para a versão *open-source* da biblioteca, foi também decidido implementar uma solução própria de recepção de IDMEF, sem que recorresse à biblioteca referida. Desta forma a Dognædis e o projecto objecto deste estágio, tornam-se completamente independentes a qualquer restrição, ou condição imposta pela *libprelude*, ou pelos seus construtores.

5.1 Arquitectura do SGA

Uma vez que se pretende que o trabalho produzido se integre no *core* e módulos da *appliance* de segurança SGA, interessa especificar detalhadamente em que é que consiste este sistema e como a integração com o SGA-IDS será feita. Para tal, recorre-se às secções seguintes para esclarecer o leitor acerca desta componente estrutural do trabalho.

5.1.1 *Appliance* de segurança SGA

Trata-se de um sistema que ambiciona não só ser um ponto de passagem de tráfego (*Gateway*), mas também uma máquina que garanta serviços de segurança a redes e a sistemas de uma infraestrutura, juntando para isso vários serviços num só pacote, visando proteger as redes dessas infraestruturas, criando acessos seguros às mesmas e garantindo politicas de segurança internas.[60]

Cada *appliance* foi desenvolvida num sistema de três camadas: a camada de serviços e de aplicação, camada de gestão e controlo e finalmente a camada referente à interface com o utilizador.[60]

Services Layer

A primeira camada, denominada de *Services Layer* pelo programador, engloba todos os mecanismos operacionais do sistema. Uma vez que o sistema é uma *appliance* de segurança e de gestão de tráfego, é nesta camada que se irão encontrar os serviços de *Firewall*, DHCP, detecção de *malware*[1], mecanismos de autenticação entre outros. Devido à forma modular como o SGA foi concebido, facilmente se poderá adicionar novos serviços que serão manipulados através da camada de gestão e controlo.

Assim, no âmbito da integração do SGA-IDS, serão incluídos nesta camada de serviços os servidores utilizados pelo projecto para implementar a recepção de informação dos sensores (Snort e OSSEC, por exemplo).

Manager Abstraction Layer

A esta camada, o programador da *appliance* deu o nome de MAL (*Manager Abstraction Layer*), que consiste numa camada de abstracção que permite a interoperabilidade entre as camadas de interface e serviços. É através da utilização desta camada que os módulos comunicam uns com os outros, pois esta mantém uma lista de funções que podem ser usadas para interacção dos vários componentes do sistema.

De forma a garantir que a implementação fosse escalável e modular, escalável no sentido em que seja possível que as funcionalidades do SGA venham no futuro a ser acedidas por outras máquinas que com este tenham conectividade e modular, enquanto sistema que permita a implementação de cada módulo de forma independente dos restantes e facilmente integrados na solução geral, o programador da *appliance* optou pela abordagem de invocação remota de funções (*Remote Procedure Calls*).[2]

Apesar da utilização desta tecnologia ser um requisito para a integração com o SGA, o autor realizou uma análise da sua viabilidade para o SGA-IDS, comparando-a com a alternativa que seria a implementação de *web-services* que apresenta as seguintes características face ao RPC:

- Permite a interoperabilidade entre sistemas. Não requer que os vários módulos sejam desenvolvidos na mesma linguagem como acontece no RPC.
- Permite mais facilmente a localização de máquinas que disponibilizam *web-services*, quando comparado com os mecanismos necessários para o mesmo no RPC.
- Representa custos de processamento e transferência de dados muito superiores, quando comparado ao RPC.

Analisando o âmbito em que a comunicação será estabelecida para o SGA-IDS, nenhuma das vantagens apresentadas pelos *web-services* constitui uma real vantagem para esta implementação. A forma como os serviços ficam expostos ao público através de HTTP representa mesmo uma desvantagem, já que implica grandes preocupações com a segurança nesse caso.

Visto que não existe interesse em disponibilizar estes serviços na web para qualquer máquina, mas apenas a máquina definidas estaticamente, desenvolvidas no mesmo ambiente de programação, as vantagens da abordagem por *web-services* anulam-se e o RPC revela-se a solução mais apropriada para o problema.

A versão actual da SGA utiliza uma tecnologia de RPC desenvolvida internamente de nome Quyckro, que consiste numa solução implementada em Python e que oferece suporte para múltiplas invocações de diversos pontos.

[1] Software com fins maliciosos.

[2] É uma forma de execução remota de procedimentos em sistemas conectados a uma rede, independentemente dos protocolos de transporte que utilizam.

Assim, esta camada permite que de um único ponto de acesso seja possível aceder às referencias de objectos que disponibilizam as funções utilizadas por cada um dos módulos do SGA. Consequentemente, torna-se mais eficaz a distribuição de componentes pela rede e serve de base no crescimento futuro da implementação, que deverá para cada módulo, criar uma classe de interacção, garantindo a modularidade dos mesmos.

Interface Layer

A *Interface Layer* é aquela a partir da qual, o utilizador interage com todo o sistema. De modo a fornecer uma interface de fácil utilização, segura e que seja uma boa base para a expansão futura, o SGA utiliza a *framework Django* para este efeito. Uma descrição mais detalhada desta *framework* pode ser encontrada na secção 5.6.2 deste Capítulo.

Tirando partido da funcionalidade de *templates* desta *framework*, é fornecida uma *template* base que define uma área exterior igual em todos os módulos e uma zona interior que pode ser definida pelo próprio. Define também um padrão ao nível de CSS (Cascading Style Sheets) e Javascript de modo que todos os módulos tenham um aspecto similar ao nível de botões, menus, formulários, etc. Excluindo o referido anteriormente, é dada total liberdade ao autor para escolher as tecnologias a utilizar na implementação da interface do seu módulo, desde que as soluções seleccionadas não tenham implicações negativas no conjunto da *appliance*.

5.2 Integração do SGA-IDS no SGA

A Figura 5.1 ilustra a comunicação entre os componentes do sistema e a forma como o projecto SGA-IDS será integrado na *appliance*.

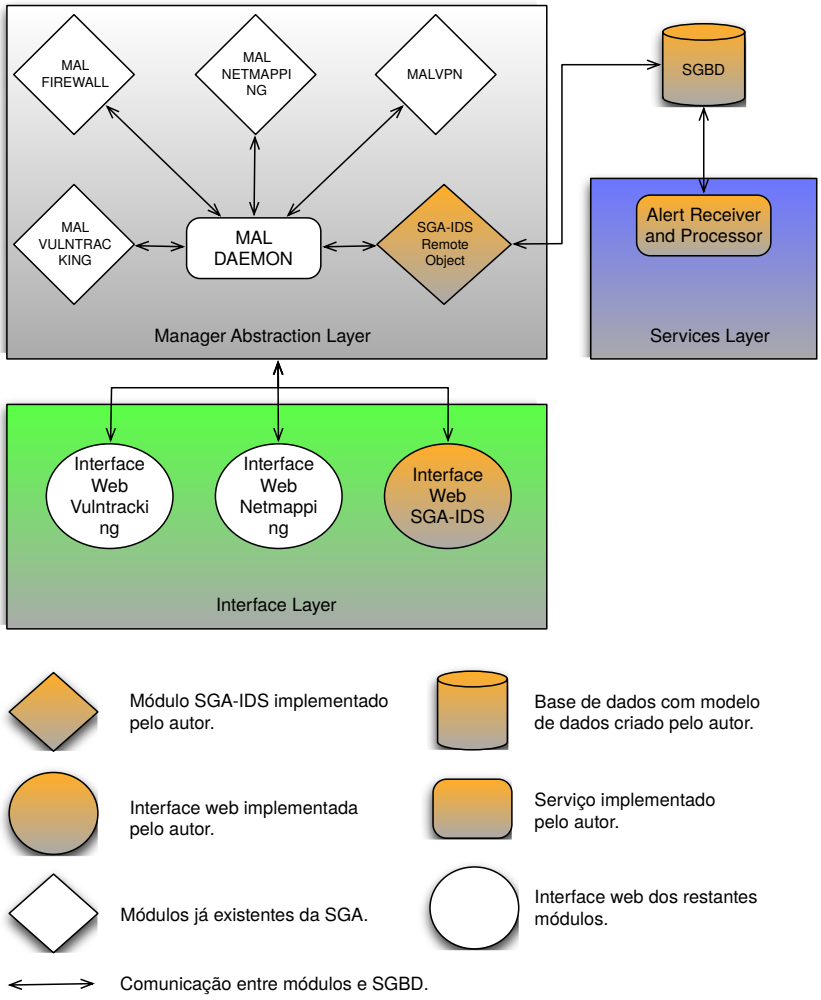

Figura 5.1: Integração do SGA-IDS com a SGA

Como se pode observar, na *Manager Abstraction Layer*, o módulo *MAL_Daemon* consiste num servidor que mantém as referências às bibliotecas especificas de cada módulo. É neste servidor que o módulo implementado pelo autor, o *SGA-IDS Remote Object* se irá registar de modo a permitir a chamada das suas funções através de RPC. A forma como este registo é efectuado e como a referencia é obtida posteriormente, pode ser consultada no apêndice E. Apesar de na figura só se encontrar esquematizado o SGBD utilizado pelo SGA-IDS por razões de simplificação do diagrama, na actual versão do SGA, cada módulo possui a sua própria base de dados, se necessária, e é responsável pelo acesso à mesma. Na *Services Layer,* está presente o módulo que recebe, processa e armazena no SGBD a informação proveniente dos sensores, e cuja execução será controlada pelo módulo *SGA-IDS Remote Object*. Este último módulo acede directamente ao SGBD, não utilizando para isso o módulo *SGA-IDS Remote Object*. Incluído na *Interface Layer*, está a aplicação web que implementa a interface com o utilizador. Esta última trata-se de uma aplicação web e, seguindo o modelo de comunicação do SGA, utiliza o módulo

MAL_Daemon para obter a referencia para as funções do *SGA-IDS Remote Object* que implementam a lógica de negócio propriamente dita.

5.3 Arquitectura Interna do SGA-IDS

Na Figura 5.2 ilustram-se os componentes principais da aplicação desenvolvida, procurando nas secções seguintes dedicadas a cada um deles, descrever a sua organização interna e como funcionam internamente, de modo a implementar as funcionalidades pretendidas para a aplicação, expostas no Capítulo 4.

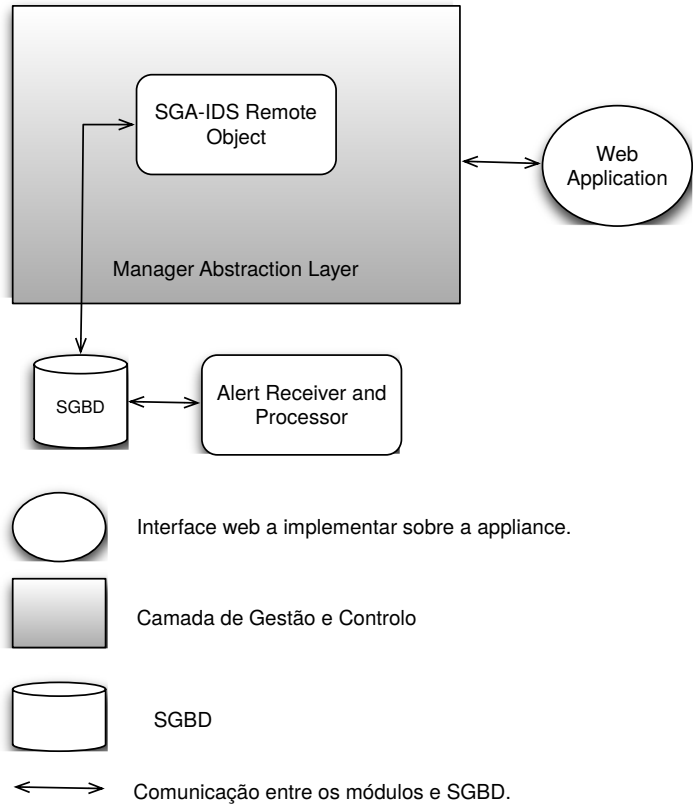

Figura 5.2: Arquitectura interna do SGA-IDS

5.3.1 SGA-IDS Remote Object

Este módulo tem como objectivo, através de *Remote Procedure Calls*, disponibilizar todas as funções aplicacionais implementadas, ao utilizador (através da Interface Layer) e aos outros módulos da plataforma, através dos seus objectos remotos (MAL-Vulntracking, MAL-netmapping, etc). Na Figura 5.3 apresenta-se a organização interna deste módulo.

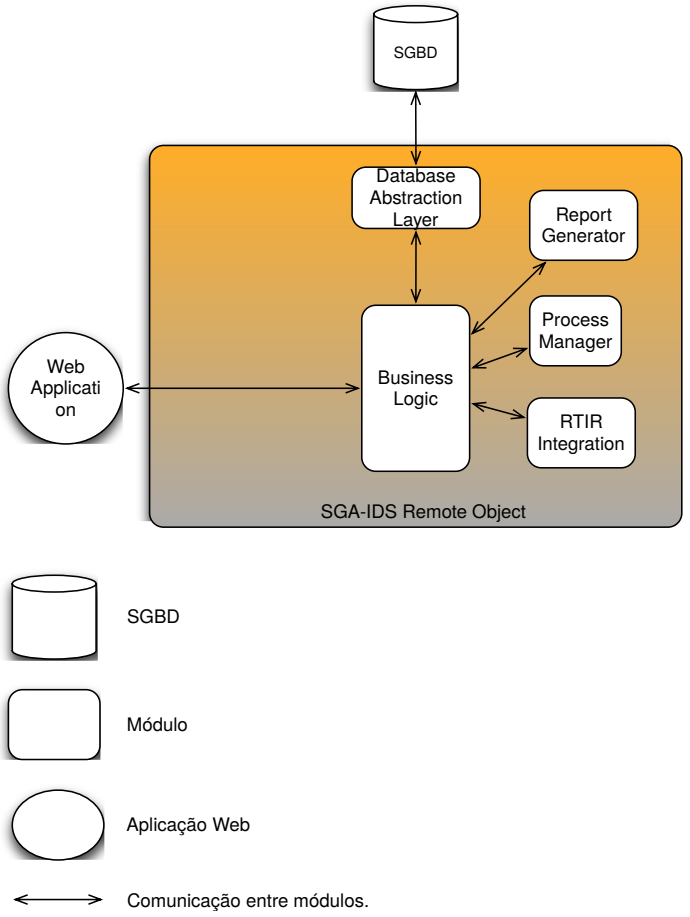

Figura 5.3: Arquitectura interna do módulo SGA-IDS Remote Object

Alerta-se o leitor de que a figura apresentada, pretende ser apenas uma representação conceptual, de facilitação de compreensão da arquitectura interna, podendo nem sempre existir de facto estas separações a nível de implementação. De seguida é descrita a função de cada um dos módulos que compõem o SGA-IDS Remote Object.

Business Logic

Este módulo é o núcleo da lógica de negócio e coordena as chamadas a todos os outros, de modo a implementar a funcionalidade requisitada.

RTIR Integration

(Implementa a funcionalidade ID-FU-006)

Dado que o projecto lida com eventos que podem sinalizar reais problemas de segurança, foi um requisito que este permitisse a criação de incidentes no RTIR, um software de gestão de incidentes

open-source. virado para as equipas de resposta a incidentes de segurança (CSIRT) e já em utilização na Dognaedis para esse fim.

Existem dois modos de integração com o RTIR. O primeiro é feito através do envio de emails para a aplicação RTIR que processa o email por determinados parâmetros e cria o incidente. A segunda é através do uso de uma API REST disponibilizada pelo RTIR, através da qual e utilizando o protocolo HTTP, o SGA-IDS pode criar incidentes automaticamente bem como aceder a informação sobre o mesmo.

Para facilitar este processo existe uma biblioteca em Python, a rtkit, que facilita a interacção com esta API. Este modo de implementação é muito vantajoso em relação à hipótese de envio de email, já que permite utilizar o padrão HTTP e evitar os problemas de segurança latentes do SMTP.

Utilizando este último modo de interacção com o RTIR, este módulo implementa a funcionalidade de criação de um incidente relativo a um evento de segurança, incluindo no texto do mesmo toda a informação disponível sobre o evento:

- ID do evento no SGBD;
- Nome do evento;
- Destino do tráfego incluindo o nome do grupo de activos a que pertence, se existir;
- Referencia[3] da vulnerabilidade que o ataque está a tentar explorar, se a informação for conhecida;
- Nome do sensor que reportou o evento;
- Data e Hora do sensor quando reportou o evento, de quando o evento foi detectado e de quando o evento foi criado;
- Severidade do evento;
- Se o ataque foi completado;
- Resposta tomada, se alguma;
- Nível de confiança do sensor na detecção efectuada;
- Fonte do tráfego incluindo o nome do grupo de activos a que pertence, se existir;
- Informação sobre o sensor que detectou o evento:
  - Analyzer ID do sensor;
  - Nome do sensor;
  - Nome do criador do sensor;
  - Modelo do sensor;
  - Versão;
  - Classe do sensor (de rede ou local);
  - Sistema operativo;
  - Versão do sistema operativo.

[3] Identificador que pode ser utilizado para aceder a informação detalhada sobre a vulnerabilidade em várias base de dados disponíveis na Internet.

Report Generator

(Implementa funcionalidade ID-FU-005)

Uma tarefa efectuada com muita frequência é a criação de relatórios sobre os eventos detectados na rede monitorizada. Estes relatórios podem ter como fim transmitir informação à gestão ou servir de evidencias, necessárias em muitas certificações da industria.

Como tal, foi um requisito que a aplicação gerasse relatórios sobre os eventos detectados num intervalo de tempo. O autor realizou um pequeno estudo sobre as bibliotecas de geração de PDF disponíveis, a fim de encontrar a melhor opção para esta implementação, chegando finalmente ao ReportLab[10]. Este tem as seguintes características que o posicionam como a melhor opção:

- Documentação completa;
- Suporte para inclusão de imagens no PDF gerado;
- Suporte para criação de tabelas de dados automaticamente a partir de matrizes;
- Suporte para corte automático das tabelas quando têm mais que uma página, e impressão do cabeçalho no inicio da nova página automaticamente;
- Suporte para criação de gráficos automaticamente a partir de matrizes;
- Suporta separar a apresentação do conteúdo, num sistema similar ao CSS, o que se revela útil em documentos de muitas páginas.

Com base no exposto, foi escolhida a biblioteca ReportLab para implementar este módulo. O módulo cria um documento PDF com a seguinte informação, para o intervalo de tempo escolhido:

- Activos que foram o destino de mais eventos;
- Activos que foram a origem de mais eventos;
- Distribuição da severidade dos eventos recebidos;
- Distribuição dos tipos de eventos recebidos;
- Lista dos eventos recebidos;
- Informação detalhada sobre cada evento individual:
  - ID do evento no SGBD;
  - Data e Hora do sensor quando reportou o evento, de quando o evento foi detectado e de quando o evento foi criado;
  - Sensor que o reportou;
  - Origem e grupo de activos a que pertence, se existir, do tráfego que despoletou o evento;
  - Destino e grupo de activos a que pertence, se existir, do tráfego que despoletou o evento;
  - Nome do evento.

Process Manager

Este módulo tem como finalidade permitir à aplicação controlar o serviço de recepção e processamento de eventos. Como tal, possui funções para correr e parar o referido serviço. O módulo limita-se a controlar o ciclo de vida dos processos que realmente implementam o serviço, não tendo qualquer comunicação com este.

Database Abstraction Layer

A utilização de código de acesso especifico a um tipo de SGBD é uma limitação à capacidade de adaptação da aplicação a uma nova arquitectura no futuro. A utilização de camadas de abstracção permite também implementar controlos de segurança em um só local, e não por toda a aplicação. Com o intuito de tomar partido destas vantagens, este módulo serve de abstracção para o SGBD em uso, e todo o acesso ao mesmo é feito através deste.

5.3.2 Web Application

(Implementa as funcionalidades ID-FU-002 a ID-FU-009, excluindo a ID-FU-008)

Este módulo implementa a interface com o utilizador através de uma aplicação web e integra-se na *Interface Layer* do SGA. A utilização de uma aplicação web para interface com o utilizador tem as seguintes vantagens para este projecto:

- Não é necessária a instalação de aplicações nas máquinas clientes que pretendam aceder ao serviço;
- A aplicação pode ser utilizada a partir de qualquer máquina que possua conectividade com a *appliance*, não impondo limites sobre a arquitectura em que se baseia a máquina;
- A utilização de aplicações web é muito familiar para o utilizador, o que aumenta a usabilidade;
- A manutenção pode ser feita centralmente e não em cada máquina cliente;

Existem também algumas desvantagens, para as quais a aplicação implementa estratégias de mitigação:

- Exige mais trabalho para garantir que a aplicação é utilizável nos diversos navegadores do mercado. A aplicação possui controlos que mudam a renderização, se necessário, para o navegador em uso;
- O código HTML, CSS e JavaScript fica disponível para o utilizador. A aplicação toma o cuidado de efectuar toda a validação de inputs, e processamento dos mesmos, do lado do servidor, para que estes não fiquem sobre o controle do utilizador;
- O simples facto da aplicação web estar disponível pode motivar ataques. A aplicação implementa vários controlos de segurança para evitar que estes ataques sejam bem sucedidos.

Na Figura 5.4 está esquematizada a arquitectura interna deste módulo.

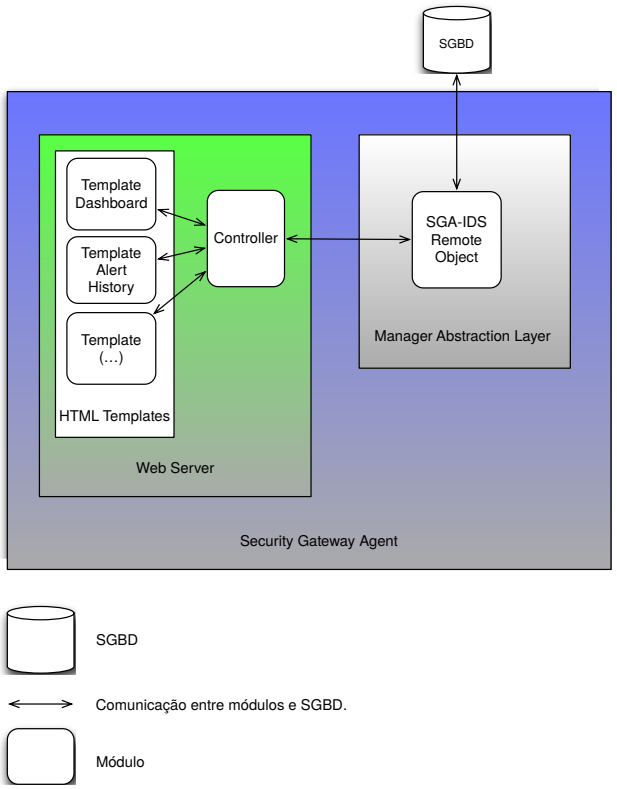

Figura 5.4: Arquitectura interna do módulo *Web Application*

Como se pode ver, e em resultado do esforço da arquitectura do SGA para a separação da interface com o utilizador do código que implementa as operações sobre os dados, a aplicação web pode mudar o endereço de rede a que se conecta de modo a controlar outra instância do SGA, sem mudar o endereço dela própria.

Outra vantagem desta abordagem é a da segurança da informação. Como a aplicação web está separada de todo o código que acede ao SGBD e do SGBD em si, se esta for comprometida por uma vulnerabilidade, o atacante não obterá acesso aos dados, mantendo-se assim a integridade e confidencialidade dos mesmos. Como se pode ver na figura, todos os acessos à base de dados são efectuados pelo módulo integrado na *Manager Abstraction Layer,* o *SGA-IDS Remote Object*.

O fluxo do módulo *Web Application* é o seguinte: Os pedidos HTTP são recebidos pelo *Web Server* com destino a uma das *templates*, o que automaticamente invoca uma chamada à função correspondente a essa *template* no módulo *Controller*. Este último utiliza o módulo *SGA-IDS Remote Object* para efectuar a operação em si, e retorna a resposta em HTML ao *Web Server*.

Na Tabela 5.1 apresenta-se a lista de funcionalidades implementada por este módulo. As funcionalidades sem identificador, foram implementadas por disponibilidade e interesse do autor apesar de não serem requeridas pela especificação.

Identificador	Descrição
ID-FU-002	Painel com situação actual/histórica (*Dashboard*).
ID-FU-003	Painel de eventos.
ID-FU-004	Histórico de eventos.
ID-FU-005	Geração de relatórios
ID-FU-006	Integração com módulo de resolução de incidentes.
ID-FU-007	Manutenção da base de dados.
ID-FU-009	Gerir grupos de activos.
	Lista de sensores e seu estado.
	Lista de activos que foram origem ou destino de eventos.

Tabela 5.1: Funcionalidades implementadas pelo módulo Web Application

5.3.3 Alert Receiver and Processor

(Implementa a funcionalidade ID-FU-001)

Este módulo tem a seu cargo a responsabilidade de implementar a funcionalidade principal da aplicação, a de receber eventos de segurança de sensores de rede e locais. Na Figura 5.5 apresenta-se a arquitectura interna deste módulo.

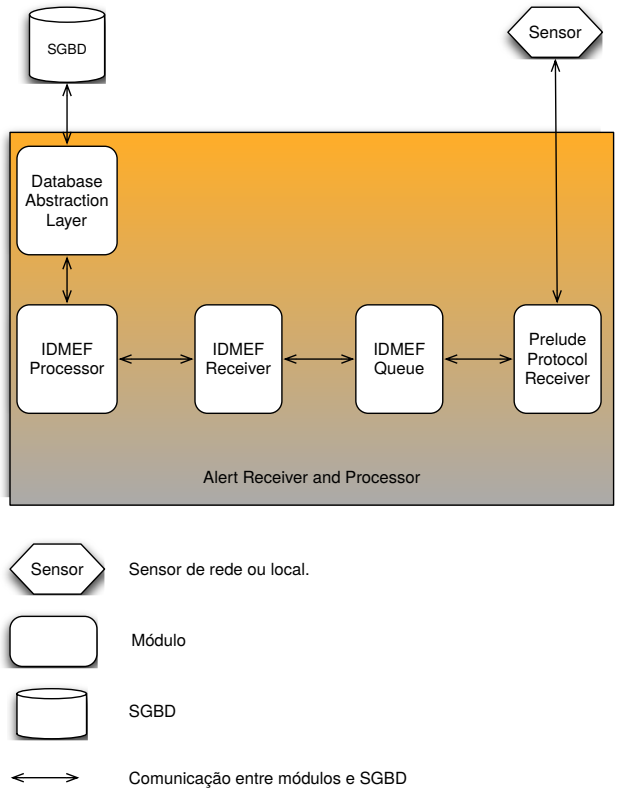

Figura 5.5: Arquitectura interna do módulo Alert Receiver and Processor

Como exposto no inicio deste Capítulo, todas as ferramentas que suportam enviar eventos em formato IDMEF utilizam para isso o protocolo *Prelude*. Como um dos requisitos para a aplicação consistia em receber eventos sem modificações aos sensores, o módulo *Prelude Protocol Receiver* presente na

figura implementa um servidor deste protocolo e, quando recebe um evento, transforma-o no documento IDMEF original e adiciona-o à fila FIFO implementada pelo módulo *IDMEF Queue.*

A razão de existir desta fila prende-se com o objectivo do autor, de isolar o sensor que reporta o evento de algum eventual atraso ou erro no processamento e armazenamento posterior deste, permitindo assim ao sensor seguir na sua execução sem esperar que todo o processo se conclua. Como se pode consultar no apêndice G, o protocolo Prelude utilizado pelos sensores não possui nenhuma forma de resposta de estado para indicar a recepção correcta do evento, e como tal, o evento é considerado reportado correctamente imediatamente a seguir a ser enviado, o que pode não ser o caso. Uma vez que a comunicação interna entre os módulos posteriores ao *Prelude Protocol Receiver* já possui a capacidade de reportar erros, este facto em conjunto com o sistema de fila de espera implementado, permite ao sistema evitar a perda de eventos. Mesmo se se tiver perdido conectividade com o SGBD, ou qualquer outro erro impedir o processamento do evento, este já se encontra na fila e será armazenado quando tal for possível. Esta ultima característica, é uma vantagem em relação à solução utilizada pela empresa anteriormente, que perdia os eventos no caso de falta de conectividade com o SGBD.

Esta fila por sua vez, é utilizada como fonte de eventos em IDMEF pelo módulo *IDMEF Receiver,* que obtém o evento e o transforma numa matriz associativa[4] que permite aceder a todas as *tags* XML e seus atributos de uma forma intuitiva e eficiente para o programador. Esta forma de matriz é utilizada então pelo módulo *IDMEF Processor* para extrair todos os dados a armazenar no SGBD. O armazenamento em si é feito pelo módulo *IDMEF Processor* através do módulo *Database Abstraction Layer,* que cumpre as mesmas funções que já foram apresentadas anteriormente para o módulo *SGA-IDS Remote Object*.

[4]Matriz em que os indices são *strings* em vez de inteiros.

5.3.4 Modelo de Dados

Nesta secção o autor pretende apresentar o modelo de dados utilizado pela aplicação e a justificação para este ter sido elaborado deste modo. Na Figura 5.6 apresenta-se o modelo de dados em si.

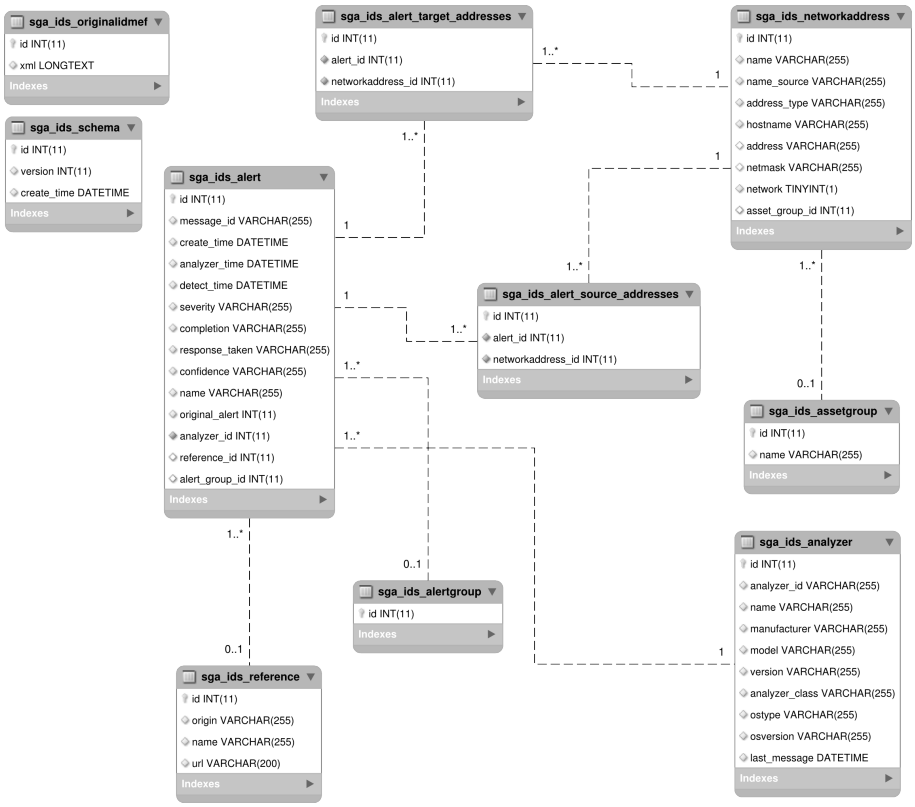

Figura 5.6: Modelo de dados

Os objectivos do modelo de dados escolhido foram, para além da integridade e segurança dos dados, a velocidade das operações sobre os mesmos. Seguidamente, o autor apresenta as justificações para algumas peculiaridades do modelo de dados.

Limitação do Número de Tabelas

Uma vez que a experiência da entidade acolhedora com o software que usava anteriormente foi que o grande número de eventos, rapidamente transformava qualquer operação de manutenção da base de dados numa espera interminável, o autor tomou algumas decisões conducentes a aumentar a velocidade de manipulação dos dados. Como tal, os dados armazenados foram limitados aos necessários para apresentação e tratamento na aplicação, e caso o utilizador pretenda aceder a mais detalhes, pode fazê-lo consultando o XML original do alerta, que também é armazenado. Esta decisão diminui o número de tabelas necessárias e, consequentemente, o número de relações a resolver pelo SGBD sempre que se pretende consultar, inserir ou apagar um evento, tornando assim as operações mais rápidas.

Detecção de Múltiplos Alertas Similares

Detectar uma sequência de eventos do mesmo tipo e com os mesmos activos de origem e destino é um bom indicador que algo se passa que exige a atenção da equipa de segurança. Para isso, foi criada uma tabela *sga_ids_alertgroup* que contém apenas uma chave primária. No processo de inserção, caso o alerta inserido anteriormente seja similar (mesma origem, alvo e tipo de ataque), é adicionada uma nova linha

à tabela *sga_ids_alertgroup* e o identificador dessa linha é usado como chave forasteira no alerta que está a ser inserido e no alerta anterior. Deste modo é possível agrupar alertas sem perder a informação referente a cada um deles, e alertar o utilizador para alertas recorrentes a ameaçar a sua infraestrutura..

Versão do Esquema de Base de Dados

De modo a prever actualizações futuras do modelo de dados e permitir à aplicação saber que versão está em uso, foi incluída uma tabela *sga_ids_schema* com uma só linha representando a versão do modelo de dados e a sua data de criação. Esta informação permite que no futuro, a aplicação possa migrar dados de uma versão do modelo de dados para outra.

5.4 Segurança da Aplicação

Como referido na secção 4.4.5 do Capítulo 4, a aplicação deverá ser segura e resistente a ataques externos.

De forma a garantir a conformidade com estes requisitos, procurar-se-á desenvolver o software de acordo com o OWASP ASVS3[61], uma série de *standards* criados para garantir a segurança da informação em aplicações Web, revelando aquelas que são as boas práticas e preocupações a ter aquando do seu desenvolvimento. Este *standard* foi desenvolvido com o intuito de que este seja utilizado como uma métrica e um guia pelos proprietários de aplicações e seus programadores, que poderão através dele definir qual o grau de integridade que pode ser atribuído a estas e os controlos de segurança sobre os quais as deverão construir, a fim de satisfazer os requisitos de segurança requeridos. Estes níveis são definidos de acordo com as preocupações que uma aplicação demonstra através dos seus mecanismos de segurança e de controlo, e são definidos através de 14 métricas apresentadas no Apêndice A. De seguida apresentam-se as estratégias utilizadas pela aplicação para a correcta implementação de cada métrica constante do ASVS:

- **V1. Arquitectura segura:** Na concepção da arquitectura apresentada neste Capítulo, a segurança da mesma foi uma preocupação central, nomeadamente:
  - Não utilizando protocolos de comunicação com vulnerabilidades estruturais, isto é, falhas que resultam da própria concepção do protocolo.
  - Tendo o cuidado de utilizar protocolos cifrados sempre que o uso de interfaces de rede fora do controlo do SGA seja necessário, ou possa ser futuramente.
  - Não implementando novas versões próprias de funcionalidades para as quais já exista uma implementação segura e de uso corrente.
- **V2. Autenticação:** O sistema de autenticação utilizado é herdado do SGA, e trata-se do sistema de autenticação fornecido pela *framework* Django, pelo que se trata de um sistema com provas dadas em termos de segurança.
- **V3. Gestão de sessões:** O sistema de gestão de sessões da aplicação web é também o utilizado pelo SGA, e fornecido pela *framework* Django. Mais uma vez é um sistema com provas dadas. Todas as páginas e funções correspondentes do módulo *Controller* da *Web Application*, implementam verificações de sessão utilizando para isso as funções fornecidas pela *framework*.
- **V4. Controlo de acessos:** Mais uma vez a *framework* utilizada revela-se uma vantagem, já que também fornece um sistema de controlo de acessos baseado num modelo *role-based*. Mais uma

vez, cada página e função correspondente no módulo Controller da Web Applications, implementa controles que verificam se o utilizador deve ter acesso aquela funcionalidade. Adicionalmente, os menus de acesso a funcionalidades a que o utilizador actual não tem permissões para aceder, não são apresentados a este [5].

- **V5. Validação de inputs:** Em termos de acesso ao SGBD, todos os acessos utilizam um módulo de abstracção do SGBD, implementado através da *framework* Django, que efectua automaticamente a validação necessária para evitar ataques de *SQL Injection*[6]. Todos os inputs são também validados de modo a evitar ataques de *cross-site scripting*[7], evitando a inclusão de código *JavaScript*.

- **V6. Encoding/Escaping de outputs:** O *escaping* de outputs é feito automaticamente pela *framework* Django, utilizada para implementar a aplicação web, aquando da elaboração das respostas HTTP.

- **V7. Criptografia:** O sistema utilizado para implementar o protocolo TLS baseia-se na biblioteca OpenSSL[62], que é utilizada pela grande maioria das implementações de TLS e possui um módulo certificado FIPS 140-2.[63]

- **V8. Logging e tratamento de erros:** Fruto do uso da *framework* DJango, todos os erros produzidos pela aplicação são armazenados em ficheiros de *logs*, incluindo toda a informação contextual do erro, e enviados para um endereço de email configurável, de modo à resolução do problema ser o mais rápida possível.

- **V9. Protecção de dados:** Foi tomado cuidado para que a aplicação não revele dados potencialmente sensíveis em termos de segurança em mensagens de erro e *logs*.

- **V10. Segurança na comunicação:** Como exposto para a métrica de arquitectura segura, todas as comunicações que possam sair do controlo do SGA são cifradas por TLS.

- **V11. Segurança HTTP:** Mais uma vantagem que advém do uso da framework Django, é a verificação automática da validade dos *requests* HTTP, protegendo a aplicação de ataques do tipo *cross-site site request forgery*[8].

Como se pode constatar, muitas das técnicas utilizadas derivam da integração com o SGA o que constitui uma outra vantagem já que tira partido do conhecimento já adquirido no desenvolvimento dos restantes módulos da *appliance*.

5.5 Segurança das Conexões

Para além do cuidado com a segurança da aplicação e a sua resistência a ataques externos, no âmbito deste projecto deve ser exercido um especial cuidado com a segurança dos dados que transitam por conexões possivelmente externas ao SGA. Como tal, todas as conexões que possam passar por interfaces de rede fora do controlo do SGA, são cifradas pelo protocolo TLS, que consiste num protocolo criptográfico

[5] São definidas funções que cada utilizador pode realizar no sistema e o controlo de acessos é feito com base nisso. Um utilizador pode possuir várias funções atribuídas.

[6] Ataque em que o atacante tenta inserir *queries* de SQL num campo de input da aplicação para que este seja executado pelo SGBD indevidamente.

[7] Ataque em que o atacante tenta inserir código *JavaScript* num input da aplicação de modo a que este seja executado pelo navegador web de outro utilizador.

[8] O atacante tenta fazer passar um *request* malicioso como se fosse um *request* válido, tentando assim realizar pedidos com as permissões de outro utilizador, sem o consentimento deste.

desenhado para assegurar um canal de comunicação seguro sobre infraestruturas inseguras.[64] Este protocolo é utilizado em larga escala na internet, em serviços com os mais variados fins.

Um dos requisitos para o uso deste protocolo, já que este usa uma infraestrutura de chaves publicas, é uma autoridade de certificados (CA - Certificate Authority) que será utilizada para criar os certificados e chaves privadas e publicas necessárias para a correcta implementação do protocolo.

Existem várias entidades que prestam este serviço, no entanto, utilizá-las acarretaria algumas desvantagens:

- Estas entidades estão fora do controlo da Dognædis.
- Seria necessária conectividade com a internet para criar os certificados. Podem existir cenários de implementação do SGA em que a *appliance* não tem ligação à internet e não é possível mudar esse facto;
- As licenças destes serviços são comerciais, motivando a procura de soluções alternativas.

Resultante destas desvantagens, será implementado um sistema de CAs para este projecto, de modo a garantir que a aplicação não está dependente de nenhuma autoridade de certificados externa. A implementação desta componente do projecto está documentada no Capítulo 6.

Como o módulo Alert Receiver and Processor é o único que possivelmente poderá ter conexões sobre infraestruturas fora do SGA, é neste módulo que se localizam as conexões a proteger, nomeadamente:

- Ligação entre o Sensor e o módulo Prelude Protocol Receiver;
- Ligação entre o módulo Prelude Protocol Receiver e o módulo IDMEF Processor.

Utilizando TLS para encriptar as conexões e identificar os intervenientes na mesma, garante-se que nenhum atacante poderá obter informação privilegiada através de capturas de tráfego ou ataques *Man-In-The-Middle*[9], e que a aplicação não se torna um vector de fuga de informação sobre a rede monitorizada, que poderia ser usada como base para elaborar ataques mais focados em activos mais importantes e como tal, mais gravosos para a infraestrutura.

[9] Ataque em que o atacante se faz passar pelo servidor, interceptando todo o tráfego, e possivelmente injectando tráfego malicioso.

5.6 Ferramentas e Tecnologias Adoptadas

Nas secções seguintes o autor apresenta as ferramentas e tecnologias adoptadas na execução deste projecto e que se revelaram as mais adequadas para que o projecto fosse implementado de acordo com a arquitectura planeada.

5.6.1 PostgreSQL

Apesar dos módulos já existentes utilizarem este SGBD, a arquitectura do SGA não inibe a utilização de outro motor de base de dados. Dito isto, o facto de o PostgreSQL já se encontrar instalado e configurado na *appliance* é uma vantagem já que evita a necessidade de mais um servidor a instalar. Apesar disso, foi efectuada uma avaliação desta opção em relação às outras que poderiam ser seleccionadas, para avaliar se se justificava uma outra opção. Depois de uma triagem dos vários SGBDs disponíveis rapidamente se chegou à conclusão que a única alternativa possível seria o *MySQL*, que é licenciado sobre a licença GPL, disponibilizando também uma versão comercial. Apesar do *MySQL* ser conhecido como mais rápido que o *PostgreSQL*, estes estudos utilizam muitas vezes o motor *MyISAM*[10] no *MySQL* que não respeita as regras ACID[11] enquanto que o *PostgreSQL* o faz completamente.[65] O respeito das regras ACID é uma grande vantagem neste projecto já que é essencial à manutenção da integridade dos dados, o que no contexto deste projecto é um requisito essencial.

Observando um estudo comparativo entre estas duas tecnologias, que utiliza as versões 5.1.30 do *MySQL* recorrendo ao motor *InnoDB* e à versão 8.3.7 do *PostgreSQL*, concluímos que os resultados ao nível de velocidade para escritas e leituras são até bastante próximos, não sendo a velocidade dos seus acessos, uma vantagem do *MySQL*.[66]

No que diz respeito à segurança de dados, ambos os SGBDs oferecem várias características que vão desde criptografia a múltiplos métodos de autenticação, sendo que o *PostgreSQL* oferece ainda suporte nativo para PLSQL (incluindo *views, triggers, constraints* e *stored procedures*) ao contrário do *MySQL*.[67]

Com base no exposto, foi escolhido o PostgreSQL como SGBD para este projecto.

5.6.2 Python

É uma linguagem inicialmente desenvolvida e publicada por Guido van Rossum em 1991, e está actualmente licenciada sobre a licença *Python Software Foundation License*[68], e suportada por um modelo de desenvolvimento comunitário dirigido por uma fundação sem fins lucrativos, *a Python Software Foundation*[69].

Suporta vários paradigmas de programação (orientada a objectos, imperativa e funcional), e cujas funcionalidades permitem uma gestão de memória eficaz. Suporta também a integração com diversas linguagens de programação. Possui actualmente um vasto leque de bibliotecas com funcionalidades adicionais ao núcleo fornecido pela própria *Python*, desenvolvidas por várias comunidades à volta da fundação, que lhe conferem facilidade de implementação de novas funções.

Apesar de ser uma restrição imposta pelo próprio SGA, possui algumas vantagens para a implementação deste projecto, entre as quais:

- **Portabilidade**: Suporta vários sistemas operativos (*Linux*, *FreeBSD*, *OpenBSD*, *Windows*, *Mac OS X*, etc), o que constitui uma vantagem. Particularmente, tem óptimo suporte para *OpenBSD* que é o sistema operativo sobre o qual corre o SGA como apresentado na secção 4.6.2;

[10] Indexed Sequential Access Method (ISAM)
[11] Atomicity, Consistency, Isolation, Durability

- **Interligação com *PostgreSQL***: Permite a integração com o SGBD PostgreSQL que será o utilizado para armazenar os dados da aplicação, conforme se apresentou na secção 5.6.1;
- **Capacidade de geração de PDF**: Um dos requisitos funcionais da aplicação é a geração de relatórios técnicos e administrativos, interessando portanto que o ambiente de desenvolvimento tenha bom suporte para a criação de ficheiros PDF contendo tabelas e gráficos. Existem várias bibliotecas com capacidade de geração de ficheiros PDF para *Python*, mas a escolhida pelo autor foi a *ReportLab* pelas razões já apresentadas na secção 5.3.1;
- **Integração com o SGA**: Apesar de ser possível utilizar outras linguagens para desenvolver módulos que se integram com o SGA, a utilização da mesma linguagem simplifica e acelera o processo. Este facto não constitui um factor decisivo, mas é uma mais valia na utilização desta linguagem;
- **Capacidade para desenvolvimento web:** O Python oferece várias plataformas Web para o desenvolvimento deste tipo de aplicações, entre as quais a Django, utilizada pelo SGA na sua *Interface Layer* para implementar a interface com o utilizador dos vários módulos. Apesar de existirem várias *frameworks* web para *Python*, o *Django* é a mais popular destas[70], e apresenta as seguintes caracteristicas:
  - Segue a arquitectura de *software model-view-controller*[12];
  - Adere ao principio DRY[13];
  - Permite que sejam criadas aplicações web, com uma forte componente de integração e reutilização de software, de grande performance (graças a um sistema de *caching* próprio) e desenhados de forma elegante, em termos muito inferiores àquelas que seriam necessários quando construídas sem recorrer à plataforma;
  - Oferece um sistema de ORM (*Object Relational Mapping*) próprio, que abstrai o SBGD especifico em uso e permite isolar o código da aplicação das alterações de SGBD;
  - Facilidade na criação de URLs limpos e intuitivos para o utilizador e os serviços habituais numa *framework* web tais como *templates*, filtros de texto, gerador e validador de formulários, e um sistema de autenticação e permissões extensível, simplificando o processo de implementação já que utilizando as funcionalidades implementadas pelo *Django*, o programador fica livre para se concentrar na lógica de negócio e não em implementar funções que deveriam ser fornecidas pela *framework*.;
  - Extenso suporte a nível literário e comunitário dada a sua vasta utilização no mercado;

5.6.3 Sistema Operativo

O sistema operativo em que a aplicação terá que correr é o OpenBSD, devido ao facto do SGA ser implementado sobre este ultimo. No entanto, o autor realizou uma pequena investigação sobre este sistema operativo para melhor entender porque foi escolhido para a base do SGA, que revelou características excelentes para suportar este projecto:

- É um sistema multi-plataforma, licenciado sobre a licença BSD, baseado no BSD UNIX[71] e desenhado exclusivamente com a segurança do sistema em mente[72].

[12] Modelo arquitectural que isola a lógica de negócio da aplicação da interface com o utilizador, permitindo alterar cada parte independentemente.

[13] Don't repeat yourself - http://c2.com/cgi/wiki?DontRepeatYourself

- Estável tanto a nível operacional como arquitectural;
  - Nível operacional: Garantido através de auditorias de código e suporte apenas a hardware seleccionado pela equipa e cujos são drivers são desenvolvidos pelos mesmos.
  - Nível arquitectural: É conservador em termos de implementação de novas funcionalidades e adição de novos pacotes, não estando sempre a ser redesenhado.
- O código desenvolvido é sempre sujeito a rigorosas regras e condutas de desenvolvimento, como a definição de utilização de código seguro, a realização de auditorias de código regulares efectuadas por equipas especializadas e dedicadas, com politicas de segurança muito bem definidas e testadas.

Capítulo 6

IMPLEMENTAÇÃO

Neste Capítulo o autor apresenta a descrição técnica da implementação dos módulos descritos no Capítulo 5 e a forma como estes interagem entre si. Para cada módulo são também incluídos os diagramas UML[73] relevantes.

6.1 Avaliação de Riscos

No Capítulo 4, secção 4.7, o autor documentou alguns riscos para a implementação, as suas consequências e as estratégias de mitigação caso se verificassem. Nesta secção apresentam-se os riscos que se verificaram e os passos tomados para os mitigar.

O risco ID-RI-001, em que o *core* do SGA poderia ainda estar em desenvolvimento e não estar estável o suficiente para integração, verificou-se e como tal o projecto limita-se a utilizar a arquitectura de comunicação do SGA, detalhada no Capítulo 5, mas foi obrigado a implementar um método próprio para controle do serviço na *Services Layer*.

O risco restante, o ID-RI-002, verificou-se também e como tal, a solução implementa um sistema de gestão de grupos de activos próprio.

Esta solução, garante que a integração completa no futuro será relativamente simples, já que o sistema já utiliza a parte mais importante da arquitectura do SGA, o modelo de comunicação.

6.2 Web Application

Relativamente à interface gráfica com o utilizador, interessa nesta secção detalhar o funcionamento do motor da aplicação web, e como este se integra com o SGA, e em segundo lugar, como foi obtido o aspecto gráfico da interface.

6.2.1 Framework Django

Como já foi apresentado no Capítulo 5, a aplicação web para interacção com o SGA-IDS foi desenvolvida sobre a plataforma de desenvolvimento web Django. Na Figura 6.1 apresenta-se como as várias classes da plataforma foram integradas no projecto e de que forma se tirou partido destas.

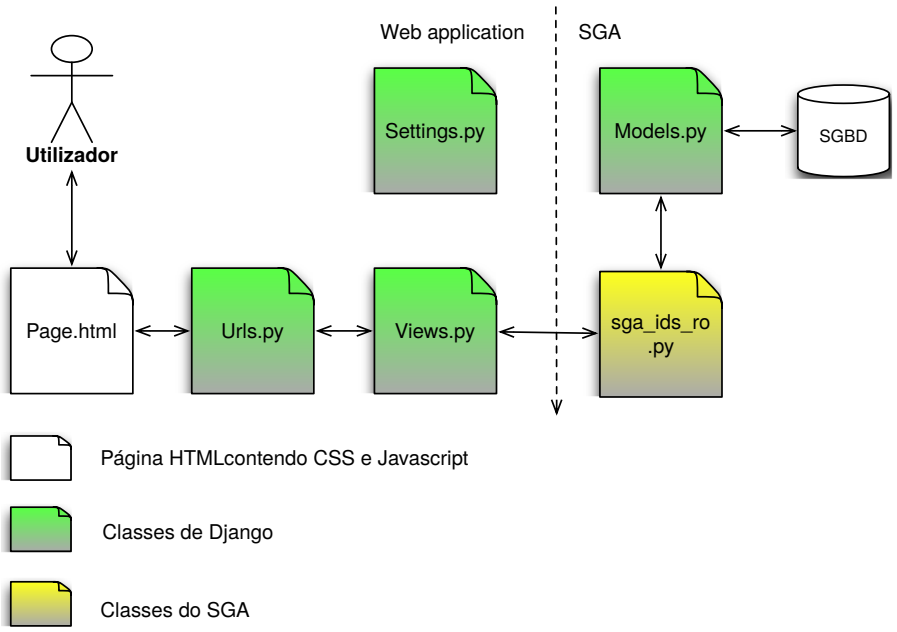

Figura 6.1: Implementação e integração com o SGA da aplicação web

A amarelo estão representadas as classes responsáveis por implementar as comunicações no SGA, e a verde as classes que fazem parte da framework Django e que são criados pela mesma aquando da sua instalação e configuradas posteriormente para corresponder às necessidades do programador. Seguidamente analisa-se cada uma destas classes, com o intuito de clarificar a implementação:

- **Settings.py:** Classe que contém um conjunto de configurações para o projecto Django, tais como definições de acesso ao SGBD, caminhos de localização de ficheiros, *plugins* do Django utilizados, entre outras;
- **Urls.py:** Este ficheiro contém a definição do esquema de mapeamento de URLs da aplicação. Cada URL que a aplicação suporta é definido neste ficheiro e associado a uma função no ficheiro Views.py. O mapeamento é feito com recurso a configurações do ficheiro Settings.py;
- **Views.py:** Controller do Django e que representa a camada lógica que interpreta os pedidos feitos pelo *browser*, chamando de seguida as funções necessárias no ficheiro sga_ids_ro.py utilizando o RPC;
- **Models.py:** Ficheiro que contém um conjunto de modelos, em que cada um corresponde a uma tabela na base de dados. Cada modelo contém ainda a especificação dos seus atributos e relações com os restantes modelos. Foi através destes modelos que foi criado o modelo de dados utilizado pela solução.
- **Pages.html:** Representa as páginas web com que o utilizador interage. Estas páginas podem ainda ser integradas com uma linguagem de templates da plataforma, que permite que as variáveis enviadas como resposta através de httpResponses possam ser manipuladas, iteradas e testadas com a finalidade de implementar renderização condicional.

Um pormenor que pode causar confusão, é que o ficheiro *sga_ids_ro.py* utiliza o ficheiro *Models.py* da *framework* Django para aceder à base de dados, apesar de não ser uma aplicação Django. Isto foi feito de modo que a classe *sga_ids_ro.py* pudesse beneficiar das funcionalidades de abstracção da base de dados e

queries programáticos fornecidos pelo sistema ORM do Django. Isso foi possível graças á possibilidade de usar somente essa parte da *framework*.

6.2.2 Bibliotecas JavaScript

Como referido no Capítulo 5, secção 5.1, a propósito da Interface Layer do SGA, a *template* fornecida pelo SGA para que o aspecto gráfico dos módulos seja consistente, define uma base em termos de bibliotecas JavaScript e CSS. O SGA faz uso da biblioteca Bootstrap para implementar esta base, e como tal para evitar duplicação e benificiar do conhecimento já existente na empresa, o SGA-IDS também a adoptou. O Bootstrap trata-se de uma ferramenta de *front-end* desenvolvida pelo Twitter, que oferece um conjunto de componentes personalizados e com aspecto coeso para inclusão numa página web, desenvolvidos através de CSS e HTML5.

Para além desta foram utilizadas a biblioteca de JavaScript *DataTables* [74], sobre a licença BSD, com a qual é possível implementar tabelas que oferecem uma vista paginada sobre um grande volume de dados fonte, carregando os dados dinamicamente e unicamente quando necessários, e permitindo também fazer pesquisas sobre uma única coluna ou várias simultaneamente. Esta biblioteca foi utilizada para implementar a renderização de informação nas páginas em que se disponibilizam listas de eventos, permitindo assim que o utilizador pesquise e ordene os dados de forma dinâmica e sem ser redireccionado para novas páginas, fornecendo assim uma interface mais rápida, intuitiva e agradável.

Para que tal fosse implementado, foi necessário implementar funções que em resposta aos pedidos gerados pelo biblioteca *DataTables*, gerassem as respostas no formato de um objecto JSON que depois é utilizado pela biblioteca para actualizar dinamicamente a tabela.

Alguns *screenshots* da aplicação podem ser consultados no apêndice C, que embora não apresentem exaustivamente todas as funcionalidades, foram incluídos para ilustrar o aspecto gráfico da mesma e como se processa a interacção do utilizador com o sistema.

6.2.3 Diagrama de Classes

Na Figura 6.2 apresenta-se o diagrama de classes do módulo *Web Application* tal como este módulo foi implementado de acordo com a arquitectura apresentada no Capítulo 5. Adicionalmente, na Figura 6.3 apresentam-se somente as classes *SGA_IDS_RemoteObject* e *Views*, para que sejam legíveis os seus atributos e métodos.

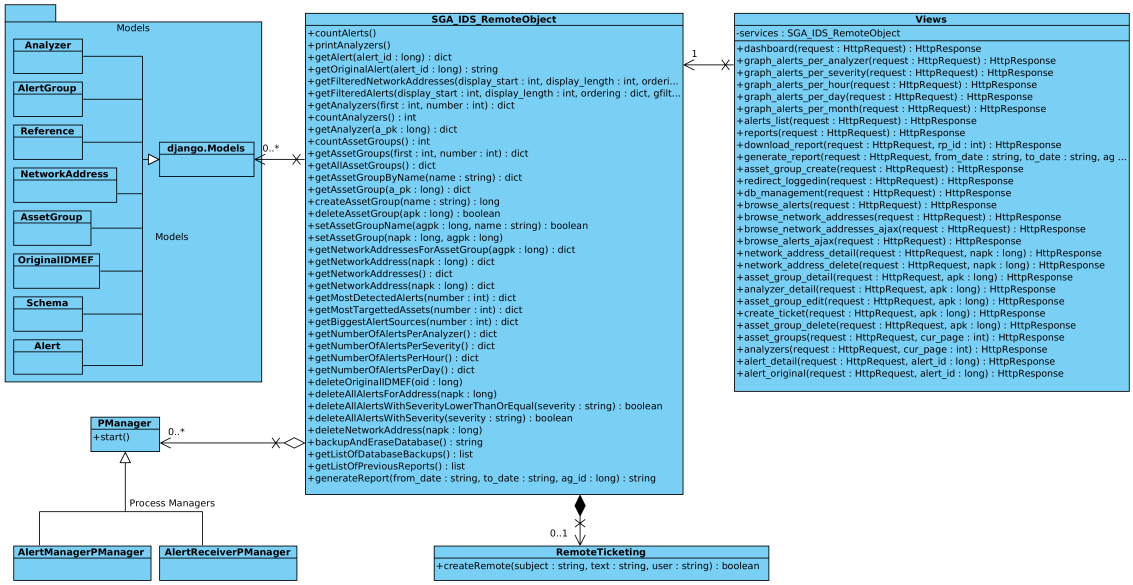

Figura 6.2: Diagrama de classes da Web Application

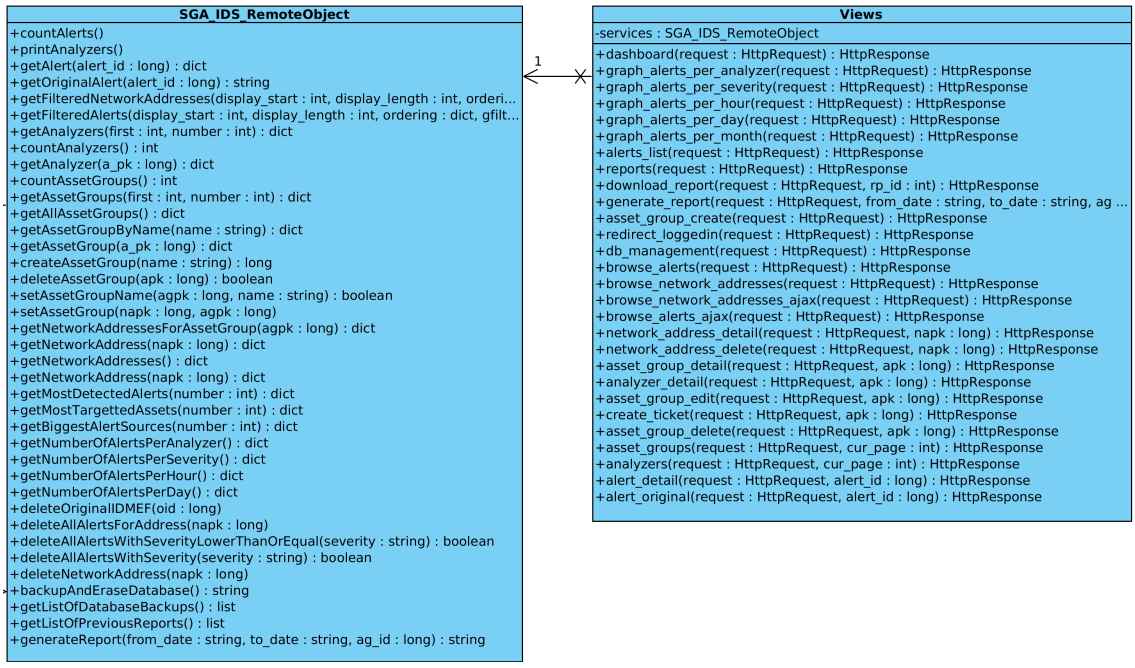

Figura 6.3: Classes SGA_IDS_RemoteObject e Views

Seguidamente apresentam-se as responsabilidades de cada classe e sua correspondência com os módulos apresentados no Capítulo 5:

SGA_IDS_RemoteObject Esta classe implementa toda a lógica de negócio e acesso ao SGBD, integrando-se na *Manager Abstraction Layer* do SGA. Todas as funções nele contidas são chamadas utilizando o

sistema de RPC do SGA já apresentado anteriormente no Capítulo 5. Como utiliza RPC, o objecto pode estar a correr numa máquina distinta da máquina que corre a aplicação web, bastando para isso que exista conectividade entre as duas.

django.Models Trata-se da classe base a partir da qual todas classes que acedem ao SGBD derivam e corresponde ao módulo *Database Abstraction Layer*. Esta classe faz parte da *framework Django* e serve de base a todas as classes que definem o modelo de dados. As classes derivadas não possuem os seus atributos documentados já que estes correspondem ao modelo de dados já apresentado na secção 5.3.4 do Capítulo 5.

RemoteTicketing Classe que implementa a interface com o sistema de gestão de incidentes RTIR de modo a permitir a criação de incidentes. Corresponde ao módulo *RTIR Integration* da arquitectura.

PManager Classe base que implementa as funções que permitem correr e parar processos, ou conjuntos de processos externos. Cada classe derivada desta, representa um processo ou conjunto de processos e são utilizadas pelo objecto *SGA_IDS_RemoteObject* para correr e parar os servidores *Alert Receiver* e *Alert Manager* que implementam o serviço *Alert Receiver and Processor* incluído na *Services Layer* do SGA, cuja implementação está documentada na secção 6.3.

Views Classe que contém todas as *Views* implementadas através da *framework* Django. Utiliza as funções do objecto *SGA_IDS_RemoteObject* através de RPC para implementar a lógica de negócio, como apresentado no Capítulo 5.

6.2.4 Diagrama de Sequência

Na Figura 6.4 esquematiza-se a sequência interna de chamadas do módulo *Web Application* no caso de o utilizador através do *Browser* abrir a página para a função de listar alertas.

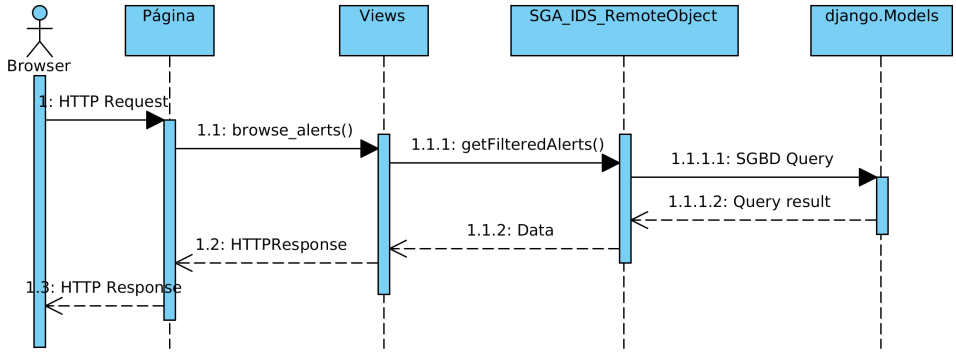

Figura 6.4: Diagrama de sequência Web Application

Do diagrama pode-se constatar que a sequência inicia-se quando o utilizador abre a página com o seu *browser*, o que provoca que o sistema de *templates* do Django invoque a função correspondente na classe *Views*, que neste caso se denomina *browse_alerts()*, com os valores dos campos do formulário presente na página passados como parâmetros à função. De seguida, a função *browse_alerts()* chama a função *get_filtered_alerts()* do objecto *SGA_IDS_RemoteObject* por RPC, que por sua vez consulta o SGBD e retorna os dados requisitados. Com base nos dados retornados, a função da classe *Views* cria um objecto *HTTPResponse* que o sistema de *templates* do Django utiliza para construir a página completa, que é enviada de volta ao *browser*.

Como se encontra esquematizado na figura, o acesso ao SGBD é feito utilizando sistema ORM do Django, apresentado no diagrama de classes da Figura 6.2. A sequência é similar para todas as outras páginas, com a excepção dos métodos chamados na classe *Views* e no objecto *SGA_IDS_RemoteObject*.

6.3 Alert Receiver and Processor

Esta secção tem o objectivo de detalhar a implementação deste que é o módulo mais importante da solução, já que implementa uma funcionalidade sem a qual todo o resto da solução seria inútil.

O autor ponderou as seguintes hipóteses de organização interna deste módulo:

- Implementar todos os módulos apresentados na arquitectura de modo a que estes fossem uma só unidade, utilizando o módulo *SGA-IDS Remote Object* da *Manager Abstraction Layer* para implementar o acesso ao SGBD. Esta solução tem a vantagem que do ponto de vista arquitectural iria concentrar o acesso à base de dados no referido módulo, diminuindo a duplicação. Tem no entanto a desvantagem que os módulos de recepção e processamento de eventos não poderiam funcionar sem conectividade com o SGA, ou quando o objecto remoto exibisse algum problema.

- Separar o servidor do protocolo Prelude e o processamento e armazenamento dos eventos em módulos que comuniquem num modelo de cliente-servidor, utilizando o módulo *SGA-IDS Remote Object* para aceder ao SGBD. Esta hipótese tem a vantagem de permitir ao servidor de protocolo Prelude separar-se do módulo de processamento dos eventos, mas continua a ter a desvantagem do armazenamento dos eventos depender de conectividade com o SGA.

- Separar o servidor do protocolo Prelude e o processamento e armazenamento dos eventos em módulos que comuniquem num modelo de cliente-servidor, e implementar o acesso ao SGBD no módulo de processamento de eventos. Esta alternativa tem a vantagem de maximizar a flexibilidade e eliminar a dependência sobre o SGA para o processamento dos eventos, mas tem a desvantagem de implicar alguma duplicação de código em relação às duas anteriores.

Tendo em conta que o protocolo Prelude não retorna nenhuma informação ao sensor sobre possíveis erros ocorridos no armazenamento, e que por isso interessa que o evento seja recebido pela solução o mais perto possível do sensor, e que há interesse em que a solução possa ser escalável a redes de grande dimensão, o autor escolheu a terceira alternativa.

Tirando partido da arquitectura de cliente-servidor escolhida acima, o módulo *Alert Receiver* permite enviar eventos para vários *Alert Manager* simultaneamente, o que em termos de fiabilidade permite armazenar os mesmos eventos em mais que um SGA para redundância.

Na Figura 6.5 apresenta-se a organização do módulo tal como implementado, incluindo a correspondência entre os módulos presentes na arquitectura apresentada no Capítulo 5 e os módulos presentes na implementação.

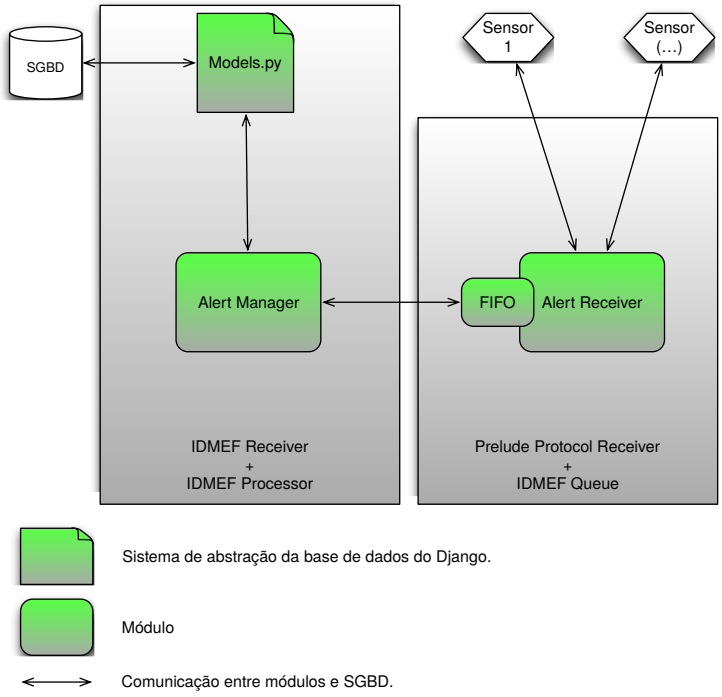

Figura 6.5: Implementação do módulo Alert Receiver and Processor

Como se pode ver na figura, o módulo *Alert Receiver* (correspondente aos módulos *Prelude Protocol Receiver* e *IDMEF Queue* da arquitectura) tem a função de receber os eventos do sensor e, através de filas de espera FIFO, enviá-los para o módulo *Alert Manager* (correspondente aos módulos *IDMEF Receiver* e *IDMEF Processor*) que tem a função de receber, processar e armazenar os eventos em IDMEF, utilizando para isso o sistema de ORM do Django.

Tomada a decisão de utilizar uma arquitectura cliente-servidor, foi necessário criar um protocolo de ligação entre os dois módulos referidos. O protocolo criado teve como objectivos a simplicidade e fornecer informação sobre o estado de armazenamento dos eventos. Em consonância com o objectivo de simplicidade, as mensagens são simples *strings* terminados em zero. A mensagem de resposta consiste num *string OK* para sucesso ou *ERROR* para erro, também terminadas em zero.

Na Figura 6.6 está representado o fluxo do protocolo, a que foi dado o nome de protocolo *Alert Manager*, através de um diagrama de sequência.

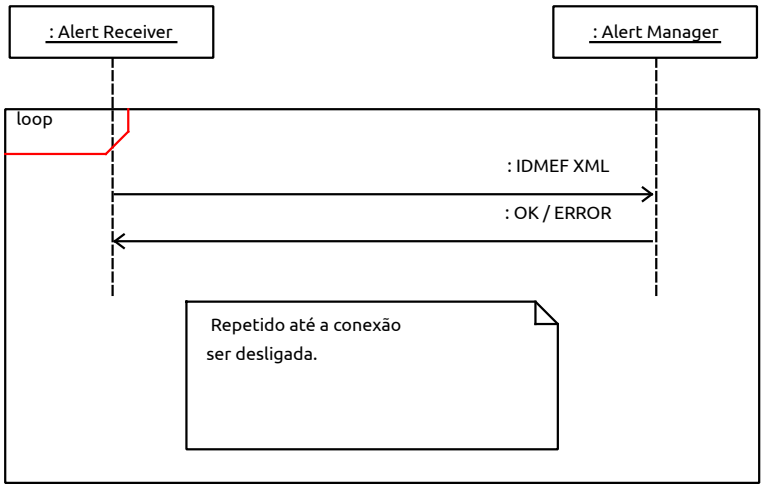

Figura 6.6: Sequência do protocolo *Alert Manager*

6.3.1 Certificate Authority

Como referido no Capítulo 5, dedicado à especificação da arquitectura, o protocolo Prelude implica a utilização de TLS, e foi tomada a decisão de implementar uma CA própria para criação dos certificados e chaves necessárias. Tomada a decisão de separar os módulos numa arquitectura cliente-servidor, tornou-se necessário proteger também essa conexão pelo que se reaproveitou a CA implementada para esse fim.

A CA foi implementada em *Bourne Shell*[75] e utiliza a ferramenta *certtool*, incluída com a biblioteca GnuTLS[64], para a criação de chaves e de certificados. De modo a ser completamente compatível com o sistema utilizado pela biblioteca *libprelude*, facilitando assim a configuração dos sensores, a organização e nomes dos ficheiros criados espelha o utilizado pela biblioteca referida. A organização e nomes referidos estão documentados no apêndice F.

6.3.2 Alert Receiver

Este módulo consiste num servidor TCP (Transmission Control Protocol) que tem como função receber mensagens IDMEF (Intrusion Detection Message Exchange Format)[76] codificadas segundo o protocolo *Prelude,* que todos os sensores com capacidade de comunicação de eventos em IDMEF utilizam, e posteriormente enviá-las para o módulo *Alert Manager*.

6.3.2.1 Diagrama de Classes

De modo a implementar a arquitectura apresentada no Capítulo 5, o módulo *Alert Receiver* foi implementado da forma representada pelo diagrama de classes da Figura 6.7.

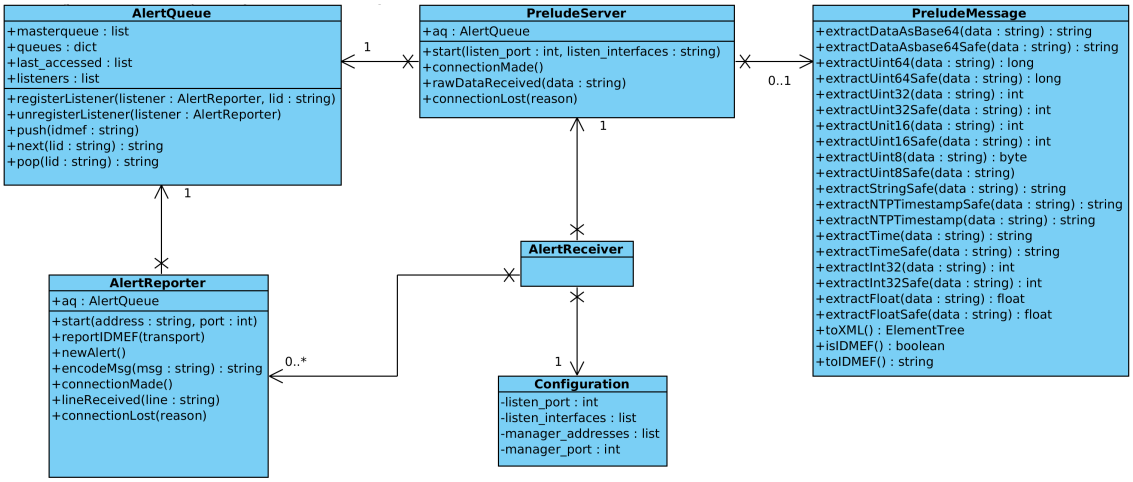

Figura 6.7: Diagrama de classes Alert Receiver

As várias classes têm as seguintes responsabilidades:

AlertReceiver Esta classe representa o ponto de entrada responsável por correr o módulo.

Configuration Classe que segue o padrão *Singleton*[1] *e* que obtem valores de configuração de um ficheiro *alert_receiver.cfg,* e os disponibiliza para as restantes classes.

PreludeServer Classe que implementa o servidor de protocolo *Prelude*. Utiliza para isso a biblioteca *Twisted*[8], que se trata de um módulo *Python* com a finalidade de simplificar a programação de servidores e clientes de rede. Para isso, adopta uma filosofia assíncrona utilizando *callbacks*[2]. Inclui também suporte para conexões SSL o que simplificou em muito a sua implementação. Um exemplo de servidor SSL implementado utilizando esta biblioteca pode ser consultado no apêndice E.

PreludeMessage Classe que efectua a transformação de uma mensagem do protocolo Prelude contendo um evento, para a sua forma original em XML. Para isso utiliza a biblioteca LXML, que se trata de um módulo Python que implementa uma API simples e rápida para a criação dinâmica de documentos XML e posterior acesso aos elementos e atributos do mesmo.[9]

AlertReporter Classe que implementa o cliente que envia os eventos em XML para o *Alert Manager* utilizando o protocolo *Alert Manager* descrito no inicio desta secção. Num processo *Alert Receiver*

[1] Classe que quando instanciada retorna sempre o mesmo objecto.
[2] Funções chamadas assincronamente quando um evento pré-definido ocorre.

haverá o mesmo número de instâncias desta classe que o número de *Alert Managers* a que o processo se conecta para reportar eventos.

AlertQueue Classe *Singleton* que implementa uma fila FIFO que recebe os eventos do objecto *PreludeServer* e é utilizada pelo objecto *AlertReporter* para obter os eventos a enviar para o *Alert Manager*.

6.3.2.2 Diagrama de Sequência

Na Figura 6.8 apresenta-se a sequência interna do módulo *Alert Receiver* desde que recebe um evento de um sensor até o envio bem sucedido do mesmo para o módulo *Alert Manager*.

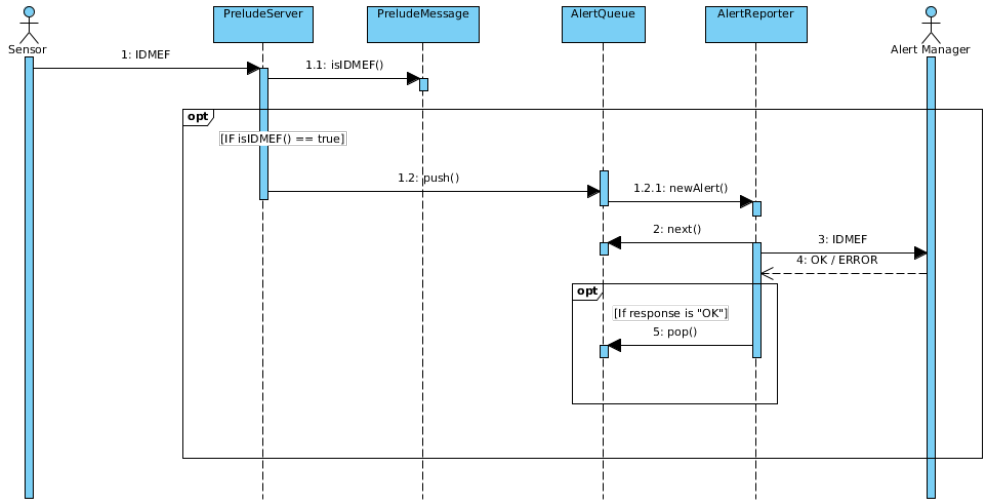

Figura 6.8: Diagrama de sequência Alert Receiver

O sensor envia um evento utilizando o protocolo Prelude que é recebido pelo objecto *PreludeServer*, que de seguida cria um objecto *PreludeMessage* com base na mensagem Prelude recebida. De seguida, chama a função *isIDMEF()* do objecto *PreludeMessage* criado, que verifica se a mensagem continha um evento IDMEF. Em caso positivo, a mensagem IDMEF é colocada na fila de espera utilizando a função push() do objecto *AlertQueue*. Este último chama a função *newAlert()* do objecto *AlertReporter* para sinalizar que este tem um novo evento para enviar. O objecto *AlertReporter* chama a função *next()* que devolve o próximo evento na fila de espera e envia-o para o *Alert Manager*. Se a resposta for de *OK,* é chamada a função *pop()* do objecto *AlertQueue* que retira a mensagem da fila. De notar que a função *next()* do objecto *AlertQueue* devolve o mesmo evento enquanto não for efectuada uma chamada *pop()*.

6.3.3 Alert Manager

Este servidor TCP/IP tem como função receber mensagens no formato XML IDMEF provenientes do *Alert Receiver*, processá-las e armazená-las no SGBD.

6.3.3.1 Diagrama de Classes

Na Figura 6.9 apresenta-se o diagrama de classes do módulo *Alert Manager* que representa a forma que foi implementada a arquitectura apresentada anteriormente no Capítulo 5.

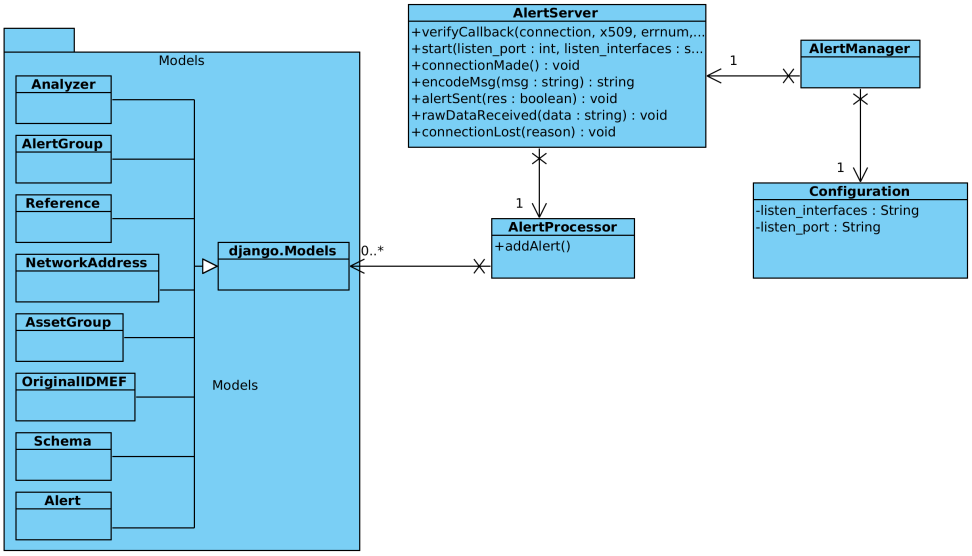

Figura 6.9: Diagrama de classes do *Alert Manager*

As várias classes têm as seguintes responsabilidades:

AlertManager Esta classe representa o ponto de entrada do programa.

Configuration Esta classe *Singleton* obtem os valores de configuração de um ficheiro *alert_manager.cfg* e disponibiliza-os para as restantes classes.

AlertServer Classe que implementa o servidor de protocolo *Alert Manager* de modo a receber os eventos do *Alert Receiver*. Utiliza para isso a biblioteca *Twisted* já descrita para o módulo *Alert Receiver*.

AlertProcessor Classe que implementa toda a lógica do processamento dos eventos IDMEF e seu armazenamento no SGBD. Utiliza para isso a biblioteca LXML, já descrita para o módulo *Alert Receiver,* para transformar o documento XML numa matriz associativa que contém os elementos e atributos, e seguidamente utiliza a matriz para extrair a informação a armazenar. O armazenamento é feito através do sistema de abstracção do SGBD descrito de seguida, e já referido também para o módulo *Web Application*.

django.Models Trata-se da classe base a partir da qual todas classes que acedem ao SGBD derivam. Como já referido para o módulo *Alert Receiver*, esta classe faz parte da *framework* Django, da qual este módulo utiliza apenas o sistema de ORM. As classes derivadas não possuem os seus atributos documentados já que estes correspondem ao modelo de dados já apresentado na secção 5.3.4 do Capítulo 5.

6.3.3.2 Diagrama de Sequência

Esquematiza-se na Figura 6.10 a sequência de chamadas do módulo Alert Manager desde que recebe um evento (IDMEF) do módulo *Alert Receiver*, passando pelo armazenamento deste e consequente envio do resultado de volta ao *Alert Receiver* (OK ou ERROR).

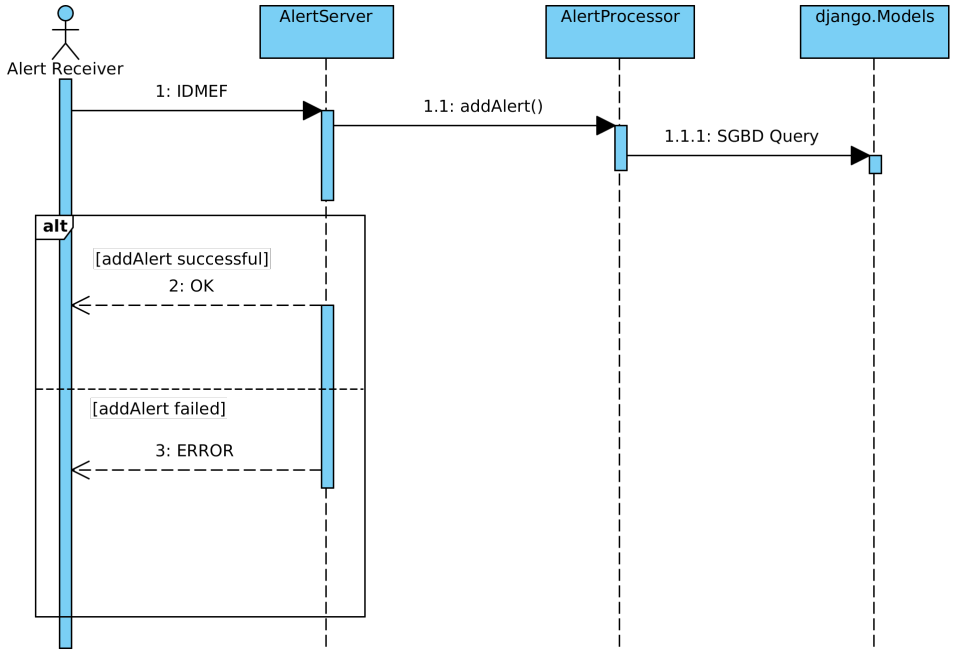

Figura 6.10: Diagrama de sequência *Alert Manager*

Como se pode ver, o *Alert Receiver* envia o IDMEF que é recebido pelo objecto *AlertServer,* que utiliza o método *addAlert()* do objecto *AlertProcessor* para o armazenar no SGBD. O objecto *AlertProcessor* utiliza o sistema de ORM do *Django,* referido na secção anterior, para comunicar com o SGBD. Dependo do sucesso ou falha no armazenamento, que é representado pelo valor de retorno de *TRUE* ou *FALSE* do método *addAlert()*, o objecto *AlertServer* envia a mensagem OK ou ERROR, respectivamente, para o *Alert Receiver*.

6.4 Implementação de Requisitos Não Funcionais

Nesta secção detalham-se as técnicas e características da solução, conducentes à implementação dos requisitos não funcionais especificados para a mesma. A validação dos requisitos especificados é feita no Capítulo 7.

6.4.1 Modularidade

Para se cumprir o requisito de Modularidade neste projecto, para além da modularidade em termos de a interface poder controlar outros SGAs, que resulta da integração com o SGA, interessa que sensores adicionais possam ser ligados à solução com a mínima modificação.

Nesse sentido, e resultado da implementação de uma CA própria, a única operação necessária para adicionar um novo sensor é utilizar a referida CA para criar a nova chave e certificado, e de seguida configurar o sensor para enviar os eventos para o SGA-IDS utilizando as chaves e certificados criados, que se encontram já no formato correcto. Não é necessário qualquer alteração de código, nem recomeçar

o *Alert Receiver*. Do mesmo modo, é possível acrescentar *Alert Receivers* a um *Alert Manager*, simplesmente criando novos certificados e sem recomeçar o *Alert Manager*.

6.4.2 Fiabilidade

No sentido de garantir a fiabilidade da solução, foram empregues várias técnicas:

- A ligação entre o *Alert Receiver* e *Alert Manager*, em caso de falha, tenta restabelecer-se periodicamente;
- Contrariamente ao protocolo *Prelude*, o protocolo *Alert Manager*, descrito na secção 6.3 do Capítulo 6, envia uma mensagem de resposta que indica o armazenamento correcto ou não do evento;
- O módulo *Alert Receiver* implementa um sistema de filas de espera para o envio de eventos para os *Alert Managers*, sendo atribuída uma fila a cada *Alert Manager*.

Estas características garantem que depois do evento dar entrada na solução implementada, este não será perdido já que se a resposta for de erro ou a conexão for desligada, o evento continua na lista de espera do *Alert Receiver* que o recebeu do sensor e este continuará a tentar enviá-lo para o *Alert Manager*. Deste modo garante-se que mesmo se a ligação entre o *Alert Receiver* e *Alert Manager* for pouco fiável não se perderão eventos.

6.4.3 Escalabilidade

Para além do modelo de dados simplificado, apresentado na secção 5.3.4 do Capítulo 5, resultaram das decisões de implementação apresentadas na secção 6.3 deste Capítulo, algumas características conducentes a uma maior escalabilidade:

- Há a possibilidade de haver mais que um *Alert Receiver*, e como tal este pode ser posicionado o mais perto possível do sensor de modo a que o envio dos eventos pelo mesmo seja mais rápido;
- A implementação das filas de espera já referidas, permite que o sensor não necessite de esperar que o evento seja enviado para o *Alert Manager* para passar ao envio do próximo evento, conduzindo assim a uma maior velocidade;
- A utilização da biblioteca *Twisted* para a implementação dos módulos Alert Receiver e Alert Manager, graças à utilização de *callbacks,* garante que nenhum dos dois faz espera activa em nenhuma parte do seu código, poupando assim os recursos da máquina onde correm.
- O código de acesso à base de dados do *Alert Manager*, é corrido num *thread*[3] separado que, quando terminado, invoca uma *callback*. Deste modo, o *Alert Manager* pode atender outros clientes enquanto o SGBD trabalha. Esta técnica é implementada graças ao suporte da biblioteca Twisted para *threads*.

[3]Código que executa simultaneamente, e que partilha os dados, do processo que o criou.

6.4.4 Usabilidade

Quanto ao desenho da interface com o utilizador e às regras utilizadas para isso, foram tidos em conta os critérios e normas de boas práticas definidas no livro *Designing The User Interface*[77], de Ben Shneiderman e Catherine Plaisant, que enumera os seguintes pontos base:

- *Learnability* - Requisitos que pretendam especificar a capacidade do utilizador em aprender a trabalhar com a interface.
  - Existe ao longo de toda a aplicação, a preocupação de que quando os campos ou funcionalidades não são claras, mostrar uma pequena caixa de texto, explicando em que consiste.
- *Understandability* - Métrica para especificação de técnicas capazes de manter o utilizador a par do funcionamento da aplicação e que lhe permitam entender a informação que lhe é apresentada.
  - Procurou-se organizar a aplicação através de diferentes menus, de modo a que o utilizador saiba sempre onde se encontra na mesma.
  - Em todas as tarefas é fornecida uma resposta ao utilizador, informando se a operação foi ou não bem sucedida;
- *Operability* - Requisitos que permitam uma melhor interacção entre a aplicação e o utilizador.
  - Sempre que foi criada uma nova interface, existiu a preocupação de validar os dados entrados nela, de modo a que não pudessem provocar inconsistências e comportamentos inesperados.
  - Na eventualidade de alguma operação ser mais demorada, o utilizador é avisado desse facto para que não pense que ocorreu alguma problema com a aplicação.
- *Attractiveness* - Especificação de necessidades ao nível visual.
  - A interface de aplicação foi desenhada com o objectivo de ser atractiva e apelativa.

6.5 Problemas no *Deployment*

Devido à maior disponibilidade de ferramentas de desenvolvimento, a solução foi desenvolvida em *Debian Linux* apesar do sistema operativo em que devia correr em produção ser o *OpenBSD*. Apesar dos cuidados que o autor teve durante o desenvolvimento só utilizando ferramentas e bibliotecas com suporte para *OpenBSD*, surgiram alguns problemas no processo de *deployment* final que se detalham de seguida, juntamente com as soluções encontradas. Depois de solucionados os problemas aqui descritos, a solução corre correctamente tanto em *OpenBSD* como *Debian Linux*.

6.5.1 Comando *date* do *OpenBSD*

Contrariamente à versão do *Linux*, o comando *date* do *OpenBSD* não possui um formato que retorne os microsegundos desde o UNIX *epoch* (1 de Janeiro de 1970). Esta funcionalidade era utilizada para o cálculo de *Analyzer IDs* durante a geração de certificados. Uma vez que não era desejável instalar outra versão do comando date a solução encontrada foi substituir microsegundos por segundos, o que apesar de não ser fiel ao modo de calculo original do *Prelude IDS*, é funcional.

6.5.2 Versões da Biblioteca GnuTLS

Inicialmente os servidores implementados utilizavam a biblioteca *GnuTLS* através do suporte do módulo *Python py-gnutls*. Esta solução funcionava perfeitamente em *Debian*, mas em *OpenBSD* a versão 2.6 da biblioteca *GnuTLS* já não existe e foi substituída pela versão 3.1 que não é compatível com o módulo *py-gnutls*. Inicialmente o autor tentou modificar o módulo para a versão 3.1 mas isso revelou-se demasiado dispendioso em termos de tempo. A solução final adoptada foi reimplementar o código TLS utilizando o módulo *PyOpenSSL* que utiliza a biblioteca *OpenSSL* que não exibe os mesmos problemas de versões entre o *Linux* e *OpenBSD*.

6.5.3 Verificação de Certificados

Infelizmente a reimplementação referida trouxe à superfície alguns bugs na biblioteca *libprelude* em termos da interacção com o *OpenSSL*, que não se manifestavam quando o servidor utilizava *GnuTLS*. Nomeadamente, a biblioteca assumia que a lista de certificados enviada pelo servidor teria somente um elemento, e a função *gnutls_certificate_get_ours()*, que deveria devolver o certificado enviado pelo cliente para o servidor, retornava *NULL* indevidamente. Estes bugs foram resolvidos com 2 *patches* para a biblioteca *libprelude*. Depois destes problemas estarem ultrapassados revelou-se um outro: a biblioteca *GnuTLS* 3.1, utilizada pela biblioteca *libprelude,* recusava-se a validar os certificados que o código de rede do *OpenSSL* enviava no *SSL Handshake*. Depois de algumas horas de investigação, executaram-se algumas modificações na CA implementada que resolveram o problema. De salientar que mais uma vez este problema não se manifestava no Linux.

Capítulo 7

VALIDAÇÃO E VERIFICAÇÃO

Neste Capítulo pretende-se expor os testes a que foi sujeita a aplicação para que a implementação dos requisitos funcionais e não funcionais fosse verificada e validada. Primeiramente são detalhados os testes de aceitação e seus resultados e de seguida validados os requisitos não funcionais através de testes, quando possível, ou de características da arquitectura implementada, quando a execução de testes é impossível.

Não foram criados testes unitários uma vez que o sistema é na sua maioria código de rede, código de acesso a base de dados e de interface com o utilizador, o que dificulta a elaboração deste tipo de testes e levaria a esgotar mais rapidamente os recursos temporais disponíveis. No entanto, o autor considera a sua implementação como uma boa adição futura ao trabalho, permitindo assim que sirvam de testes de regressão para alterações que se venham a implementar.

7.1 Testes de Aceitação

Durante a implementação o autor realizou testes não automatizados das funcionalidades que concluía para deste modo avaliar a sua correcta implementação. Na Tabela 7.1 apresenta-se a especificação e o resultado desses testes para cada requisito funcional. Como se pode constatar, todas as funcionalidades foram implementadas exceptuando a funcionalidade ID-FU-008 que foi classificada com prioridade 3 e que não foi possível implementar devido à falta de tempo para tal, ficando assim para trabalho futuro.

Para correr os testes foi necessário um sistema base comum a todos os testes especificados. O sistema base consiste num sistema OpenBSD com o software a ser testado instalado. O software foi instalado numa directoria arbitrária que se designa por /base/ nos pontos seguintes. Esta directoria pode ser qualquer uma desde que o utilizador a possa aceder e executar o *software*.

A configuração base do sistema é a seguinte e está esquematiza na Figura 7.1.

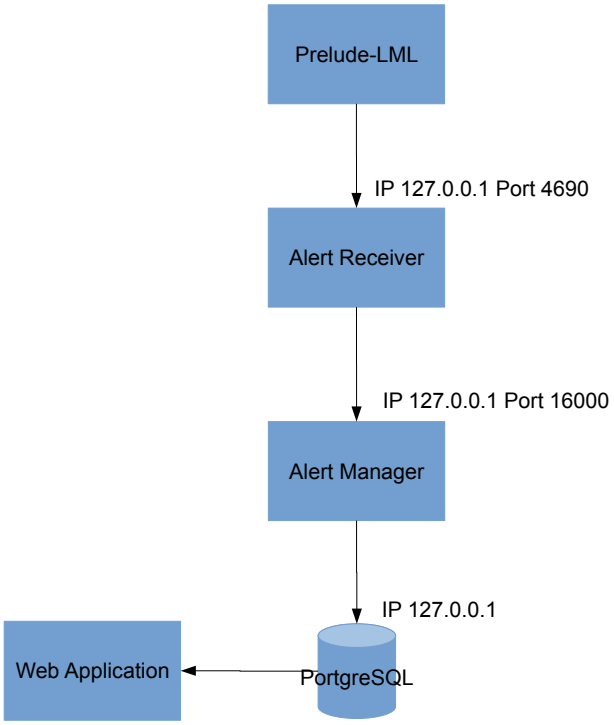

Figura 7.1: Ambiente de teste para os testes de aceitação

- 1 Sensor *Prelude-LML* configurado para reportar eventos para o IP 127.0.0.1 porto 4690. Este sensor deve estar configurado para enviar um evento sempre que um utilizador se autenticar como *root*. Apesar de possuir as configurações atrás referidas, os certificados e chaves criptográficas, necessárias à ligação entre este sensor e o *Alert Receiver*, ainda não devem estar criadas.
- 1 *Alert Receiver* configurado para aceitar conexões no IP 127.0.0.1 e porto 4690 e reportar os eventos recebidos para o IP 127.0.0.1 e porto 16000.
- 1 *Alert Manager* configurado para aceitar conexões no IP 127.0.0.1 e porto 16000 e armazenar os eventos num SGBD *PortgreSQL* a correr localmente.
- 1 *Web Application* a correr localmente e a aceitar conexões no IP 127.0.0.1 porto 8000 e a obter os dados do mesmo SGBD *PostgreSQL* em que o *Alert Manager* acima armazena os eventos.
- Os certificados e chaves necessárias para o *Alert Receiver* se ligar ao *Alert Manager* já se encontram criados e a ligação já foi testada.

7.1.1 ID-TA-001: Receber alertas de sensores

Pré-condições	1. Configuração base.
Procedimento	1. Entrar na directoria "/base/alert_receiver/certtool_ca/". 2. Correr o comando "./cactl –add prelude-lml". 3. Copiar os ficheiros da directoria "/base/alert_receiver/certs/prelude-lml" para a directoria "/etc/prelude/profile/prelude-lml/". 4. Correr o sensor *Prelude-LML*. 5. Autenticar-se como *root* no sistema operativo.
Resultado esperado	É recebido um evento pelo *Alert Receiver*, este é enviado para o *Alert Manager* que o armazena no SGBD o que o torna visível na consola web.
Resultado	✓
Requisitos	ID-RF-001, ID-RF-002, ID-RF-003

7.1.2 ID-TA-002: Adicionar sensores

Pré-condições	1. Configuração base.
Procedimento	1. Entrar na directoria "/base/alert_receiver/certtool_ca/". 2. Correr o comando "./cactl –add prelude-lml". 3. Copiar os ficheiros da directoria "/base/alert_receiver/certs/prelude-lml" para a directoria "/etc/prelude/profile/prelude-lml/". 4. Correr o sensor *Prelude-LML*.
Resultado esperado	Uma nova linha correspondente ao sensor aparece na página "Analyzers" da aplicação web.
Resultado	✓
Requisitos	ID-RF-001

7.1.3 ID-TA-003: Remover sensores

Pré-condições	1. Configuração base. 2. O sensor Prelude-LML com os certificados e chaves resultantes do teste ID-TA-002.
Procedimento	1. Obter o numero de série em hexadecimal do certificado do sensor Prelude-LML que se pretende remover através do comando "certtool -i < /base/alert_receiver/certs/prelude-lml/client.keycrt | grep Serial". 2. Inserir esse valor numa linha no ficheiro /base/alert_receiver/certs/clients.untrusted.
Resultado esperado	O *Alert Receiver* irá fechar todas as tentativas de ligação do sensor.
Resultado	✓
Requisitos	ID-RF-002, ID-RF-003

7.1.4 ID-TA-004: Estatisticas

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver o *dashboard*. 2. Carregar na opção "Dashboard" do menu.
Resultado esperado	A consola web gera e apresenta os gráficos de estatísticas.
Resultado	✓
Requisitos	ID-RF-004

7.1.5 ID-TA-005: Actualização automática

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver eventos. 2. Carregar na opção "Realtime Alerts" do menu.
Resultado esperado	A consola mostra os últimos eventos recebidos e actualiza-se automaticamente periodicamente.
Resultado	✓
Requisitos	ID-RF-005

7.1.6 ID-TA-006: Alternar a actualização automática

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver eventos. 2. Carregar na opção "Realtime Alerts" do menu. 3. Carregar no *switch* "Actualização automática".
Resultado esperado	A consola mostra os últimos eventos recebidos, mas não se actualiza automaticamente após o utilizador desligar o *switch*.
Resultado	✓
Requisitos	ID-RF-006

7.1.7 ID-TA-007: Filtragem por grupo de activos no painel de alertas

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para criar grupos de activos e ver eventos. 2. Carregar na opção “Asset Groups” do menu. 3. Carregar no botão “Create”. 4. Definir um nome e carregar no botão “Create”. 5. Adicionar alguns endereços de rede ao selector “Network Addresses”. 6. Carregar no botão “Submit”. 7. Escolher a opção “Realtime Alerts”. 8. Escolher um grupo de activos no selector “Asset Groups”.
Resultado esperado	A consola mostra os últimos eventos recebidos referentes (têm como origem ou destino) ao grupo de activos seleccionado.
Resultado	✓
Requisitos	ID-RF-007, ID-RF-017, ID-RF-018

7.1.8 ID-TA-008: Filtragem por tempo

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver eventos. 2. Carregar na opção “*Browse Alerts*” do menu. 3. Escolher uma data máxima ou/e data mínima.
Resultado esperado	A consola mostra os eventos recebidos durante o intervalo de tempo definido.
Resultado	✓
Requisitos	ID-RF-008

7.1.9 ID-TA-009: Filtragem por grupo de activos no histórico de alertas

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para criar grupos de activos e ver eventos. 2. Carregar na opção “Asset Groups” do menu. 3. Carregar no botão “Create”. 4. Definir um nome e carregar no botão “Create”. 5. Adicionar alguns endereços de rede ao selector “Network Addresses”. 6. Carregar no botão “Submit”. 7. Escolher a opção “Browse Alerts”. 8. Escolher um grupo de activos no selector “Asset Groups”.
Resultado esperado	A consola mostra os eventos recebidos referentes (têm como origem ou destino) ao grupo de activos seleccionado.
Resultado	✓
Requisitos	ID-RF-009, ID-RF-017, ID-RF-018

7.1.10 ID-TA-010: Filtragem por tempo nos relatórios

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para criar relatórios. 2. Carregar na opção “Reports” do menu. 3. Escolher uma data máxima ou/e data mínima nos selectores “From:” e “To:”. 4. Carregar no botão “Generate”.
Resultado esperado	A consola cria o relatório sobre os eventos recebidos durante o intervalo de tempo definido.
Resultado	✓
Requisitos	ID-RF-010

7.1.11 ID-TA-011: Filtragem por grupo de activos nos relatórios

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para criar grupos de activos e ver eventos. 2. Carregar na opção "Asset Groups" do menu. 3. Carregar no botão "Create". 4. Definir um nome e carregar no botão "Create". 5. Adicionar alguns endereços de rede ao selector "Network Addresses". 6. Carregar no botão "Submit". 7. Escolher a opção "Reports" do menu. 8. Escolher um grupo de activos no selector "Asset Group". 9. Carregar no botão "Generate".
Resultado esperado	A consola cria o relatório sobre os eventos referentes (têm como origem ou destino) ao grupo de activos seleccionado.
Resultado	✓
Requisitos	ID-RF-011, ID-RF-017, ID-RF-018

7.1.12 ID-TA-012: Criação de incidente

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver eventos e criar incidentes. 2. Carregar na opção "*Browse Alerts*" do menu. 3. Carregar no botão "Actions" de um dos eventos apresentados e escolher a opção "Create ticket". 4. Introduzir um assunto na caixa "Subject" 5. Editar a informação presente na caixa "Description" se necessário. 6. Carregar no botão "Submit".
Resultado esperado	O ticket correspondente ao incidente é criado no sistema *RTIR*. A web console apresenta uma mensagem de sucesso.
Resultado	✓
Requisitos	ID-RF-012

7.1.13 ID-TA-013: Eliminação de activos

Pré-condições	1. Configuração base. 2. Existir pelo menos um endereço de rede inserido na SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para ver endereços de rede e apagar endereços de rede. 2. Carregar na opção "Network Addresses" do menu. 3. Carregar no botão "Actions" de um dos endereços de rede e esolher a opção "Delete". 4. Escolher "OK" na mensagem de confirmação que o sistema apresenta de seguida.
Resultado esperado	Os eventos referentes ao endereço de rede são eliminados do SGBD e o próprio endereço de rede é também eliminado e removido do seu grupo de activos, caso tivesse um. A consola apresenta uma mensagem de sucesso nesse sentido.
Resultado	✓
Requisitos	ID-RF-013

7.1.14 ID-TA-014: Rotação da base de dados

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para gerir a base de dados. 2. Carregar na opção "*Database Management*" do menu. 3. Carregar no botão "Backup". 4. Carregar no botão "OK" da menagem de confirmação apresentada pela consola.
Resultado esperado	É gravado um ficheiro de backup da base de dados e a base de dados é reinicializada sem dados.
Resultado	✓
Requisitos	ID-RF-014

7.1.15 ID-TA-015: Eliminação de eventos por severidade

Pré-condições	1. Configuração base. 2. Existirem eventos já armazenados no SGBD.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para gerir a base de dados. 2. Carregar na opção "*Database Management*" do menu. 3. Escolher uma severidade no selector à direita 4. Carregar no botão "OK" da mensagem de confirmação apresentada pela consola.
Resultado esperado	São apagados os eventos correspondentes à selecção.
Resultado	✓
Requisitos	ID-RF-014

7.1.16 ID-TA-016: Criar grupos de activos

Pré-condições	1. Configuração base.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para criar grupos de activos. 2. Carregar na opção "*Asset Groups*" do menu. 3. Carregar no botão "Create". 4. Definir um nome para o novo grupo de activos na caixa "Name". 5. Carregar no botão "Create".
Resultado esperado	É criado o grupo de activos com o nome definido.
Resultado	✓
Requisitos	ID-RF-017

7.1.17 ID-TA-017: Apagar grupos de activos

Pré-condições	1. Configuração base. 2. Pelo menos um grupo de activos já criado.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para apagar grupos de activos. 2. Carregar na opção "*Asset Groups*" do menu. 3. Carregar no botão "Actions" de um grupo de activos e escolher a opção "Delete". 4. Escolher "OK" na mensagem de confirmação.
Resultado esperado	O grupo de activos é apagado e a consola apresenta uma mensagem nesse sentido.
Resultado	✓
Requisitos	ID-RF-017

7.1.18 ID-TA-018: Editar grupos de activos

Pré-condições	1. Configuração base. 2. Pelo menos um grupo de activos já criado.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para editar grupos de activos. 2. Carregar na opção "*Asset Groups*" do menu. 3. Carregar no botão "Actions" de um grupo de activos e escolher a opção "Edit". 4. Modificar o nome ou os endereços de rede pertencentes ao grupo. 5. Carregar no botão "Submit".
Resultado esperado	O grupo de activos é modificado e a consola apresenta uma mensagem nesse sentido.
Resultado	✓
Requisitos	ID-RF-018

7.1.19 ID-TA-019: Adicionar activos a um grupo

Pré-condições	1. Configuração base. 2. Pelo menos um grupo de activos já criado. 3. Pelo menos um endereço de rede que não pertence a nenhum grupo de activos.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para editar grupos de activos. 2. Carregar na opção "*Asset Groups*" do menu. 3. Carregar no botão "Actions" de um grupo de activos e escolher a opção "Edit". 4. Adicionar o endereço de rede ao grupo de activos através do selector "Network Addresses". 5. Carregar no botão "Submit".
Resultado esperado	O endereço é adicionado ao grupo de activos e a consola apresenta uma mensagem de sucesso.
Resultado	✓
Requisitos	ID-RF-019

7.1.20 ID-TA-020: Remover activos de um grupo

Pré-condições	1. Configuração base. 2. Pelo menos um grupo de activos já criado com pelo menos um endereço de rede.
Procedimento	1. Aceder à consola web autenticado como um utilizador com permissões para editar grupos de activos. 2. Carregar na opção "*Asset Groups*" do menu. 3. Carregar no botão "Actions" de um grupo de activos e escolher a opção "Edit". 4. Remover um endereço de rede através do selector "Network Addresses". 5. Carregar no botão "Submit".
Resultado esperado	O endereço é removido do grupo de activos e a consola apresenta uma mensagem de sucesso.
Resultado	✓
Requisitos	ID-RF-019

7.1.21 Resultados

Na Tabela 7.1 apresenta-se, para cada requisito funcional, se foi aceite ou não após a execução dos testes de aceitação correspondentes. Como se pode constatar, apenas o requisito ID-RF-016 não foi aceite porque não foi implementado devido à insuficiência de tempo disponível, como já referido em capítulos anteriores.

Requisito	**Resultado**
ID-RF-001: Receber alertas de sensores	✓
ID-RF-002: Adicionar e remover sensores	✓
ID-RF-003: Permissões de sensores	✓
ID-RF-004: Estatísticas	✓
ID-RF-005: Actualização automática	✓
ID-RF-006: Alternar a actualização automática	✓
ID-RF-007: Filtragem por grupo de activos	✓
ID-RF-008: Filtragem por tempo	✓
ID-RF-009: Filtragem por grupo de activos	✓
ID-RF-010: Filtragem por tempo	✓
ID-RF-011: Filtragem por grupo de activos	✓
ID-RF-012: Criação de incidente	✓
ID-RF-013: Eliminação de activos	✓
ID-RF-014: Rotação da base de dados	✓
ID-RF-015: Eliminação de eventos por severidade	✓
ID-RF-016: Obtenção da captura de tráfego	Não implementado.
ID-RF-017: Criar e apagar grupos de activos	✓
ID-RF-018: Editar grupo de activos	✓
ID-RF-019: Adicionar e remover activos de um grupo	✓

Tabela 7.1: Resultados dos testes funcionais

7.2 Fiabilidade

Este requisito funcional foi tido em conta na arquitectura e, apesar de não ser testado fisicamente neste Capítulo, este é completamente cumprido em resultado das decisões arquitecturais e de implementação tomadas em relação aos protocolos e módulos criados, como descrito nos capítulos 5 e 6.

7.3 Segurança

Este requisito não funcional ou atributo de qualidade, foi um dos principais focos da arquitectura e implementação do projecto, o que originou várias decisões:

- Durante a implementação, o autor trabalhou para seguir as regras do OWASP ASVS, descritas no apêndice A, que consistem num conjunto de guias que um programador deve seguir para criar uma aplicação web segura.
- Todas as ligações que podem passar por infraestruturas inseguras, são encriptadas por TLS.
- O próprio SGBD utilizado, o PostgreSQL, encripta a conexão através do TLS.

Assim, a segurança da solução é uma consequência das decisões e método no desenvolvimento da mesma.

7.4 Performance e Escalabilidade

No contexto deste projecto um dos limites à escalabilidade do mesmo é a velocidade com que consegue receber e armazenar os eventos. Consequentemente, uma maior performance nessas funções aumenta a carga que a solução consegue suportar.

Para testar a performance do sistema, e consequentemente a escalabilidade do mesmo, foi criado um sensor simples que se conecta simultaneamente um número predeterminado de vezes a um *Alert Receiver*, simulando assim esse numero de clientes, e envia o mesmo evento 100 vezes por cada conexão com um período de 0.1 segundos entre eventos, armazenando o tempo exacto em que enviou o mesmo. O *Alert Manager* foi modificado para gravar o tempo exacto do armazenamento do evento. Todos os processos estão a correr na mesma máquina e comunicam através de TCP/IP pela *interface Loopback*. O teste foi corrido para 2,4,8,16,32 e 64 clientes, e 5 vezes para cada numero de clientes, sendo que os números que se apresentam são a média dos resultados obtidos. Deste modo é possível testar o efeito do aumento de número de clientes no tempo de armazenamento dos eventos, sem que a velocidade da rede interfira. Na Figura 7.2 está esquematizado o ambiente de teste.

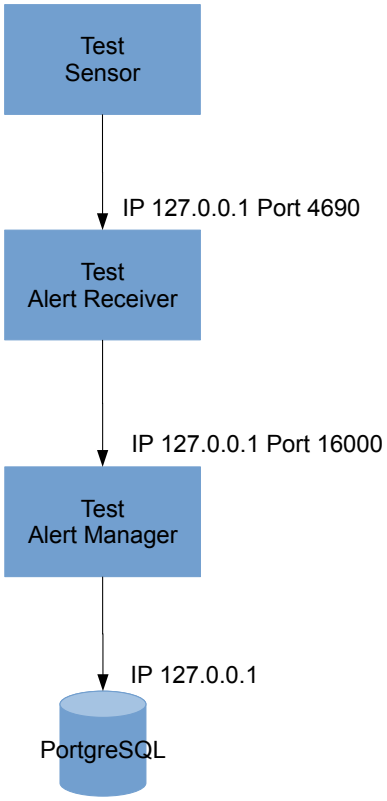

Figura 7.2: Ambiente de teste para o teste de performance e escalabilidade

A máquina sobre a qual foram corridos os testes foi a seguinte:

CPU Intel Mobile i5 M-450 2.4GHz

RAM 8GB DDR3

Disco 256GB Crucial M4 SSD

Sistema Operativo OpenBSD 5.3

SGBD PostgreSQL 9.1 com as configurações por omissão.

De seguida apresentam-se os gráficos dos resultados obtidos pelo processo de testes. A Figura 7.3 representa o tempo em segundos mínimo, máximo e médio que os eventos têm que esperar até serem armazenados pelo *Alert Manager* depois de serem enviados para o *Alert Receiver,* para cada número de clientes. A Figura 7.4 representa o tempo necessário para o envio do total dos eventos e o armazenamento dos mesmos, para cada número de clientes.

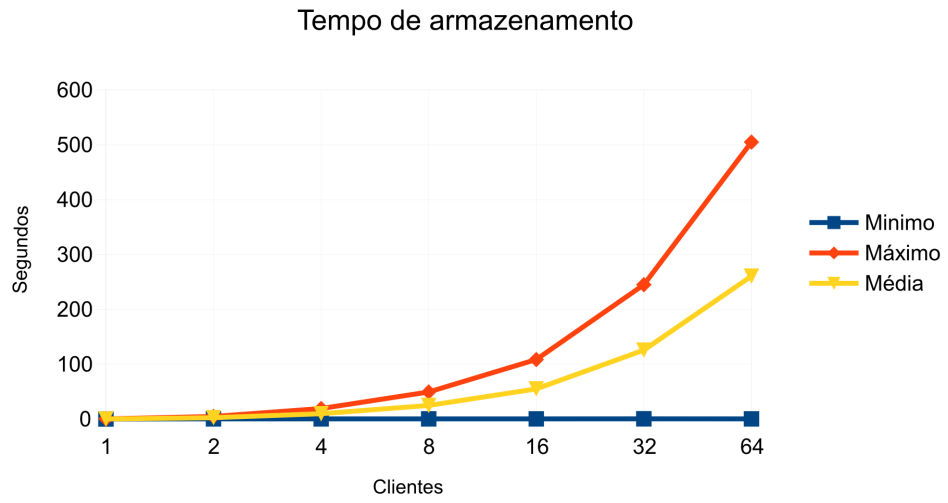

Figura 7.3: Tempo de armazenamento

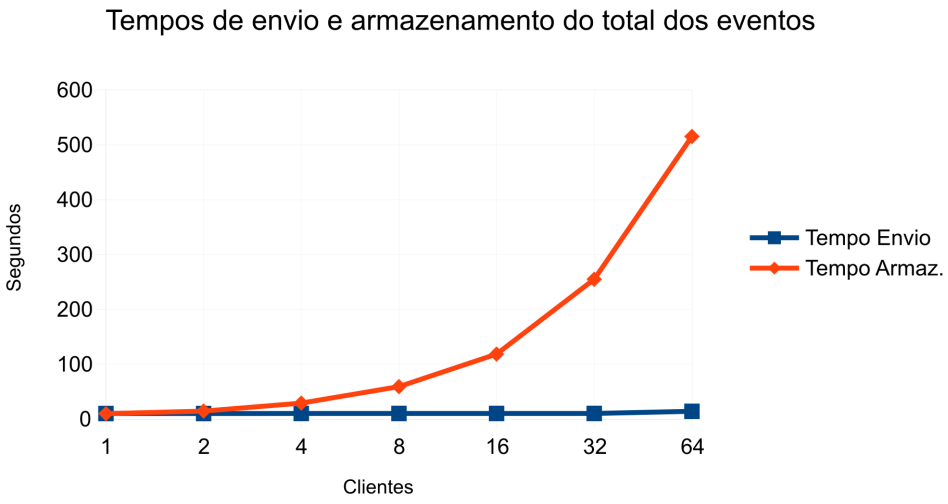

Figura 7.4: Tempo de envio e armazenamento do total dos eventos

Como esperado, o tempo de envio dos eventos do sensor para o *Alert Receiver* é muito pequeno já que não está dependente do acesso a disco ou limitado pela velocidade da rede. Já o tempo de armazenamento aumenta de acordo com o número de clientes simultâneos, sensivelmente na mesma proporção, uma vez que o número de eventos a armazenar é dobrado com cada aumento de número de clientes. A grande discrepância entre os valores mínimos e máximos de armazenamento de eventos apresentada na Figura 7.3 pode ser explicada pela dependência sobre o SGBD do armazenamento dos eventos, o que implica que, apesar do evento já ter sido recebido pelo *Alert Receiver*, este ainda não foi enviado para o *Alert Manager* já que o primeiro ainda se encontra à espera da resposta ao evento anterior e esta só

será enviada depois do armazenamento no SGBD. Deste teste pode-se concluir que, como esperado, a capacidade e performance do armazenamento de eventos pelo sistema é dependente da performance do SGBD utilizado.

Na Figura 7.5 apresenta-se o gráfico do numero de eventos armazenados, por segundo, para cada número de clientes.

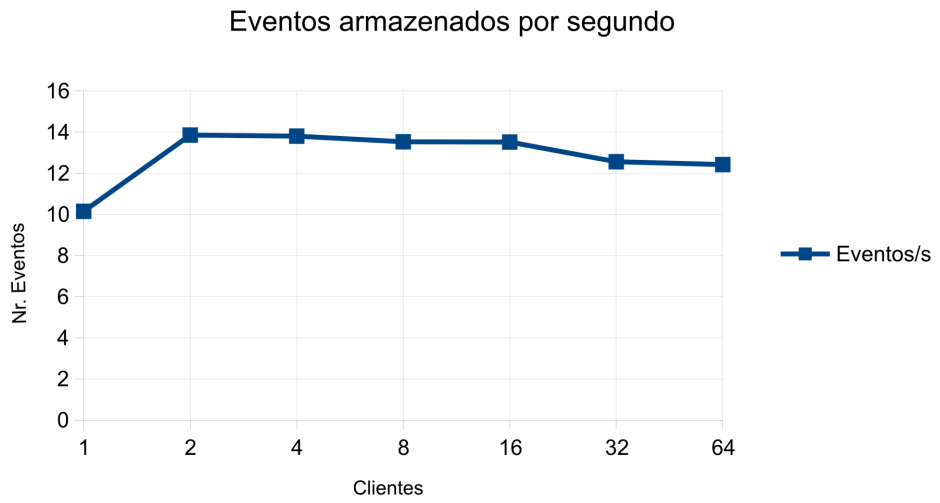

Figura 7.5: Eventos armazenados por segundo

O resultado mais baixo para só 1 cliente é facilmente explicado pelo facto de o sistema demorar 10 segundos a enviar os eventos e muito perto de 10 segundos para os armazenar, do que se pode concluir que com esse numero de clientes e eventos, o teste está longe de esgotar a capacidade do sistema.

Excluindo o resultado para 1 cliente, pela razão referida anteriormente, pode-se concluir que os requisitos não funcionais de performance e escalabilidade estão totalmente cumpridos já que que o sistema armazena sempre no mínimo 12 eventos por segundo, o que resulta em 720 eventos por minuto.

Tendo como base os valores dos gráficos do tempo de envio total dos eventos e do numero de eventos por segundo, pode-se ver que quando o numero de clientes cresce para 64, o tempo de envio aumenta visivelmente pela primeira vez, e que aos 32 clientes, o numero de eventos por segundo também diminui marcadamente. O primeiro é explicado pelo elevado numero de clientes começar a esgotar o tempo de processador disponível para o *Alert Receiver*, enquanto que o segundo resulta do número de eventos que o *Alert Manager* tenta armazenar simultaneamente, aproximadamente 32, se atrasarem mutuamente de uma forma mais marcada. Com base no anterior, situa-se o limite de escalabilidade, mantendo boa performance, em 16 clientes simultâneos neste teste.

No entanto, e tomando partido da arquitectura do SGA-IDS, a primeira limitação poderia ser evitada dividindo os clientes por duas máquinas e configurando um *Alert Receiver* por máquina, que reportariam os eventos para um *Alert Manager* central noutra máquina. A segunda limitação, constitui essencialmente uma limitação do SGBD e poderia ser diminuída por optimizações nas configurações do mesmo.

7.5 Instalação na Rede da Entidade Acolhedora

Depois de terminada a implementação e corrida a bateria de testes especificada anteriormente, a aplicação foi configurada numa máquina *OpenBSD* da entidade acolhedora e o sensor de rede *Snort* e o sensor local OSSEC foram configurados para enviar eventos para o sistema. A solução continua em uso dentro da empresa, integrada no sistema de monitorização da rede interna.

Capítulo 8

CONCLUSÃO E TRABALHO FUTURO

8.1 Conclusão

Ao longo do estágio o autor tentou dar resposta ao problema de armazenamento, consulta e gestão de eventos de segurança detectados numa rede de comunicação. Surge desse esforço, a implementação do SGA-IDS, um gestor de informação e eventos de segurança que permite armazenar eventos de todos os sensores de rede e locais mais utilizados, e realizar pesquisas sobre os mesmos.

Para o autor, este estágio permitiu-lhe trabalhar com algumas tecnologias que já conhecia superficialmente, mas que não faziam parte do seu percurso académico ou uso pessoal, e adicionalmente familiarizar-se com, e aprender, alguns conceitos na área da segurança informática.

Durante o estudo das alternativas existentes no mercado, constatou-se que nenhuma das soluções existentes estava de acordo com os objectivos apresentados para este projecto. Muitas limitavam o numero de sensores com que podiam comunicar, outros não eram satisfatórios em termos de escalabilidade, e ainda outros estavam quase abandonados pelo seu criador.

O dgs.SGA-IDS distingue-se das soluções já existentes, tentando reunir todas as características que a empresa ambicionava obter, já que permite comunicar com a quase totalidade dos sensores de rede e locais em uso no mercado, em simultâneo, e possui capacidade de escalar para grandes volumes de dados.

Relativamente à implementação, esta envolveu algum esforço de investigação já que o protocolo de comunicação utilizado carecia de documentação. No entanto, findo este processo de investigação foi produzida a documentação necessária, que só por si constitui um acrescento de valor na área que se insere este projecto.

Para a empresa, o trabalho do autor constituiu uma mais-valia já que permitiu a substituição imediata do sistema que era utilizado anteriormente, e que causava já muito trabalho manual de resolução de problemas causados por deficiências na sua arquitectura, com um custo de tempo demasiado alto em tarefas que deviam ser rotineiras e rápidas.

Resultado deste projecto, o dgs.SGA-IDS, a Dognædis pode agora garantir aos seus clientes, que os administradores das suas redes de comunicação estão continuamente informados dos eventos de segurança detectados na rede sobre a qual são responsáveis, fazendo assim da segurança informática um processo continuo e holístico, condição base para o seu sucesso.

8.2 Trabalho Futuro

Apesar da aplicação preencher todos os requisitos funcionais e não funcionais, com a excepção do requisito referente à funcionalidade ID-FU-008 (Obter a captura do tráfego), e ter passado quase imediatamente a ser utilizado na prática, é sempre possível melhorar a solução implementada.

Como continuação do trabalho o autor gostaria de implementar algumas melhorias e novas ideias, tais como:

- Integrar completamente a aplicação com o SGA, utilizando as APIs e módulos que ainda não estavam disponíveis ou estáveis aquando da implementação, como referido no Capítulo 6.
- Estudar o melhor modo de implementar a funcionalidade ID-FU-008 (Obter a captura de tráfego), e de seguida implementá-la. Esta funcionalidade pode constituir um grande factor de diferenciação da solução, já que permitiria a análise de eventos de segurança estando na posse de todo o contexto que lhe deu origem.
- Tornar o armazenamento dos eventos pelo *Alert Manager* assíncrono, de modo a não atrasar o envio por parte do *Alert Receiver* e aumentar ainda mais a escalabilidade da solução;
- Implementar hierarquias de *Alert Managers* de modo a que um *Alert Manager* pudesse, para além de armazenar o evento num SGBD, propagá-lo para outro *Alert Manager*. Deste modo poderia haver *Alert Managers* com visões segmentadas da situação da rede e outros com a visão total;
- Implementar um sistema de regras para classificação automática de eventos, acelerando assim o trabalho de triagem;
- Investigar a viabilidade de uso de XML *Databases* e *XQuery* uma vez que os eventos consistem de XML, e a utilização destas tecnologias permitiria realizar pesquisas sobre todos os elementos e atributos, e não unicamente sobre os atributos extraídos do XML e armazenados em tabelas do SGBD.

Referências Bibliográficas

[1] "Security information and event management," Jan. 2013. [Online]. Available: http://en.wikipedia.org/wiki/Security_information_and_event_management

[2] "ISO/IEC 27001:2005 Information technology — Security techniques — Information security management systems — Requirements," Aug. 2013. [Online]. Available: http://www.iso27001security.com/html/27001.html

[3] "ISO/IEC 27002:2005 Information technology — Security techniques — Code of practice for information security management," Aug. 2013. [Online]. Available: http://www.iso27001security.com/html/27002.html

[4] "ISO/IEC 27035:2011 Information technology — Security techniques — Information security incident management," Aug. 2013. [Online]. Available: http://www.iso27001security.com/html/27035.html

[5] "Computer Security Incident Response Team," Aug. 2013. [Online]. Available: http://www.csirt.org

[6] "Python Programming Language," Aug. 2013. [Online]. Available: http://python.org

[7] "Django - The Web framework for perfectionists with deadlines," Aug. 2013. [Online]. Available: https://www.djangoproject.com

[8] "Twisted," Aug. 2013. [Online]. Available: https://twistedmatrix.com/trac

[9] "lxml - Processing XML and HTML with Python," Aug. 2013. [Online]. Available: http://lxml.de

[10] "ReportLab: Content to PDF solutions," Aug. 2013. [Online]. Available: http://www.reportlab.com

[11] "PostgreSQL: The world's most advanced open source database," Aug. 2013. [Online]. Available: http://www.postgresql.org

[12] "Debian - The Universal Operating System ," Aug. 2013. [Online]. Available: http://www.debian.org

[13] "OpenBSD," Aug. 2013. [Online]. Available: http://www.openbsd.org

[14] "Python IDE & Django IDE for Web developers : JetBrains PyCharm," Aug. 2013. [Online]. Available: http://www.jetbrains.com/pycharm

[15] "Apache Subversion," Aug. 2013. [Online]. Available: https://subversion.apache.org

[16] "LyX – The Document Processor," Aug. 2013. [Online]. Available: http://www.lyx.org

[17] I. Sommerville, *Software Engineering 9*. Pearson Education, 2011. [Online]. Available: http://www.bibsonomy.org/bibtex/2853391dfed1142faf056c6247fd153ee/livany

[18] "Intrusion detection system," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Intrusion_detection_system

[19] "Sourcefire VRT (Vulnerability Research Team)," Aug. 2013. [Online]. Available: http://www.sourcefire.com/security-technologies/snort/vulnerability-research-team

[20] "Emerging Threats," Aug. 2013. [Online]. Available: http://www.emergingthreats.net

[21] "Snort," Aug. 2013. [Online]. Available: http://www.snort.org

[22] "GNU General Public License, version 2," Aug. 2013. [Online]. Available: https://www.gnu.org/licenses/gpl-2.0.html

[23] "TCPDUMP/LIBPCAP public repository ," Aug. 2013. [Online]. Available: http://www.tcpdump.org

[24] "Suricata," Aug. 2013. [Online]. Available: http://suricata-ids.org

[25] "OpenCL - The open standard for parallel programming of heterogeneous systems," Aug. 2013. [Online]. Available: http://www.khronos.org/opencl

[26] "CUDA Parallel Computing Platform," Aug. 2013. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[27] "The Bro Network Security Monitor," Aug. 2013. [Online]. Available: http://www.bro.org

[28] "BSD licenses," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/BSD_licenses

[29] "The programming language Lua," Aug. 2013. [Online]. Available: http://www.lua.org

[30] "Host-based intrusion detection system," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Host-based_intrusion_detection_system

[31] "Logfile," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Logfile

[32] "Cryptographic hash function," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function

[33] "Rootkit," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Rootkit

[34] "OSSEC | Home | Open Source SECurity," Aug. 2013. [Online]. Available: http://www.ossec.net

[35] "Samhain Labs | samhain," Aug. 2013. [Online]. Available: http://la-samhna.de/samhain

[36] "Prelude-LML," Aug. 2013. [Online]. Available: https://www.prelude-ids.org/wiki/1/PreludeLml

[37] "Regular expression," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Regular_expression

[38] "The Sagan Log Analysis Engine," Aug. 2013. [Online]. Available: http://sagan.quadrantsec.com

[39] “AIDE - Advanced Intrusion Detection Environment ,” Aug. 2013. [Online]. Available: http://aide.sourceforge.net

[40] “The GNU General Public License v3.0 - GNU Project - Free Software Foundation (FSF),” Aug. 2013. [Online]. Available: https://www.gnu.org/licenses/gpl.html

[41] “Syslog,” Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Syslog

[42] “The Intrusion Detection Message Exchange Format (IDMEF),” Aug. 2013. [Online]. Available: https://tools.ietf.org/html/rfc4765

[43] “Wireshark - Go Deep.” Aug. 2013. [Online]. Available: https://www.wireshark.org

[44] “ELSA,” Aug. 2013. [Online]. Available: http://code.google.com/p/enterprise-log-search-and-archive

[45] “Reliable log management - syslog-ng Open Source Edition,” Aug. 2013. [Online]. Available: http://www.balabit.com/network-security/syslog-ng/opensource-logging-system

[46] “Sphinx | Open Source Search Server ,” Aug. 2013. [Online]. Available: http://sphinxsearch.com

[47] “Sguil - Open Source Network Security Monitoring,” Aug. 2013. [Online]. Available: http://sguil.sourceforge.net

[48] “Tcl SourceForge Project,” Aug. 2013. [Online]. Available: http://tcl.sourceforge.net

[49] “The Squert project,” Aug. 2013. [Online]. Available: http://www.squertproject.org

[50] “ACARM | Main / home,” Aug. 2013. [Online]. Available: http://www.acarm.wcss.wroc.pl

[51] “Prelude-IDS - WikiStart - Prelude Tracker ,” Aug. 2013. [Online]. Available: https://www.prelude-ids.org

[52] “The Internet Engineering Task Force (IETF),” Aug. 2013. [Online]. Available: http://www.ietf.org

[53] “CS fr,” Aug. 2013. [Online]. Available: http://www.c-s.fr

[54] “Snorby - All about simplicity,” Aug. 2013. [Online]. Available: https://snorby.org

[55] “Ruby Programming Language,” Aug. 2013. [Online]. Available: http://www.ruby-lang.org/en

[56] “Aanval - Snort, Suricata and Syslog Intrusion Detection, Correlation and Threat Management ,” Aug. 2013. [Online]. Available: https://www.aanval.com

[57] “Exploit (computer security),” Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Exploit_(computer_security

[58] “RTIR: RT for Incident Response,” Aug. 2013. [Online]. Available: http://bestpractical.com/rtir

[59] R. Meyer, “Challenges of Managing an Intrusion Detection System (IDS) in the Enterprise ,” *SANS Institute Reading Room*, Mar. 2008.

[60] S. G. Alves, “Security Gateway Agent - documentação,” Tech. Rep., 2010.

[61] "OWASP Application Security Verification Standard Project," Aug. 2013. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

[62] The OpenSSL Project, "OpenSSL: The Open Source toolkit for SSL/TLS," April 2003, www.openssl.org.

[63] O. S. S. Institute, *OpenSSL FIPS 140-2 UserGuide* , Jun. 2012.

[64] "The GnuTLS Transport Layer Security Library," Aug. 2013. [Online]. Available: http://www.gnutls.org

[65] A. S. P. Ltd, "MySQL vs. postgreSQL comparison." Aug. 2013. [Online]. Available: http://www.anchor.com.au

[66] "MySQL vs. PostgreSQL benchmarks," Aug. 2013. [Online]. Available: http://www.randombugs.com/linux/mysql-postgresql-benchmarks.html

[67] "MySQL vs PostgreSQL," Aug. 2013. [Online]. Available: http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL

[68] "PythonSoftwareFoundationLicenseFaq," Aug. 2013. [Online]. Available: https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq

[69] "Python Software Foundation," Aug. 2013. [Online]. Available: http://www.python.org/psf

[70] "Best web frameworks - usage," Aug. 2013. [Online]. Available: http://www.bestwebframeworks.com/web-frameworks-usage/#python

[71] "Berkeley Software Distribution," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Berkeley_Software_Distribution

[72] "OpenBSD Project Goals," Aug. 2013. [Online]. Available: http://www.openbsd.org/goals.html

[73] "Unified Modeling Language," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Unified_Modeling_Language

[74] "DataTables (table plug-in for jQuery) ," Aug. 2013. [Online]. Available: http://datatables.net

[75] "Bourne shell," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Bourne_shell

[76] IETF, "The Intrusion Detection Message Exchange Format (IDMEF)," Jul. 2013. [Online]. Available: http://tools.ietf.org/html/rfc4765

[77] B. Shneiderman and C. Plaisant, *Designing the User Interface - Strategies for Effective Human-Computer Interaction (5. ed.).* Addison-Wesley, 2010.

[78] F. P. 140-2, "Security Requirements for Cryptographic Modules," 2002.

[79] "UNIX time," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Unix_time

[80] "Perfect Forward Secrecy," Aug. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Perfect_forward_secrecy

Apêndices

Apêndice A

OWASP ASVS

O presente apêndice apresenta aquelas que são as 14 métricas apresentadas pelo documento ASVS do OWASP, que representam um conjunto standards reservados para garantir a segurança da informação em aplicações Web, revelando aquelas que são as boas práticas e preocupações a ter, quando desenvolvidos estes tipos de sistemas. Detalhes relativos a cada um dos testes de segurança enumerados e formas de validação acerca da sua implementação, podem ser verificados na bibliografia supracitada.

- V1. Arquitectura segura: Documentação e definição da arquitectura básica de segurança da aplicação para que se garanta e integridade e precisão de verificações de segurança sobre esta. Identificação de todas as bibliotecas a utilizar na aplicação e componentes exteriores a esta, verificar que uma arquitectura de alto nível para a aplicação foi definida, entre outros.
- V2. Autenticação: Gerar e manipular as credenciais de uma conta de utilizador de forma segura.
- V3. Gestão de sessões: Utilizar de forma segura HTTP *Requests*, *Responses*, *sessions*, *cookies*, *headers* e *logging*, de forma a que a gestão de sessões seja feita de forma segura.
- V4. Controlo de acessos: Definir mecanismos de controle de acesso. Deverá ser feito em várias camadas da aplicação.
- V5. Validação de *Inputs*: Validar inputs introduzidos pelo utilizador de forma a que se processe de forma segura para a aplicação e para cliente.
- V6. *Encoding*/*Escaping* de Outputs: validar outputs introduzidos pelo utilizador de forma a que se processem de forma segura para a aplicação e para cliente.
- V7. Criptografia: Verificar as condições de encriptação de uma aplicação, gestão de chaves, gestão de números aleatórios, ou operações de *hashing*. Validar os módulos criptográficos usados pela aplicação por FIPS 140-2 (Federal Information Processing Standard)[78].
- V8. *Logging* e tratamento de erros: Verificar e monitorizar os eventos de segurança da aplicação, levando à identificação de comportamentos da mesma que possam significar ataques a esta.
- V9. Proteção de dados: Verificar a protecção de dados sensíveis da aplicação (número de identificação, eventos de um sistema no caso de um SIEM, entre outros).
- V10. Segurança na comunicação: Verificar que todas as comunicações entre módulos da aplicação ou com aplicações externas se encontram apropriadamente seguras.

- V11. Segurança HTTP: Verificar a segurança relacionada com os HTTP *requests*, *responses*, *sessions*, *cookies*, *headers* e *logging*.
- V12. Configurações de segurança: Verificar o armazenamento seguro de todas as informações sobre as configurações da aplicação, que definem o comportamento desta relacionado com a sua segurança.
- V13. Procura de código Malicioso: Definir procura de software malicioso que possa estar presente no código desenvolvido ou modificado para criação da aplicação. Deverá também ser verificada a integridade das bibliotecas, ficheiros de configuração ou código utilizado pela aplicação.
- V14. Segurança Interna: Verificar se a aplicação se defende a ela mesma contra falhas de implementação. Por exemplo a verificação de que a aplicação protege correctamente variáveis e recursos que sejam utilizados de forma concorrente.

Apêndice B

TEMPLATE DO SGA

Neste apêndice inclui-se uma *screenshot* da *template* fornecida pelo SGA, de modo a que a interface seja consistente entre os vários módulos.

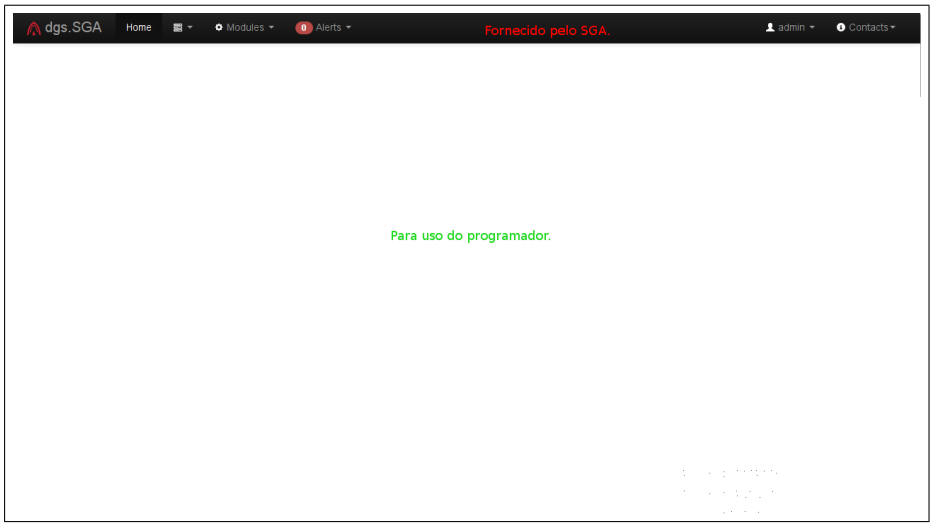

Figura B.1: Template SGA

Apêndice C

SCREENSHOTS DA APLICAÇÃO

Neste apêndice apresentam-se algumas *screenshots* da interface com o utilizador da aplicação, para que se possa ter uma ideia geral da mesma. Apresenta-se também uma *screenshot* de um relatório criado pela aplicação.

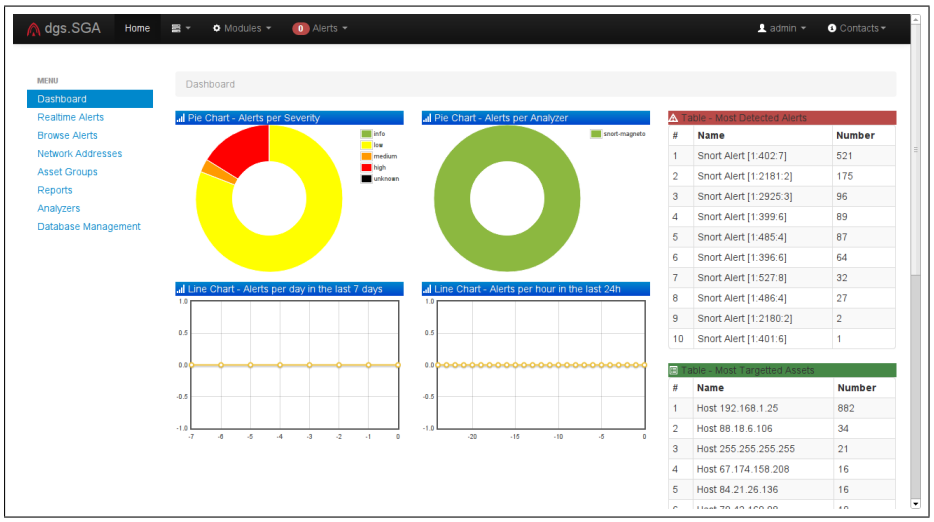

Figura C.1: *Screenshot* do *Dashboard*

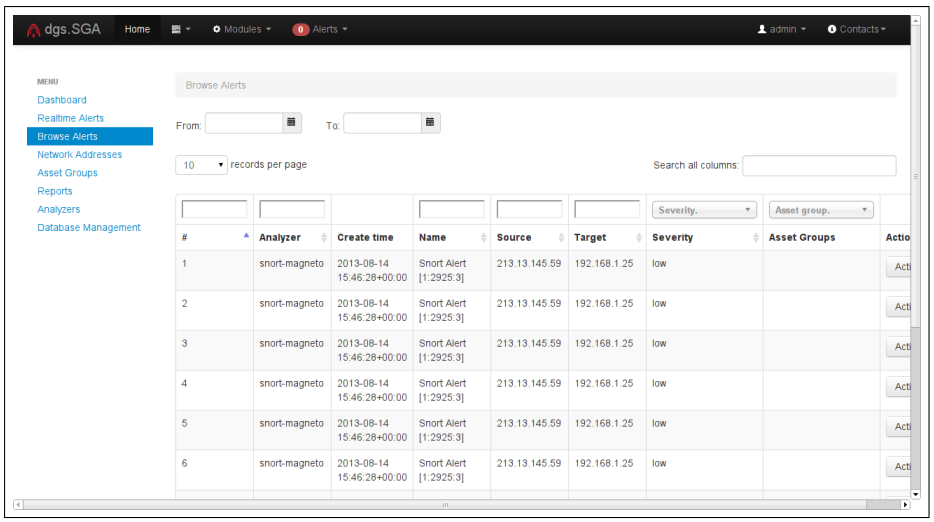

Figura C.2: *Screenshot* do histórico de alertas

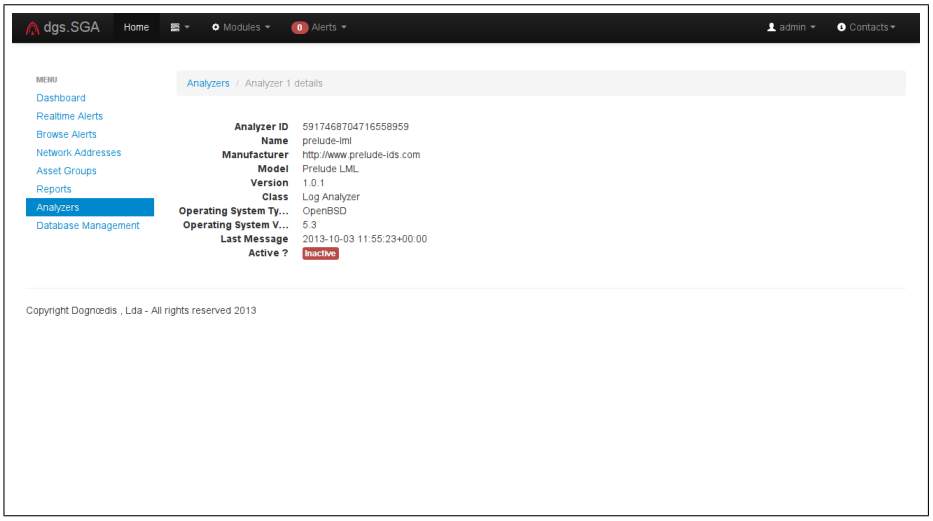

Figura C.3: *Screenshot* das informações sobre o sensor

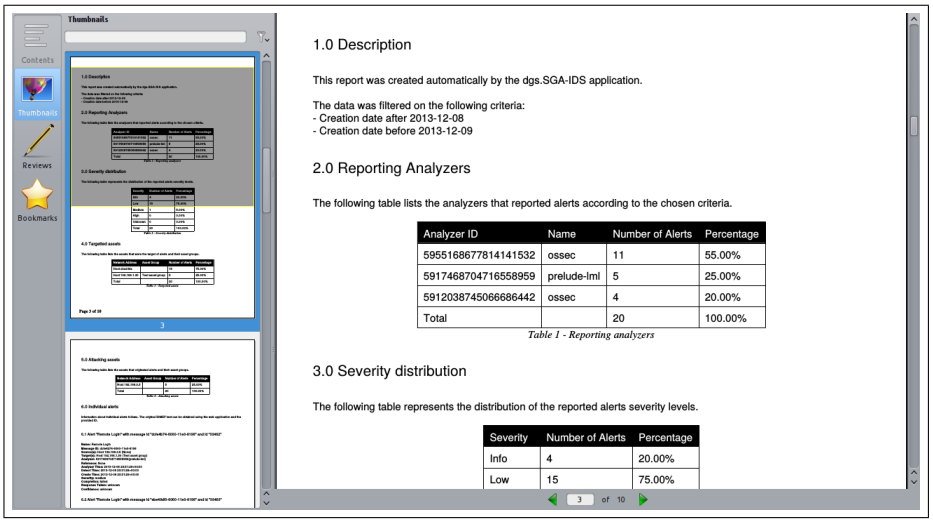

Figura C.4: *Screenshot* de um relatório

Apêndice D

CASOS DE USO DETALHADOS

Neste apêndice inclui-se a especificação detalhada de todos os casos de uso implementados pela aplicação e uma matriz de rastreio que permite saber rapidamente que caso de uso implementa um determinado requisito funcional, e vice versa.

D.1 ID-UC-001: Consultar dashboard

Nome	Consultar *dashboard*
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador escolhe o menu "*Dashboard*" e visualiza as tabelas: • 10 eventos mais detectados; • 10 activos mais atacados; • 10 activos que mais atacam. E os gráficos: • Distribuição de severidade dos eventos recebidos; • Distribuição de sensores que enviaram eventos; • Eventos recebidos nas ultimas 24 horas; • Eventos recebidos nos últimos 7 dias.
Implementa	ID-RF-004

Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para consultar o *dashboard*.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolhe o menu "Dashboard". 1.2: O sistema apresenta a página do *dashboard* com as tabelas e gráficos descritos.
Fluxos alternativos	Nenhum.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.2 ID-UC-002: Ver eventos em tempo real

Nome	Ver eventos em tempo real
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador escolhe o menu "*Realtime alerts*" e visualiza os últimos 10 eventos recebidos ordenados pela ordem de recepção. O utilizador pode também: • Mudar o número de eventos que vê (10, 25, 50 ou 100); • Filtrar os eventos por sensor, nome, origem, destino e grupo de activos. • Ligar e desligar a actualização automática da lista. • Ver os detalhes do evento. • Criar um *ticket* no RTIR com a informação do evento.
Implementa	ID-RF-005, ID-RF-006, ID-RF-007

Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para ver eventos em tempo real.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolhe o menu "*Realtime alerts*". 1.2: O sistema apresenta a caixa de selecção do numero de eventos a listar em simultâneo. 1.3: O sistema apresenta o botão que desliga e liga a actualização automática. 1.4: O sistema apresenta as caixas de texto para filtrar por sensor, nome, origem e destino. 1.5: O sistema apresenta a caixa de selecção para filtrar por grupo de activos. 1.6: O sistema apresenta os últimos eventos recebidos filtrados pelos valores das caixas de texto e de selecção. 1.7: O sistema apresenta, para cada um dos eventos, e se o utilizador possuir permissão para isso, um botão para ver todos os detalhes do evento. 1.8: O sistema apresenta, para cada um dos eventos, e se o utilizador possuir permissão para isso, um botão para criar um *ticket* no RTIR com os detalhes do evento. 1.9: O sistema, se a actualização automática está ligada, espera 5 segundos e de seguida volta para o passo 1.6.

Fluxos alternativos	A1: A seguir a 1.8 A1.1: O utilizador entra um valor numa das caixas de texto ou escolhe uma opção na caixa de selecção. A1.2: Volta para o passo 1.6 do fluxo principal. A2: A seguir a 1.8 A2.1: O utilizador carrega no botão que alterna a actualização automática. A2.2: O sistema desliga ou liga a actualização automática conforme o valor actual. A2.3: O sistema volta para o passo 1.8 do fluxo principal. A3: A seguir a 1.8 A3.1: O utilizador muda o numero de eventos a ver simultaneamente. A3.2: Volta para o passo 1.8 do fluxo principal. A4: A seguir a 1.8 A4.1: O utilizador carrega no botão para ver detalhes de um evento. A4.2: O sistema invoca o caso de uso "ID-UC-003: Ver detalhes de evento". A5: A seguir a 1.8 A5.1: O utilizador carrega no botão para criar um *ticket* no RTIR com os detalhes de um evento. A5.2: O sistema invoca o caso de uso "ID-UC-004: Criar ticket".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.3 ID-UC-003: Ver detalhes de evento

Nome	Ver detalhes do evento
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode ver todos os detalhes armazenados acerca do evento. Pode também ver o XML original do evento.
Implementa	

Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para ver detalhes de eventos.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolheu a opção para ver os detalhes de um alerta. 1.2: O sistema apresenta a seguinte informação sobre o evento: • Nome; • "Message ID" do XML original do evento; • Endereços de origem, se existirem; • Endereços de destino, se existirem; • Sensor que enviou o evento; • Referencia do evento, se existir; • Data e tempo de envio do evento pelo sensor; • Data e tempo da detecção do evento; • Data e tempo da criação do evento; • Severidade do evento; • Grau de sucesso do evento, se existente; • Resposta dada ao evento, se existente; • Confiança na detecção, se existente. 1.3: O sistema apresenta um botão, se o utilizador possuir a permissão para isso, que permite consultar o XML original do evento.
Fluxos alternativos	A1: A seguir a 1.3 A1.1: O utilizador clica no botão para ver o XML original do evento. A1.2: O sistema apresenta o XML original do evento.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.4 ID-UC-004: Criar ticket

Nome	Criar *ticket*
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode criar um *ticket* no sistema RTIR com a informação do evento e do sensor que o enviou.
Implementa	ID-RF-012
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para criar *tickets*.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolheu a opção para criar um *ticket* com a informação do evento. 1.2: O sistema apresenta uma caixa de texto para o assunto do *ticket*. 1.3: O sistema apresenta uma caixa de texto com a informação do evento e sensor que o enviou já adicionada. 1.4: O sistema apresenta um botão para o envio do *ticket*. 1.5: O utilizador entra um assunto e, opcionalmente, modifica a informação sobre o ticket. 1.6: O utilizador clica no botão para enviar. 1.7: O sistema tenta criar o *ticket*. 1.8: O sistema apresenta uma mensagem de sucesso.
Fluxos alternativos	A1: A seguir a 1.6 (Falta o assunto) A1.1: O sistema mostra uma mensagem de erro. A2: A seguir a 1.7 (A criação do ticket falha) A2.1: O sistema mostra uma mensagem de erro.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.5 ID-UC-005: Consultar eventos

Nome	Consultar eventos

Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador escolhe o menu "*Browse alerts*" e pode consultar todos os eventos recebidos pelo sistema. O utilizador pode também: • Mudar o número de eventos que vê de uma vez (10, 25, 50 ou 100); • Filtrar os eventos por numero, sensor, nome, origem, destino, severidade, grupo de activos e todos os critérios simultaneamente. • Filtrar os eventos por um intervalo temporal com a granularidade de dia. • Ver os detalhes do evento. • Criar um *ticket* no RTIR com a informação do evento.
Implementa	ID-RF-008, ID-RF-009
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para consultar eventos.
Pós-condições	Não aplicável.

Fluxo principal	1.1: O utilizador escolhe o menu "*Browse alerts*". 1.2: O sistema apresenta a caixa de selecção do numero de eventos a listar em simultâneo. 1.3: O sistema apresenta a caixa de texto para procura em todos os critérios. 1.4: O sistema apresenta as caixas de selecção de intervalo de tempo. 1.5: O sistema apresenta as caixas de texto para filtrar por numero,sensor, nome, origem e destino. 1.6: O sistema apresenta as caixas de selecção para filtrar por severidade e grupo de activos. 1.7: O sistema apresenta os eventos recebidos filtrados pelos valores das caixas de texto e de selecção. 1.8: O sistema apresenta, para cada um dos eventos, e se o utilizador possuir permissão para isso, um botão para ver todos os detalhes do evento. 1.9: O sistema apresenta, para cada um dos eventos, e se o utilizador possuir permissão para isso, um botão para criar um *ticket* no RTIR com os detalhes do evento.
Fluxos alternativos	A1: A seguir a 1.9 A1.1: O utilizador entra um valor numa das caixas de texto ou escolhe uma opção nas caixas de selecção. A1.2: Volta para o passo 1.7 do fluxo principal. A2: A seguir a 1.9 A2.1: O utilizador selecciona uma data mínima e/ou uma data máxima nas caixas de selecção do intervalo de tempo. A2.2: Volta para o passo 1.7 do fluxo principal. A3: A seguir a 1.9 A3.1: O utilizador carrega no botão para ver detalhes de um evento. A3.2: O sistema invoca o caso de uso "ID-UC-003: Ver detalhes de evento". A4: A seguir a 1.9 A4.1: O utilizador carrega no botão para criar um *ticket* no RTIR com os detalhes de um evento. A4.2: O sistema invoca o caso de uso "ID-UC-004: Criar ticket".

Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.6 ID-UC-006: Gerir endereços de rede

Nome	Gerir endereços de rede
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode consultar todos os endereços de rede do sistema. O utilizador pode também: • Mudar o número de endereços que vê de uma vez (10, 25, 50 ou 100); • Filtrar os eventos por numero, sensor, nome, origem, destino, severidade, grupo de activos e todos os critérios simultaneamente. • Ver os detalhes do endereço. • Apagar o endereço e os eventos com origem ou destino nele.
Implementa	ID-RF-013
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para consultar endereços.
Pós-condições	Não aplicável.

Fluxo principal	1.1: O utilizador escolhe o menu "*Network Addresses*". 1.2: O sistema apresenta a caixa de selecção do numero de endereços a listar em simultâneo. 1.3: O sistema apresenta a caixa de texto para procura em todos os critérios. 1.4: O sistema apresenta as caixas de texto para filtrar por numero e nome. 1.5: O sistema apresenta a caixa de selecção para filtrar por grupo de activos. 1.6: O sistema apresenta os endereços filtrados pelos valores das caixas de texto e de selecção. 1.7: O sistema apresenta, para cada um dos endereços, e se o utilizador possuir permissão para isso, um botão para ver todos os detalhes do mesmo. 1.8: O sistema apresenta, para cada um dos endereços, e se o utilizador possuir permissão para isso, um botão para apagar o endereço e todos os eventos com origem ou destino nele.
Fluxos alternativos	A1: A seguir a 1.8 A1.1: O utilizador entra um valor numa das caixas de texto ou escolhe uma opção na caixa de selecção. A1.2: Volta para o passo 1.6 do fluxo principal. A2: A seguir a 1.8 A2.1: O utilizador carrega no botão para ver detalhes de um endereço. A2.2: O sistema invoca o caso de uso "ID-UC-007: Ver detalhes de endereço". A3: A seguir a 1.8 A3.1: O utilizador carrega no botão para apagar o endereço e os alertas relativos a ele. A3.2: O sistema invoca o caso de uso "ID-UC-008: Apagar endereço".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.7 ID-UC-007: Ver detalhe de endereço

Nome	Ver detalhe de endereço
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)

Actor	Utilizador
Descrição	O utilizador consulta todos os detalhes sobre um endereço. Pode também apagar o endereço e todos os eventos com origem ou destino nele.
Implementa	ID-RF-013
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para ver detalhes de endereços.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolheu a opção para ver os detalhes de um endereço. 1.2: O sistema apresenta os seguintes detalhes sobre o endereço: • Nome; • Fonte do nome (DNS, WINS, etc); • Tipo de endereço; • Nome da máquina; • Endereço IP; • Mascara de rede, se existente; • Se o endereço é uma rede; • Grupo de activos do endereço, se existente. 1.3: O sistema apresenta um botão para apagar o endereço e os eventos com origem ou destino nele.
Fluxos alternativos	A1: A seguir a 1.3 A1.1: O utilizador clica o botão para apagar o endereço e os eventos relativos a ele. A1.2: O sistema invoca o caso de uso "ID-UC-008: Apagar endereço".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.8 ID-UC-008: Apagar endereço

Nome	Apagar endereço
Criado por	Ricardo Ferreira (26/08/2013)

Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode apagar um endereço e todos os eventos com origem ou destino neste.
Implementa	ID-RF-013
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para apagar endereços.
Pós-condições	O endereço e todos os alertas com origem ou destino neste são apagados.
Fluxo principal	1.1: O utilizador escolhe apagar o endereço e os eventos relativos a ele. 1.2: O sistema apresenta um pedido de confirmação. 1.3: O utilizador confirma. 1.4: O sistema apaga os eventos com origem ou destino no endereço e de seguida o endereço em si. 1.5: O sistema apresenta uma mensagem de sucesso.
Fluxos alternativos	A1: A seguir a 1.2 A1.1: O utilizador não confirma. A1.2: O sistema cancela a operação.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor. E2: Impossível efectuar a operação. E2.1: O sistema apresenta uma mensagem de erro com a informação.

D.9 ID-UC-009: Gerir grupos de activos

Nome	Gerir grupos de activos
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode consultar os grupos de activos definidos. Pode também criar, editar ou apagar um grupo de activos.
Implementa	ID-RF-017, ID-RF-018, ID-RF-019

Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para gerir grupos de activos.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolhe o menu "*Asset groups*". 1.2: O sistema apresenta o botão para criar um novo grupo de activos. 1.3: O sistema apresenta a lista de grupos de activos. 1.4: O sistema apresenta, para cada grupo de activos, e se o utilizador tiver a permissão para isso, um botão para ver detalhes sobre o grupo. 1.5: O sistema apresenta, para cada grupo de activos, e se o utilizador tiver a permissão para isso, um botão para editar o grupo. 1.6: O sistema apresenta, para cada grupo de activos, e se o utilizador tiver a permissão para isso, um botão para apagar o grupo.
Fluxos alternativos	A1: A seguir a 1.6 A1.1: O utilizador clica no botão para ver detalhes do grupo de activos. A1.2: O sistema invoca o caso de uso "ID-UC-011: Ver detalhe de grupo de activos". A1: A seguir a 1.6 A1.1: O utilizador clica no botão para editar o grupo de activos. A1.2: O sistema invoca o caso de uso "ID-UC-013: Editar grupo de activos". A1: A seguir a 1.6 A1.1: O utilizador clica no botão para apagar o grupo de activos. A1.2: O sistema invoca o caso de uso "ID-UC-012: Apagar grupo de activos". A1: A seguir a 1.6 A1.1: O utilizador clica no botão adicionar um grupo de activos. A1.2: O sistema invoca o caso de uso "ID-UC-010: Adicionar grupo de activos".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.10 ID-UC-010: Adicionar grupo de activos

Nome	Adicionar grupo de activos
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode criar um grupo de activos definindo o seu nome e posteriormente editando o mesmo para adicionar activos.
Implementa	ID-RF-017
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para adicionar grupos de activos.
Pós-condições	O grupo de activo foi criado na base de dados.
Fluxo principal	1.1: O utilizador escolheu criar um grupo de activos. 1.2: O sistema apresenta uma caixa de texto para o nome do grupo de activos. 1.3: O sistema apresenta um botão para criar o grupo de activos. 1.4: O utilizador introduz um nome na caixa de texto. 1.5: O utilizador clica no botão para criar o grupo de activos. 1.6: O sistema cria o grupo de activos na base de dados sem nenhum endereço membro. 1.7: O sistema invoca o caso de uso "ID-UC-013: Editar grupo de activos".
Fluxos alternativos	A1: A seguir a 1.3 (Nome não introduzido) A1.1: O utilizador clica no botão para criar o grupo de activos. A1.2: O sistema apresenta uma mensagem com a informação da obrigatoriedade de introduzir um nome. A1.3: Volta para o passo 1.1 do fluxo principal.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor. E2: Impossível efectuar a operação. E2.1: O sistema apresenta uma mensagem de erro com a informação.

D.11 ID-UC-011: Ver detalhe de grupo de activos

Nome	Ver detalhe de grupo de activos
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode consultar todos os detalhes do grupo de activos: • Nome • Uma lista de todos os activos que lhe pertencem. O utilizador pode também editar o grupo de activos.
Implementa	
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para consultar detalhes de grupos de activos.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolheu ver os detalhes de um grupo de activos. 1.2: O sistema apresenta o nome do grupo de activos e uma lista dos nomes de activos que fazem parte do grupo de activos. 1.3: O sistema apresenta o botão para editar o grupo de activos.
Fluxos alternativos	A1: A seguir a 1.3 A1.1: O utilizador clica no botão para editar o grupo de activos. A1.2: O sistema invoca o caso de uso "ID-UC-013: Editar grupo de activos".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.12 ID-UC-012: Apagar grupo de activos

Nome	Apagar grupo de activos
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode apagar um grupo de activos.

Implementa	ID-RF-017
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para apagar grupos de activos.
Pós-condições	O grupo de activos é apagado da base de dados e os seus activos são retirados dele.
Fluxo principal	1.1: O utilizador escolheu apagar um grupo de activos. 1.2: O sistema pede confirmação. 1.4: O utilizador confirma. 1.5: O sistema apaga o grupo de activos da base de dados. 1.6: O sistema apresenta uma mensagem de sucesso.
Fluxos alternativos	A1.1: A seguir a 1.2 (O utilizador cancela) A1.1: O utilizador não confirma. A1.2: O sistema invoca o caso de uso "ID-UC-009: Gerir grupos de activos".
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.13 ID-UC-013: Editar grupo de activos

Nome	Editar grupo de activos
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode editar um grupo de activos. Pode mudar o nome e/ou adicionar e remover activos.
Implementa	ID-RF-018, ID-RF-019
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para apagar grupos de activos.
Pós-condições	O grupo de activos é modificado na base de dados.

Fluxo principal	1.1: O utilizador escolheu editar um grupo de activos. 1.2: O sistema apresenta uma caixa de texto com o nome actual do grupo de activos. 1.3: O sistema apresenta uma caixa de selecção múltipla com todos os activos do sistema e com os activos que já são membros do grupo pré-seleccionados. 1.4: O sistema apresenta um botão para gravar as alterações. 1.5: O utilizador modifica o nome do grupo ou os activos seleccionados na caixa de selecção múltipla. 1.6: O utilizador clica no botão para gravar as alterações. 1.7: O sistema grava as alterações na base de dados. 1.8: O sistema apresenta uma mensagem de sucesso.
Fluxos alternativos	A1: A seguir a 1.4 (Nome vazio) A1.1: O utilizador introduz um nome vazio na caixa de texto. A1.2: O utilizador clica no botão para gravar as alterações. A1.3: O sistema apresenta uma mensagem a informar que um nome não vazio é obrigatório.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.14 ID-UC-014: Gerir relatórios

Nome	Gerir relatórios
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode gerar relatórios relativos a um intervalo de tempo e um grupo de activos. O utilizador pode descarregar relatórios gerados previamente.
Implementa	ID-RF-010, ID-RF-011

Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para gerar relatórios.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolhe o menu "*Reports*". 1.2: O sistema apresenta um selector de data mínima e outro de data máxima. 1.4: O sistema apresenta uma caixa de selecção do grupo de activos sobre o qual será gerado o relatório. 1.5: O sistema apresenta um botão para iniciar a geração do relatório. 1.6: O sistema, se existirem relatórios gerados anteriormente, apresenta uma caixa de selecção com os mesmos e um botão para descarregar o relatório seleccionado na caixa de selecção. 1.6: O utilizador introduz uma data mínima, uma data máxima e um grupo de activos. 1.7: O utilizador clica no botão de geração de relatório. 1.8: O sistema apresenta uma mensagem a informar que a operação pode ser demorada. 1.9: O sistema gera o relatório para o intervalo de tempo e grupo de activos introduzidos.
Fluxos alternativos	A1: A seguir a 1.6 A1.1: O utilizador escolhe um relatório gerado anteriormente na caixa de selecção. A1.2: O utilizador clica no botão para descarregar esse relatório. A1.3: O sistema envia o relatório para ser descarregado pelo utilizador.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor. E2: Erro na geração do relatório. E2.1: O sistema apresenta uma mensagem de erro a informar do facto.

D.15 ID-UC-015: Consultar sensores

Nome	Consultar sensores

Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador
Descrição	O utilizador pode consultar os sensores que já comunicaram com o sistema. Pode também consultar todos os detalhes de um sensor.
Implementa	
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para gerar relatórios.
Pós-condições	Não aplicável.
Fluxo principal	1.1: O utilizador escolhe o menu "*Analyzers*". 1.2: O sistema apresenta uma lista com os sensores que já comunicaram com o sistema. 1.3: O sistema apresenta, se o utilizador possuir permissão para isso, um botão para consultar todos os detalhes de cada sensor.
Fluxos alternativos	A1: A seguir a 1.3 A1.1: O utilizador clica no botão para ver os detalhes de um sensor. A1.2: O sistema apresenta todos os detalhes na base de dados sobre esse sensor.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.16 ID-UC-016: Gerir base de dados

Nome	Gerir base de dados
Criado por	Ricardo Ferreira (26/08/2013)
Actualizado por	Ricardo Ferreira (26/08/2013)
Actor	Utilizador

Descrição	O utilizador pode: • Eliminar todos os eventos com severidade igual à especificada; • Eliminar todos os eventos com uma severidade menor ou igual à especificada; • Criar uma cópia de segurança da base de dados e esvaziar a base dados actual. • Descarregar uma cópia de segurança gerada anteriormente.
Implementa	ID-RF-014, ID-RF-015
Pré-condições	• O utilizador está autenticado na SGA. • O utilizador possui a permissão para gerir a base de dados.
Pós-condições	As alterações são efectuadas na base de dados.

Fluxo principal	1.1: O utilizador escolhe o menu "*Database management*". 1.2: O sistema apresenta uma caixa de selecção para a severidade. 1.3: O sistema apresenta um botão para apagar todos os eventos com severidade igual à seleccionada. 1.4: O sistema apresenta um botão para apagar todos os eventos com severidade menor ou igual à seleccionada. 1.5: O sistema apresenta um botão para criar uma cópia de segurança e de seguida esvaziar a base de dados. 1.6: O sistema apresenta, se existirem cópias de segurança geradas anteriormente, uma caixa de selecção para as mesmas e um botão para descarregar a cópia seleccionada. 1.7: O utilizador clica no botão para criar um cópia de segurança e esvaziar a base de dados. 1.8: O sistema pede confirmação. 1.9: O utilizador confirma. 1.20: O sistema apresenta uma mensagem a informar que a operação pode demorar. 1.21: O sistema cria uma cópia de segurança e esvazia a base de dados. 1.22: O sistema apresenta uma mensagem de sucesso.

Fluxos alternativos	A1: A seguir a 1.8 (O utilizador cancela) A1.1: O utilizador cancela. A1.2: O sistema volta para o passo 1.2 do fluxo principal. A2: A seguir a 1.6 A2.1: O utilizador clica no botão para apagar os eventos com severidade igual à seleccionada. A2.2: O sistema pede confirmação. A2.3: O utilizador confirma. A2.4: O sistema apaga os eventos que satisfazem o critério. A2.5: Volta para o passo 1.2 do fluxo principal. A3: A seguir a 1.6 A3.1: O utilizador clica no botão para apagar os eventos com severidade menor ou igual à seleccionada. A3.2: O sistema pede confirmação. A3.3: O utilizador confirma. A3.4: O sistema apaga os eventos que satisfazem o critério. A3.5: Volta para o passo 1.2 do fluxo principal. A4: A seguir a 1.6 A4.1: O utilizador clica no botão para descarregar uma cópia de segurança gerada anteriormente. A4.2: O sistema envia o ficheiro da cópia de segurança para descarregar. A4.5: Volta para o passo 1.2 do fluxo principal.
Excepções	E1: Erro no acesso à base de dados. E1.1: O sistema apresenta a página de erro no servidor.

D.17 Matriz de Rastreio

Na Tabela D.17 apresenta-se a matriz de rastreio que permite fazer a correspondência entre cada caso de uso e os requisitos funcionais que implementa.

Casos de uso	Requisitos funcionais																		
	ID-RF-001	ID-RF-002	ID-RF-003	ID-RF-004	ID-RF-005	ID-RF-006	ID-RF-007	ID-RF-008	ID-RF-009	ID-RF-010	ID-RF-011	ID-RF-012	ID-RF-013	ID-RF-014	ID-RF-015	ID-RF-016	ID-RF-017	ID-RF-018	ID-RF-019
ID-UC-001				✓															
ID-UC-002					✓	✓	✓												
ID-UC-003																			
ID-UC-004												✓							
ID-UC-005								✓	✓										
ID-UC-006													✓						
ID-UC-007													✓						
ID-UC-008													✓						
ID-UC-009																	✓	✓	✓
ID-UC-010																	✓		
ID-UC-011																			
ID-UC-012																	✓		
ID-UC-013																		✓	✓
ID-UC-014										✓	✓								
ID-UC-015																			
ID-UC-016															✓	✓			

Tabela D.17: Matriz de rastreio entre casos de uso e requisitos funcionais

Apêndice E

EXEMPLOS DE CÓDIGO

E.1 Sistema de RPC

O módulo *Quyckro* foi criado pela Dognædis para a nova versão do SGA, e é utilizado para implementar o sistema de RPC que as aplicações web da *Interface Layer* utilizam na comunicação com o módulo da *Manager Abstraction Layer* que implementa a lógica de negócio da aplicação, como apresentado no Capítulo 5. Na Figura E.1 apresenta-se um exemplo de definição de serviços e na Figura E.2 a sua chamada utilizando o sistema referido. O código presente na primeira figura define uma classe cujos métodos podem ser chamados remotamente, de nome *MyRemoteObject*, contendo um método *dummy()*, que é chamado pelo código presente na segunda figura.

```
#using plainsocket only, DEBUG purposes
ServerSocketFactory.set_stypes(PlainSocket)

#My remote object definition
class MyRemoteObject(RemoteObject):
        def __init__(self):
                super(MyRemoteObject,self).__init__()
        def dummy(self):
                logger.warning("Dummy function called!")
                return "dummy"

# register remote object
RemoteFactory.register(name="test",obj=MyRemoteObject())
# create a unlimited queue
queue = Queue(0)
# create our application workers
workers = ThreadPool("Minions",10,queue)
ss = SocketSelector('127.0.0.1',13373,socket_factory=ServerSocketFactory,workers=workers)
ss.start()
logger.info("Remote object and server initialized!")

# signal handling
def catch_signals(signr, stack):
        global ss, can_exit, workers
        logger.info("Caught signal " + str(signr)+ ". Exiting.")
        workers.hard_kill()
        ss.stop()
        logger.info("Exited.")
        can_exit = True
        sys.exit(signr)

signal.signal(signal.SIGABRT, catch_signals)
signal.signal(signal.SIGFPE, catch_signals)
signal.signal(signal.SIGTERM, catch_signals)
signal.signal(signal.SIGILL, catch_signals)
signal.signal(signal.SIGINT, catch_signals)
signal.signal(signal.SIGSEGV, catch_signals)

logger.info("Press Ctrl+C to exit.")
can_exit = False
while not can_exit:
        sleep(3)
```

Figura E.1: Exemplo de definição de serviços RPC

```
logger = logging.getLogger()

INIT = False
ro = None
queue = None
workers = None
cm = None

def getRemote(instance=""):
        global INIT, ro, queue, workers, cm
        if(not INIT):
                ServerSocketFactory.set_stypes(PlainSocket)
                ClientSocketFactory.set_stypes(PlainSocket)
                queue = Queue(0)
                workers = ThreadPool("InstanceWorkers",10,queue)
                cm = ConnectionManager("127.0.0.1",13373,10,workers=workers)
                ro = RemoteFactory.remote("sga_ids_ro",cm)
        return ro

services = getRemote()

services.dummy()
```

Figura E.2: Exemplo de chamada de serviços RPC

E.2 Servidor Baseado em Twisted

Na Figura E.3 apresenta-se um exemplo de um servidor SSL com verificação de certificados implementado utilizando a biblioteca *Twisted*.

```
from OpenSSL import SSL
from twisted.internet import ssl, reactor
from twisted.internet.protocol import Factory, Protocol

class Echo(Protocol):
        def dataReceived(self, data):
                self.transport.write(data)

def verifyCallback(connection, x509, errnum, errdepth, ok):
        if not ok:
                print 'invalid␣cert␣from␣subject:', x509.get_subject()
                return False
        else:
                print "Certs␣are␣fine"
                return True

if __name__ == '__main__':
        factory = Factory()
        factory.protocol = Echo
        myContextFactory = ssl.DefaultOpenSSLContextFactory(
                'keys/server.key', 'keys/server.crt'
        )
    ctx = myContextFactory.getContext()
        ctx.set_verify(
                SSL.VERIFY_PEER | SSL.VERIFY_FAIL_IF_NO_PEER_CERT,
        verifyCallback
        )
    # Since we have self-signed certs we have to explicitly
    # tell the server to trust them.
    ctx.load_verify_locations("keys/ca.pem")

        reactor.listenSSL(8000, factory, myContextFactory)
    reactor.run()
```

Figura E.3: Exemplo de servidor SSL em Twisted

Apêndice F

INVESTIGAÇÃO DO PROTOCOLO PRELUDE

F.1 Ambiente de Teste

De forma a ter acesso a todas as partes do sistema, incluindo as chaves criptográficas de ambos os lados da conexão, foi instalado um sistema com o sistema operativo *Debian Linux* e configurado um sensor *Prelude-LML* e um *Prelude Manager* (o componente do *Prelude IDS* que realmente recebe os eventos). O *Prelude-LML* foi configurado para reportar os eventos para o *Prelude Manager* através da *interface* de rede *Loopback*[1]. O *Prelude Manager* foi configurado para armazenar os eventos para um SGBD *PostgreSQL* local. O código fonte da *libprelude* versão 1.0.1 foi descarregado e descomprimido para consulta. O sistema encontra-se esquematizado na Figura F.1.

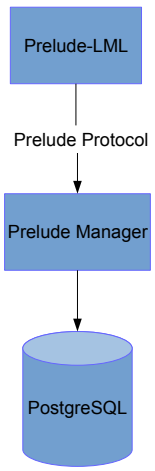

Figura F.1: Ambiente de teste

F.2 *Certificate Authority*

Como já foi referido, o Prelude-IDS utiliza TLS com certificados criados pela CA que inclui para encriptação e autenticação dos clientes e servidores. O comando *prelude-admin* é utilizado para gerar os certificados. Para remover todas as dependências sobre o *Prelude IDS* foi necessário criar uma CA de substituição compatível com a original.

[1] Interface de rede local à máquina.

No sistema *Prelude IDS*, o *Prelude Manager* é a CA e o servidor, enquanto que o cliente é o sensor. Depois de instalar o sistema *Prelude IDS* e configurá-lo, os certificados foram encontrados em pastas próprias para cada cliente e servidor dentro da pasta */etc/prelude/profile/*. O conteúdo de cada um dos ficheiros presentes nessas pastas é explicado nas Tabelas F.1 e F.2.

Ficheiro	**Conteúdo**
server.ca	Certificado do CA.
server.keycrt	Certificado do servidor criado pelo CA.
analyzerid	Contem uma linha com o *Analyzer ID* do servidor.
config	Ficheiro de configuração que contem duas linhas e é o mesmo para todos os clientes e servidores.
key	Chave privada do servidor.

Tabela F.1: Ficheiros do servidor

Ficheiro	**Conteúdo**
server.ca	Certificado da CA.
server.keycrt	Certificado do servidor criado pela CA.
analyzerid	Contem uma linha com o Analyzer ID do cliente.
config	Ficheiro de configuração que contem duas linhas e é o mesmo para todos os clientes e servidores.
key	Chave privada do cliente.
client.keycrt	Certificado do cliente criado pela CA.
client.trusted	Ficheiro que contém os certificados dos servidores em que o cliente confia. Se o certificado do servidor não estiver incluído neste ficheiro, o cliente recusa ligar-se.

Tabela F.2: Ficheiros do cliente

Os certificados foram inspeccionados com o comando *certtool* para revelar as peculiaridades que o protocolo pode requerer. As Figuras F.2, F.3 e F.4 mostram o *output* para o certificado da CA, servidor e cliente, respectivamente. A partir deste *output* e do comando *"prelude-admin list -l"* utilizado para listar todos os sensores e servidores, apresentado na Figura F.5, é imediatamente aparente que o campo *dnQualifier* contem o *Analyzer ID* do programa respectivo (servidor ou cliente).

```
root@magneto:/etc/prelude/profile/prelude-manager# certtool -i < server.ca
X.509 Certificate Information:
        Version: 3
        Serial Number (hex): 45cf235249700b00
        Issuer: dnQualifier=3219684956819269
        Validity:
                Not Before: Sun Sep 01 23:35:33 UTC 2013
                Not After: Tue Jan 19 03:14:07 UTC 2038
        Subject: dnQualifier=3219684956819269
        Subject Public Key Algorithm: RSA
        Algorithm Security Level: Legacy (2048 bits)
                Modulus (bits 2048):
                        00:99:d8:e8:81:1b:22:76:6b:66:92:b1:4b:13:0c:b5
                        4e:08:db:2e:a1:f4:85:a3:b7:f8:10:7f:73:28:df:cb
                        b0:7c:58:df:71:59:36:69:60:19:79:b8:54:0f:d2:66
                        88:69:2e:03:f2:ee:03:12:e3:31:91:37:64:78:8c:20
```

Figura F.2: Informação do certificado do CA

```
root@magneto:/etc/prelude/profile/prelude-manager# certtool -i < server.keycrt
X.509 Certificate Information:
        Version: 3
        Serial Number (hex): 45cf235249700b00
        Issuer: dnQualifier=3219684956819269
        Validity:
                Not Before: Sun Sep 01 23:35:33 UTC 2013
                Not After: Tue Jan 19 03:14:07 UTC 2038
        Subject: dnQualifier=3219684956819269
        Subject Public Key Algorithm: RSA
        Algorithm Security Level: Legacy (2048 bits)
                Modulus (bits 2048):
                        00:99:d8:e8:81:1b:22:76:6b:66:92:b1:4b:13:0c:b5
                        4e:08:db:2e:a1:f4:85:a3:b7:f8:10:7f:73:28:df:cb
                        b0:7c:58:df:71:59:36:69:60:19:79:b8:54:0f:d2:66
                        88:69:2e:03:f2:ee:03:12:e3:31:91:37:64:78:8c:20
```

Figura F.3: Informação do certificado do servidor

```
root@magneto:/etc/prelude/profile/prelude-lml# certtool -i < client.keycrt
X.509 Certificate Information:
        Version: 3
        Serial Number (hex): 521f1270
        Issuer: dnQualifier=5911974131578670374
        Validity:
                Not Before: Thu Aug 29 09:20:48 UTC 2013
                Not After: Fri Aug 29 09:20:48 UTC 2014
        Subject: CN=6,dnQualifier=5917468704716558959
        Subject Public Key Algorithm: RSA
        Algorithm Security Level: Normal (2432 bits)
                Modulus (bits 2432):
                        00:bb:8e:87:60:91:3a:39:0f:33:9a:39:2e:77:fc:7a
                        b3:0e:19:c9:8f:fb:5d:1c:cc:2b:d4:d4:70:63:5b:9e
                        4e:b7:d4:b0:df:7c:85:ab:c3:bf:39:bf:13:99:2a:a8
                        52:0d:eb:8e:c2:43:94:75:30:c5:de:90:af:61:8e:d9
                        c5:da:75:d1:34:c8:40:fe:71:a8:bd:c8:e7:bb:0b:f5
```

Figura F.4: Informação do certificado do cliente

```
root@magneto:/etc/prelude/profile/prelude-lml# prelude-admin list -l
Profile          UID     GID     AnalyzerID          Permission      Issuer AnalyzerID
-------------------------------------------------------------------------------------
prelude-manager prelude prelude 3219684956819269    n/a             n/a
prelude-lml     root    root    5917468704716558959 idmef:w admin:r 5911974131578670374
```

Figura F.5: *Output* do “prelude-admin list -l”

Adicionalmente, o *Common Name* do certificado do cliente consiste num número. Depois de alguma procura no código fonte da *libprelude* por funções que obtenham o *Common Name*, os dados representados por esse número foram encontrados no ficheiro *src/tls-utils.c* como se apresenta na Figura F.6.

```
int tls_certificate_get_permission(gnutls_session session,
                                   prelude_connection_permission_t *permission)
{
        int ret, tmp;
        char buf[1024];
        gnutls_x509_crt cert;
        size_t size = sizeof(buf);
        const gnutls_datum *data;

        data = gnutls_certificate_get_ours(session);
        if ( ! data )
                return prelude_error_verbose(PRELUDE_ERROR_TLS, "could not get own certificate");

        ret = gnutls_x509_crt_init(&cert);
        if ( ret < 0 )
                return prelude_error_verbose(PRELUDE_ERROR_TLS, "error initializing certificate: %s",
                        gnutls_strerror(ret));

        ret = gnutls_x509_crt_import(cert, data, GNUTLS_X509_FMT_DER);
        if ( ret < 0 ) {
                gnutls_x509_crt_deinit(cert);
                return prelude_error_verbose(PRELUDE_ERROR_TLS, "error importing certificate: %s",
                        gnutls_strerror(ret));
        }

        ret = gnutls_x509_crt_get_dn_by_oid(cert, GNUTLS_OID_X520_COMMON_NAME, 0, 0, buf, &size);
        if ( ret < 0 ) {
                gnutls_x509_crt_deinit(cert);
                return prelude_error_verbose(PRELUDE_ERROR_TLS, "could not get certificate CN field: %s",
                        gnutls_strerror(ret));
        }

        ret = sscanf(buf, "%d", &tmp);
        if ( ret != 1 ) {
                gnutls_x509_crt_deinit(cert);
                return prelude_error_verbose(PRELUDE_ERROR_TLS, "certificate analyzerid value '%s' is invalid", buf);
        }

        *permission = (prelude_connection_permission_t) tmp;
        gnutls_x509_crt_deinit(cert);

        return 0;
}
```

Figura F.6: Significado do número no *Common Name*

Depois de uma procura pela definição do tipo *prelude_connection_permission_t*, os valores possíveis para as permissões foram encontrados no ficheiro número *src/include/prelude-connection.h.* Trata-se de uma máscara de bits que representa as permissões do cliente com o servidor e pode tomar qualquer combinação dos valores na Tabela F.3.

Permissão	**Valor**	**Descrição**
PRELUDE_CONNECTION_PERMISSION_IDMEF_READ	1	O cliente pode receber mensagens IDMEF.
PRELUDE_CONNECTION_PERMISSION_ADMIN_READ	2	O cliente pode ler mensagens ADMIN.
PRELUDE_CONNECTION_PERMISSION_IDMEF_WRITE	4	O cliente pode enviar mensagens IDMEF.
PRELUDE_CONNECTION_PERMISSION_ADMIN_WRITE	8	O cliente pode enviar mensagens OPTION.

Tabela F.3: Valores de permissões

A única peça de informação que falta é o processo de geração do *Analyzer ID*, processo esse que

foi encontrado no ficheiro *prelude-admin/prelude-admin.c* e que se apresenta na Figura F.7. Do código retira-se que o Analyzer ID consiste num valor de 64 bits em que os primeiros 32 bits contém os segundos desde o *UNIX epoch*[79] e os segundos 32 bits contém o resto em microsegundos do valor anterior.

```
static uint64_t generate_analyzerid(void)
{
        struct timeval tv;
        union {
                uint64_t val64;
                uint32_t val32[2];
        } combo;

        gettimeofday(&tv, NULL);

        combo.val32[0] = tv.tv_sec;
        combo.val32[1] = tv.tv_usec;

        return combo.val64;
}
```

Figura F.7: Função de geração de *Analyzer IDs*.

Munido da informação aqui apresentada, foi possível implementar uma CA compatível com a CA incluída com o sistema *Prelude IDS* e que gera certificados e chaves que quando copiados para as pastas correctas são aceites pela *libprelude,* limitando deste modo as dependências sobre a bilbioteca referida aos sensores, que relembro não são desenvolvidos pela Dognædis.

F.3 Desencriptação do Tráfego

Com o ambiente de testes descrito em F.1, obteve-se o porto em que o *Prelude Manager* está à escuta utilizando o comando *netstat*, como demonstrado na Figura F.8.

```
root@magneto:~# netstat -tlpn|grep prelude
tcp     0  0 127.0.0.1:4690      0.0.0.0:*           LISTEN    10427/prelude-manag
```

Figura F.8: *Output* do comando *netstat*

Como temos controlo dos dois lados da conexão, é possível capturar e desencriptar tráfego TLS com o analisador de protocolos *Wireshark*[43]. No entanto, para que isso seja possível é necessário desligar a troca de chaves *Diffie Hellman,* porque esta garante *Perfect Forward Secrecy[80]* às ligações TLS o que impede a sua desencriptação. Depois de algumas tentativas, o autor descobriu que o valor *"EXPORT"* na opção *tls-options* dos ficheiros */etc/prelude/default/client.conf* e */etc/prelude-manager/prelude-manager.conf* desliga a troca de chaves *Diffie Hellman*, como exemplificado na Figura F.9.

```
#
# TLS options (only available with GnuTLS 2.2.0 or higher):
#
# Sets priorities for the ciphers, key exchange methods, macs and
# compression methods.
#
# "NORMAL" option enables all "secure" ciphersuites. The 256-bit
# ciphers are included as a fallback only. The ciphers are sorted by
# security margin.
#
# "SECURE128" flag enables all "secure" ciphersuites with ciphers up to
# 128 bits, sorted by security margin.
#
# "SECURE256" flag enables all "secure" ciphersuites including the 256
# bit ciphers, sorted by security margin.
#
# "EXPORT" all the ciphersuites are enabled, including the low-security
# 40 bit ciphers.
#
# "NONE" nothing is enabled. This disables even protocols and
# compression methods.
#
# Note that much more settings might be enabled or disabled using this
# option: please see gnutls_priority_init(3) for more details.
#
# The default settings is "NORMAL".
tls-options = EXPORT
```

Figura F.9: Opção de configuração *tls-options*

Inserindo a chave primária do *Prelude Manager* na lista de chaves RSA do *Wireshark*, associada com o endereço 127.0.0.1, porto 4690 e protocolo SSL, antes de iniciar uma captura enquanto se força o *Prelude-LML* a emitir eventos (autenticando-se como *root* por exemplo), podemos agora ver todo o tráfego desencriptado, como se vê na Figura F.10.

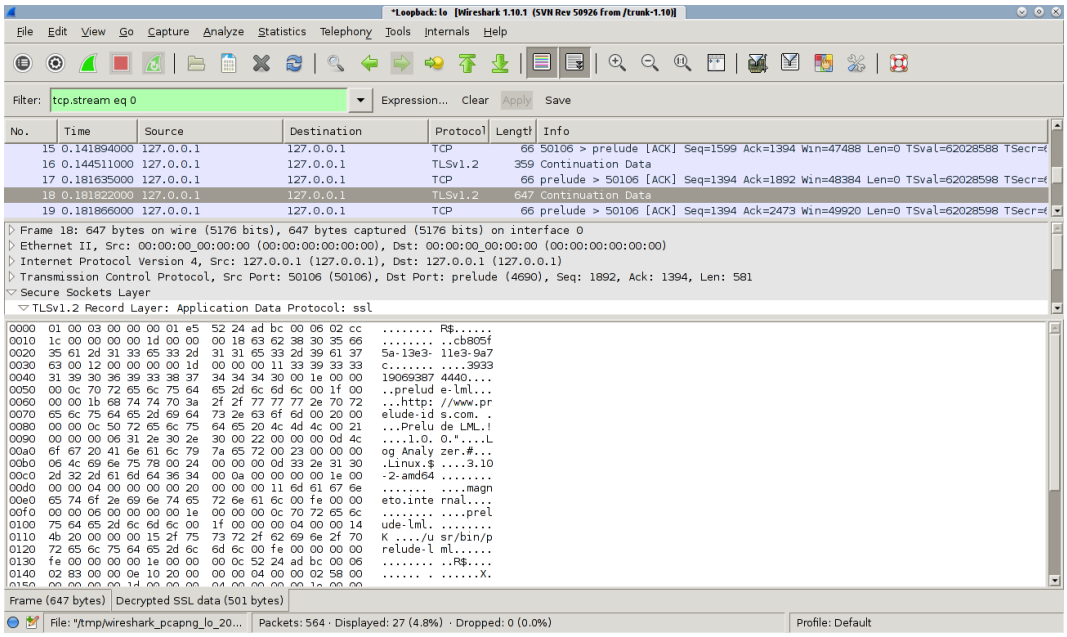

Figura F.10: Tráfego Prelude desencriptado

Escolhendo a opção *"Follow SSL Stream..."* no *Wireshark* podemos ver o tráfego colorido em vermelho para dados enviados pelo servidor e azul para dados enviados pelo cliente. O tráfego também se encontra diferenciado pela presença de um espaçamento maior antes do tráfego enviado pelo servidor como se pode ver na Figura F.11. Neste momento o tráfego foi gravado para um ficheiro e aberto num editor hexadecimal que oferece funções como conversões entre hexadecimal e decimal e entre ordem de bytes de rede e máquina, facilitando assim a interpretação dos dados.

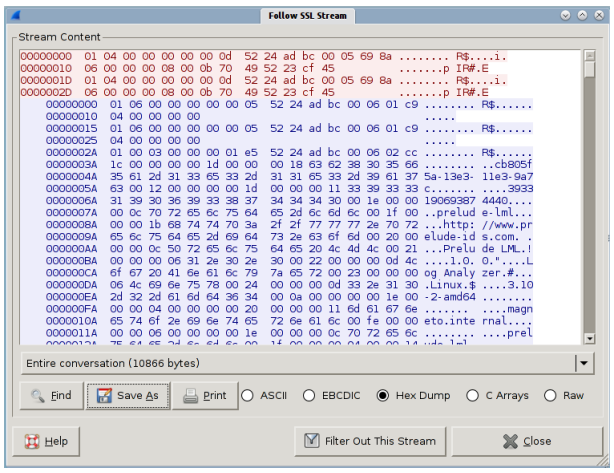

Figura F.11: Tráfego colorido

F.4 Fluxo do Protocolo

Examinando o tráfego apresentado anteriormente, é preciso determinar o que o servidor envia para o cliente imediatamente a seguir à negociação TLS. Fazendo uso da única documentação escrita da *libprelude* presente em *docs/api/html/libprelude-prelude-msg.html*, podemos encontrar o formato do cabeçalho das mensagens do protocolo, que é apresentado na Tabela F.4.

Campo	Comprimento	Descrição
versão	8 bits	Versão do protocolo.
tipo	8 bits	Tipo da mensagem.
prioridade	8 bits	Prioridade da mensagem.
fragmento	8 bits	Fragmento.
comprimento	32 bits	Comprimento da mensagem em bytes excluindo o cabeçalho.

Tabela F.4: Cabeçalho das mensagens *Prelude*

O campo versão é ignorado por agora e contém a versão do protocolo. O campo tipo indica o tipo de mensagem que pode ser um dos valores na Tabela F.5, encontrados no ficheiro *prelude-message-id.h* na directoria *src/include/*. Alguns valores têm a descrição de não relevante e não serão explorados já que o objectivo é implementar a recepção de eventos em IDMEF e não reimplementar completamente o servidor *Prelude Manager*. Estas mensagens podem ser exploradas no futuro se necessário, partindo da informação aqui apresentada.

Tipo	Valor	Descrição
PRELUDE_MSG_IDMEF	0	Uma mensagem XML IDMEF.
PRELUDE_MSG_ID	3	Não relevante.
PRELUDE_MSG_AUTH	4	Mensagem do servidor indicando a aceitação ou não dos certificados enviados pelo cliente.
PRELUDE_MSG_CM	5	Não relevante.
PRELUDE_MSG_CONNECTION_CAPABILITY	6	Mensagem do cliente indicando que permissões pretende.
PRELUDE_MSG_OPTION_REQUEST	7	Não relevante.
PRELUDE_MSG_OPTION_REPLY	8	Não relevante.

Tabela F.5: Tipos de Mensagem *Prelude*

O campo prioridade pode ser utilizado pelo servidor para priorizar mensagens. Se o campo fragmento for diferente de 0 (zero) então outra parte da mensagem será enviada de seguida pelo servidor, precedida de mais um cabeçalho de mensagem *Prelude*. O campo comprimento contém o comprimento em bytes excluindo o cabeçalho. No mesmo ficheiro de documentação encontra-se a informação que no interior da mensagem *Prelude*, esta pode conter *tags* com o formato apresentado na Tabela F.6.

Campo	**Comprimento**	**Descrição**
tag	8 bits	Tipo de *Tag*.
comprimento	32 bits	Comprimento dos dados excluindo este cabeçalho.
dados	<comprimento> bytes	Dados.
0xFE	8 bits	Byte que assinala o fim da *Tag*.

Tabela F.6: Formato de *Tag*

Cada campo dados pode conter mais *tags* no mesmo formato de forma análoga ao XML. Os valores possíveis para as *tags* foram encontrados no ficheiro *src/include/idmef-message-id.h*.

Portanto, primeiro o servidor envia uma mensagem do tipo PRELUDE_MSG_AUTH com o valor PRELUDE_MSG_AUTH_SUCCEED significando que aceita o cliente. O outro valor possível para essa mensagem é o PRELUDE_MSG_AUTH_FAILED que sinaliza a falha de autenticação. Depois de receber esta mensagem do servidor, o cliente envia uma mensagem do tipo PRELUDE_MSG_CONNECTION_CAPABILITY, indicando as permissões que requer. As permissões são enviadas na forma de um byte que contém a máscara de bits já referida em F.2, seguida por 32 bits que são ignorados pela *libprelude*.

Depois desta troca de mensagens o cliente começa a enviar mensagens do tipo PRELUDE_MSG_IDMEF e o servidor mantém-se silencioso a partir deste ponto. Examinando os ficheiros *src/include/idmef-message-id.h*, *src/include/idmef-tree-wrap.h* e *src/idmef-message-read.h* começando pela função *idmef_message_read(...)* podemos ver que a estrutura das mensagens pode ser derivada do código fonte de modo quase directo, com se vê na Figura F.12.

```
/**
 * idmef_message_read:
 * @message: Pointer to a #idmef_message_t object.
 * @msg: Pointer to a #prelude_msg_t object, containing a message.
 *
 * Read an idmef_message from the @msg message, and
 * store it into @message.
 *
 * Returns: 0 on success, a negative value if an error occured.
 */
int idmef_message_read(idmef_message_t *message, prelude_msg_t *msg)
{
        int ret;
        void *buf;
        uint8_t tag;
        uint32_t len;

        while ( 1 ) {
                ret = prelude_msg_get(msg, &tag, &len, &buf);
                if ( ret < 0 )
                        return ret;

                switch ( tag ) {

                        case IDMEF_MSG_MESSAGE_VERSION: {
                                prelude_string_t *tmp = NULL;

                                ret = prelude_extract_string_safe(&tmp, buf, len, msg);
                                if ( ret < 0 )
                                        return ret;

                                idmef_message_set_version(message, tmp);
                                break;
                        }

                        case IDMEF_MSG_ALERT_TAG: {
                                int ret;
                                idmef_alert_t *tmp = NULL;

                                ret = idmef_message_new_alert(message, &tmp);
                                if ( ret < 0 )
                                        return ret;



                                ret = idmef_alert_read(tmp, msg);
                                if ( ret < 0 )
                                        return ret;
```

Figura F.12: idmef_message_read()

Como exemplo, para uma mensagem do tipo PRELUDE_MSG_IDMEF contendo uma tag IDMEF do tipo IDMEF_MSG_ALERT_TAG é chamada a função *idmef_alert_read(...)* do mesmo ficheiro que procurar por todas as tags possíveis no interior desta até todas as *tags* terem sido fechadas ou o fim dos dados ser encontrado.

F.5 Formato de Dados

Dentro das *tags* estão normalmente dados para o texto interior das *tags* XML ou atributos das mesmas. Estes dados seguem uma codificação que depende do seu tipo, que se apresenta na Tabela F.7. As funções da *libprelude* que descodificam os valores estão no ficheiro *src/include/prelude-extract.h* ou *src/idmef-message-read.c*. Todos os números maiores que 8 bits são armazenados em ordem de bytes da rede. As funções apresentadas são as acabadas em *_safe* já que estas efectuam verificação de tamanho dos dados e são portanto mais seguras.

Tipo	Formato	Função na libprelude
Unsigned Byte	Sem modificação	prelude_extract_uint8_safe(...)
Unsigned Word	16 bit Word	prelude_extract_uint16_safe(...)
Unsigned Integer	32 bit Unsigned Integer	prelude_extract_uint32_safe(...)
Integer	32 bit Signed Integer	prelude_extract_int32_safe(...)
64 bit Unsigned Integer	64 bit Unsigned Integer	prelude_extract_uint64_safe(...)
64 bit Signed Integer	64 bit signed Integer	prelude_extract_int64_safe(...)
Float	32 bit Float	prelude_extract_float_safe(...)
String	String terminado em 0	prelude_extract_string_safe(...)
Time	32 bits Unsigned Integer com os segundos desde a UNIX epoch seguido de 32 bits Unsigned Integer com o resto em microsegundos e 32 bits Unsigned Integer com a diferença em minutos para o GMT.	prelude_extract_time_safe(...)

Tabela F.7: Formatos de dados

Existe um tipo especial de dados que inclui a indicação do tipo de dados nele contidos. Este é útil para quando o tipo de dados não pode ser ditado pela estrutura do XML. O tipo de dados é indicado por um valor 32 bits que determina o tipo de dados interior. As funções já apresentadas são usadas para os dados interiores.

Apêndice G

PROTOCOLO PRELUDE

Neste apêndice descreve-se o protocolo *Prelude*. Este protocolo tem como função encapsular mensagens IDMEF, que se trata de um esquema XML definido pelo RFC 4765, com o intuito de facilitar a comunicação de eventos de segurança. A informação aqui presente foi fruto de um esforço de investigação com recurso a capturas de tráfego de rede e sua análise com o Wireshark[43] com base na análise do código fonte em *C* da biblioteca *libprelude*.

G.1 Componentes do Prelude-IDS

Para entender a documentação que se segue torna-se necessário especificar os componentes envolvidos no sistema Prelude-IDS e as suas funções no contexto do protocolo:

Sensor Representa qualquer sensor com a capacidade de envio de eventos para o sistema *Prelude-IDS*. O próprio *Prelude-IDS* inclui o HIDS *Prelude-LML*.

Prelude-manager Servidor que recebe os eventos dos sensores utilizando para isso o protocolo descrito neste apêndice.

Libprelude Biblioteca incluída com o Prelude-IDS e que é utilizada quer pelos sensores, quer pelo prelude-manager para implementar o protocolo descrito.

G.2 Encriptação e Autenticação

O protocolo utiliza ligações sobre o protocolo TCP/IP e porta 4690. Para manter a segurança dos dados as ligações são encriptadas por TLS utilizando certificados criados através da CA incluída com o *prelude-manager*. O campo dnQualifier dos certificados gerados para cada cliente e servidor especifica o identificador do analisador (Analyzer ID no RFC 4765) de cada cliente ou servidor. O campo *CommonName* (CN) do certificado do cliente é um número que especifica as permissões que o cliente possui com o servidor que gerou o certificado. Este número é uma máscara de bits em que cada bit representa uma permissão de acordo com a tabela G.1.

Permissão	Valor
PRELUDE_CONNECTION_PERMISSION_IDMEF_READ	1
PRELUDE_CONNECTION_PERMISSION_ADMIN_READ	2
PRELUDE_CONNECTION_PERMISSION_IDMEF_WRITE	4
PRELUDE_CONNECTION_PERMISSION_ADMIN_WRITE	8

Tabela G.1: Tabela de valores de permissões

G.3 Estrutura das Mensagens

O protocolo consiste numa troca de mensagens entre o *sensor* e *prelude-manager*. As mensagens podem ser de informação protocolar ou encapsular documentos XML IDMEF, que são codificados em formato binário ao invés de XML puro para uma maior compressão. A estrutura das mensagens é detalhada na Tabela G.2.

Nome	Tipo de Dados	Tamanho	Descrição
Versão	*Unsigned Byte*	8 bits	Versão do protocolo.
Tag	*Unsigned Byte*	8 bits	Código que indica de que tipo de mensagem se trata.
Prioridade	*Unsigned Byte*	8 bits	Prioridade da mensagem.
Is_Fragment	*Unsigned Byte*	8 bits	Indica se a mensagem é só uma parte e há mais um fragmento.
Comprimento	*Unsigned Integer*	32 bits	Indica o comprimento dos dados em bytes excluindo o cabeçalho.
Segundos	*Unsigned Integer*	32 bits	Segundos desde o UNIX epoch.
Micro Segundos	*Unsigned Integer*	32 bits	Parte em Micro Segundos do valor acima
Conteúdo	*Byte*	Comprimento x Bytes	Conteúdo da mensagem.

Tabela G.2: Cabeçalho de Mensagem Prelude

O conteúdo dos campos é descrito de seguida:

Versão É usado para indicar a versão do protocolo, para que o servidor se possa adaptar. Até este momento, tem sempre o valor 1.

Tag Indica o tipo de mensagem. Os valores possíveis estão contidos na Tabela G.3.

Prioridade É normalmente sempre 0 (zero) e representa a prioridade da mensagem. Pode ser usado pelo servidor para priorizar a mensagem.

Is_Fragment A mensagem pode ser dividida em vários fragmentos. No caso de este não ser o ultimo fragmento, este campo é diferente de 0 (zero) e será enviada outra mensagem de seguida cujo conteudo deve ser somado a esta. O último fragmento tem o valor 0 (zero) neste campo.

Comprimento Comprimento do campo conteúdo em *bytes*.

Segundos Segundos desde o *UNIX epoch* (1 de Janeiro de 1970).

Micro segundos O resto em micro-segundos do valor acima. Valor em micro-segundos entre dois valores de segundos consecutivos acima.

Conteúdo O conteúdo da mensagem em si.

Tipo	Valor	Descrição
PRELUDE_MSG_IDMEF	0	Uma mensagem XML IDMEF.
PRELUDE_MSG_ID	3	Não relevante.
PRELUDE_MSG_AUTH	4	Mensagem do servidor indicando a aceitação ou não dos certificados enviados pelo cliente.
PRELUDE_MSG_CM	5	Não relevante.
PRELUDE_MSG_CONNECTION_CAPABILITY	6	Mensagem do cliente indicando que permissões pretende.
PRELUDE_MSG_OPTION_REQUEST	7	Não relevante.
PRELUDE_MSG_OPTION_REPLY	8	Não relevante.

Tabela G.3: Tipos de mensagem Prelude

Os valores com a descrição "Não relevante." são mensagens que não são necessárias para o objectivo da implementação, que é um receptor de eventos. No entanto, com a informação detalhada neste documento seria possível descobrir o significado dessas mensagens também.

O campo conteúdo da mensagem pode ser um valor, uma *tag* que representa um valor ou *tags* XML. Cada *Tag* por sua vez pode conter valores ou outras *Tags*. As *tags* que podem ser embutidas dentro de outras são definidas pelo RFC 4765. O objectivo desta estrutura é imitar a organização do XML. Existe uma *tag* especial de fecho (IDMEF_MSG_END_OF_TAG) com o valor 0xFE (253 representado em hexadecimal) que tem o objectivo de representar o fechar de qualquer *tag* XML.

Nome	Tipo de Dados	Tamanho	Descrição
Tag	Unsigned Byte	8 bits	Código que indica de que tipo de conteúdo se trata (valor ou tag).
Comprimento	Unsigned Integer	32 bits	Indica o comprimento dos dados em bytes excluindo estes dados.
Conteudo	Depende da *tag*	Comprimento x bytes	Conteúdo.
0xFE	Byte	8 bits	Fecho de tag XML.

Tabela G.4: Formato de tag

G.4 Fluxo do protocolo

Na Figura G.1 encontra-se um diagrama de sequência que procura descrever o fluxo do protocolo depois da conexão ser estabelecida e do *TLS Handshake* ser bem sucedido.

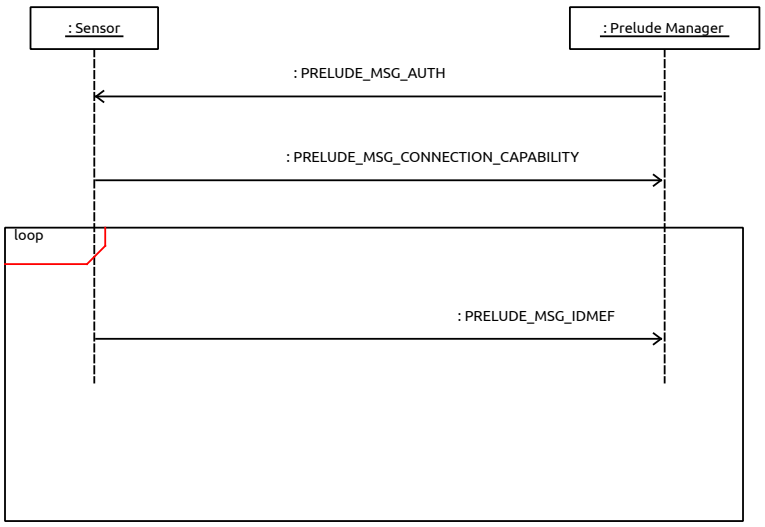

Figura G.1: Sequência do protocolo Prelude

O fluxo que o protocolo segue é o seguinte:

1. O servidor (Prelude Manager) envia uma PRELUDE_MSG_AUTH com o valor PRELUDE_MSG_AUTH_SUCCEED ou PRELUDE_MSG_AUTH_FAILED que sinaliza que os certificados foram verificados com resultado positivo ou negativo, respectivamente. Esta verificação é efectuada independentemente do *TLS Handshake* já que o *prelude-manager* possui uma lista de certificados que devem ser rejeitados.

2. O cliente (Sensor) envia uma mensagem PRELUDE_MSG_CONNECTION_CAPABILITY com o valor das permissões que pretende ter (por exemplo PRELUDE_CONNECTION_PERMISSION_IDMEF_WRITE para poder enviar alertas).

3. O cliente envia as mensagens dos eventos consecutivamente, não havendo resposta do servidor. Estas são PRELUDE_MSG_IDMEF e podem conter MSG_IDMEF_ALERT_TAG ou MSG_IDMEF_HEARTBEAT_TAG, que correspondem respectivamente à *tag alert* ou *heartbeat* do IDMEF, utilizando a estrutura descrita no ponto G.3. Uma *tag alert* corresponde a um evento de segurança enquanto que uma *tag heartbeat* corresponde a uma mensagem que o sensor envia periodicamente com o seu estado. Ambas estão documentadas na secção G.5 que se segue.

A ligação não é desligada até o sensor ou servidor ser parado. No caso de ser o servidor a desligar a ligação, o sensor entra em modo de *fallback* e grava as mensagens para posterior envio quando uma tentativa de ligação for bem sucedida.

G.5 Tags

Nesta secção serão descritas as *tags* e os valores possíveis para o seu texto interior e atributos. As *tags* que podem ser incluídas no interior de outras podem ser consultadas no RFC 4765. A sua especificação total aqui seria demasiado extensa.

G.5.1 IDMEF_MSG_ADDITIONAL_DATA_TAG

Classe no RFC	AdditionalData
Valor	0

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ADDITIONAL_DATA_TYPE	29	type	String
IDMEF_MSG_ADDITIONAL_DATA_MEANING	30	meaning	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_ADDITIONAL_DATA_DATA	31	Ver nota.	Ver nota.

Tabela G.5: Tag IDMEF_MSG_ADDITIONAL_DATA_TAG

NOTA: No caso da *sub-tag* IDMEF_MSG_ADDITIONAL_DATA_DATA o nome da *tag* varia com o tipo de dados especificado anteriormente pela *tag* IDMEF_MSG_ADDITIONAL_DATA_TYPE e a codificação pelo valor *Integer* de 32 bits contido na *tag*. O valor para a *sub-tag* encontra-se na mensagem a seguir e este deve ser extraído usando a descodificação adequada. Os tipos e nomes de *sub-tag* correspondentes são detalhados na Tabela G.6 e os tipos especificados pelo Integer e a codificação correspondente estão presentes na Tabela G.7.

Tipo	**Valor**	**Sub-tag**
error	-1	N/A
string	0	string
byte	1	byte
character	2	character
date-time	3	date-time
integer	4	integer
ntpstamp	5	ntpstamp
portlist	6	portlist
real	7	real
boolean	8	boolean
byte-string	9	byte-string
xmltext	10	xml

Tabela G.6: Correspondência entre tipos e subtags

Tipo	**Valor**	**Codificação**
unknown	0	N/A
char	1	Unsigned Byte
byte	2	Base64
uint32	3	Unsigned 32 bit Integer
uint64	4	Unsigned 64 bit Integer
float	5	32 bit Float
char-string	6	String terminado em 0 (zero).
byte-string	7	Base64

Tabela G.7: Correspondência entre tipos e codificação

G.5.2 IDMEF_MSG_REFERENCE_TAG

Classe no RFC	Reference
Valor	1

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_REFERENCE_ORIGIN	29	origin	String
IDMEF_MSG_REFERENCE_MEANING	32	meaning	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_REFERENCE_NAME	30	name	String
IDMEF_MSG_REFERENCE_URL	31	url	String

Tabela G.8: Tag IDMEF_MSG_REFERENCE_TAG

G.5.3 IDMEF_MSG_CLASSIFICATION_TAG

Classe no RFC	Classification
Valor	2

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_CLASSIFICATION_IDENT	29	origin	String
IDMEF_MSG_CLASSIFICATION_TEXT	30	meaning	String

Tabela G.9: Tag IDMEF_MSG_CLASSIFICATION_TAG

G.5.4 IDMEF_MSG_USER_ID_TAG

Classe no RFC	UserID
Valor	3

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_USER_ID_IDENT	29	ident	String
IDMEF_MSG_USER_ID_TYPE	30	type	Enumeração: -1: "error", 0: "original-user" 1: "current-user" 2: "target-user" 3: "user-privs" 4: "current-group" 5: "group-privs" 6: "other-privs"
IDMEF_MSG_USER_ID_TTY	31	tty	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_USER_ID_NAME	32	name	String
IDMEF_MSG_USER_ID_NUMBER	33	number	String

Tabela G.10: Tag IDMEF_MSG_USER_ID_TAG

G.5.5 IDMEF_MSG_USER_TAG

Classe no RFC	User
Valor	4

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_USER_IDENT	29	ident	String
IDMEF_MSG_USER_CATEGORY	30	category	Enumeração: -1: "error", 0: "unknown", 1: "application", 2: "os-device",

Tabela G.11: Tag IDMEF_MSG_USER_TAG

G.5.6 IDMEF_MSG_ADDRESS_TAG

Classe no RFC	Address
Valor	5

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ADDRESS_IDENT	29	ident	String
IDMEF_MSG_ADDRESS_CATEGORY	30	category	Enumeração: -1: "error" 0: "unknown" 1: "atm" 2: "e-mail" 3: "lotus-notes" 4: "mac" 5: "sna" 6: "vm" 7: "ipv4-addr" 8: "ipv4-addr-hex" 9: "ipv4-net" 10: "ipv4-net-mask" 11: "ipv6-addr" 12: "ipv6-addr-hex" 13: "ipv6-net" 14: "ipv6-net-mask"
IDMEF_MSG_ADDRESS_VLAN_NAME	31	vlan-name	String
IDMEF_MSG_ADDRESS_VLAN_NUM	32	vlan-num	Unsigned Int 32

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_ADDRESS_ADDRESS	33	address	String
IDMEF_MSG_ADDRESS_NETMASK	34	netmask	String

Tabela G.12: Tag IDMEF_MSG_ADDRESS_TAG

G.5.7 IDMEF_MSG_PROCESS_TAG

Classe no RFC	Process
Valor	6

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_PROCESS_IDENT	29	ident	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_PROCESS_NAME	30	name	String
IDMEF_MSG_PROCESS_PID	31	pid	String
IDMEF_MSG_PROCESS_PATH	32	path	Unsigned Int 32 bits
IDMEF_MSG_PROCESS_ARG	33	arg	String
IDMEF_MSG_PROCESS_ENV	34	env	String

Tabela G.13: Tag IDMEF_MSG_PROCESS_TAG

G.5.8 IDMEF_MSG_WEB_SERVICE_TAG

Classe no RFC	WebService
Valor	7

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_WEB_SERVICE_URL	29	url	String
IDMEF_MSG_WEB_SERVICE_CGI	30	cgi	String
IDMEF_MSG_WEB_SERVICE_HTTP_METHOD	31	http-method	String
IDMEF_MSG_WEB_SERVICE_ARG	32	arg	String

Tabela G.14: Tag IDMEF_MSG_WEB_SERVICE_TAG

G.5.9 IDMEF_MSG_SNMP_SERVICE_TAG

Classe no RFC	SNMPService
Valor	8

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_SNMP_SERVICE_OID	29	oid	String
IDMEF_MSG_SNMP_SERVICE_MESSAGE_PROCESSING_MODEL	30	messageProcessingModel	Unsigned Int 32 bits
IDMEF_MSG_SNMP_SERVICE_SECURITY_MODEL	31	securityModel	Unsigned Int 32 bits
IDMEF_MSG_SNMP_SERVICE_SECURITY_NAME	32	securityName	String
IDMEF_MSG_SNMP_SERVICE_SECURITY_LEVEL	33	securityLevel	Unsigned Int 32 bits
IDMEF_MSG_SNMP_SERVICE_CONTEXT_NAME	34	contextName	String
IDMEF_MSG_SNMP_SERVICE_CONTEXT_ENGINE_ID	35	contextEngineID	String
IDMEF_MSG_SNMP_SERVICE_COMMAND	36	serviceCommand	String
IDMEF_MSG_SNMP_SERVICE_COMMUNITY	37	serviceCommunity	String

Tabela G.15: Tag IDMEF_MSG_SNMP_SERVICE_TAG

G.5.10 IDMEF_MSG_SERVICE_TAG

Classe no RFC	Service
Valor	9

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_SERVICE_IDENT	29	ident	String
IDMEF_MSG_SERVICE_IP_VERSION	30	ip_version	Unsigned Int 8 bit
IDMEF_MSG_SERVICE_IANA_PROTOCOL_NUMBER	31	iana_protocol_number	Unsigned Int 8 bit
IDMEF_MSG_SERVICE_IANA_PROTOCOL_NAME	32	iana_protocol_name	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_SERVICE_NAME	33	name	String
IDMEF_MSG_SERVICE_PORT	34	port	Unsigned Int 16 bits
IDMEF_MSG_SERVICE_PORTLIST	35	portlist	String
IDMEF_MSG_SERVICE_PROTOCOL	36	protocol	String

Tabela G.16: Tag IDMEF_MSG_SERVICE_TAG

G.5.11 IDMEF_MSG_NODE_TAG

Classe no RFC	Node
Valor	10

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_NODE_IDENT	29	ident	String
IDMEF_MSG_NODE_CATEGORY	30	category	Enumeração: -1: "error" 0: "unknown" 1: "ads" 2: "afs" 3: "coda" 4: "dfs" 5: "dns" 6: "hosts" 7: "kerberos" 8: "nds" 9: "nis" 10: "nisplus" 11: "nt" 12: "wfw"
IDMEF_MSG_NODE_LOCATION	31	location	String
IDMEF_MSG_NODE_NAME	32	name	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_SERVICE_NAME	33	name	String
IDMEF_MSG_SERVICE_PORT	34	port	Unsigned Int 16 bits
IDMEF_MSG_SERVICE_PORTLIST	35	portlist	String
IDMEF_MSG_SERVICE_PROTOCOL	36	protocol	String

Tabela G.17: Tag IDMEF_MSG_NODE_TAG

G.5.12 IDMEF_MSG_SOURCE_TAG

Classe no RFC	Source
Valor	11

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_SOURCE_IDENT	29	ident	String
IDMEF_MSG_SOURCE_SPOOFED	30	spoofed	Enumeração: -1: "error" 0: "unknown" 1: "yes" 2: "no"
IDMEF_MSG_SOURCE_INTERFACE	31	interface	String

Tabela G.18: Tag IDMEF_MSG_SOURCE_TAG

G.5.13 IDMEF_MSG_FILE_ACCESS_TAG

Classe no RFC	File Access
Valor	12

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_FILE_ACCESS_PERMISSION	29	permission	Int 32 bit

Tabela G.19: Tag IDMEF_MSG_FILE_ACCESS_TAG

G.5.14 IDMEF_MSG_INODE_TAG

Classe no RFC	Inode
Valor	13

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_INODE_CHANGE_TIME	29	change-time	Tempo
IDMEF_MSG_INODE_NUMBER	30	number	Unsigned Int 32 bit
IDMEF_MSG_INODE_MAJOR_DEVICE	31	major-device	Unsigned Int 32 bit
IDMEF_MSG_INODE_MINOR_DEVICE	32	minor-device	Unsigned Int 32 bit
IDMEF_MSG_INODE_C_MAJOR_DEVICE	33	c-major-device	Unsigned Int 32 bit
IDMEF_MSG_INODE_C_MINOR_DEVICE	34	c-minor-device	Unsigned Int 32 bit

Tabela G.20: Tag IDMEF_MSG_INODE_TAG

G.5.15 IDMEF_MSG_CHECKSUM_TAG

Classe no RFC	Checksum
Valor	14

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_CHECKSUM_ALGORITHM	31	algorithm	Enumeração: -1: "error" 1: "MD4" 2: "MD5" 3: "SHA1" 4: "SHA2-256" 5: "SHA2-384" 6: "SHA2-512" 7: "CRC-32" 8: "Haval" 9: "Tiger" 11: "Gost"

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_CHECKSUM_VALUE	29	value	String
IDMEF_MSG_CHECKSUM_KEY	30	key	String

Tabela G.21: Tag IDMEF_MSG_CHECKSUM_TAG

G.5.16 IDMEF_MSG_FILE_TAG

Classe no RFC	File
Valor	15

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_FILE_IDENT	29	ident	String
IDMEF_MSG_FILE_NAME	30	name	String
IDMEF_MSG_FILE_PATH	31	path	String
IDMEF_MSG_FILE_CREATE_TIME	32	create-time	Tempo
IDMEF_MSG_FILE_MODIFY_TIME	33	modify-time	Tempo
IDMEF_MSG_FILE_ACCESS_TIME	34	access-time	Tempo
IDMEF_MSG_FILE_CATEGORY	37	category	Enumeração: -1: "error" 1: "current" 2: "original"
IDMEF_MSG_FILE_FSTYPE	38	fstype	Enumeração: -1: "error" 1: "ufs" 2: "efs" 3: "nfs" 4: "afs" 5: "ntfs" 6: "fat16" 7: "fat32" 8: "pcfs" 9: "joliet" 10: "iso9660"

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_FILE_DATA_SIZE	35	value	String
IDMEF_MSG_FILE_DISK_SIZE	36	key	String

Tabela G.22: Tag IDMEF_MSG_FILE_TAG

G.5.17 IDMEF_MSG_LINKAGE_TAG

Classe no RFC	Linkage
Valor	16

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_LINKAGE_CATEGORY	29	category	Enumeração: -1: "error" 1: "hardlink" 2: "mount-point" 3: "reparse-point" 4: "shortcut" 5: "stream" 6: "symbolic-link"

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_LINKAGE_NAME	30	name	String
IDMEF_MSG_LINKAGE_PATH	31	path	String

Tabela G.23: Tag IDMEF_MSG_LINKAGE_TAG

G.5.18 IDMEF_MSG_TARGET_TAG

Classe no RFC	Target
Valor	17

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_TARGET_IDENT	29	ident	String
IDMEF_MSG_TARGET_DECOY	30	decoy	Enumeração: -1: "error" 0: "unknown" 1: "application" 2: "os-device"
IDMEF_MSG_TARGET_INTERFACE	31	interface	String

Tabela G.24: Tag IDMEF_MSG_TARGET_TAG

G.5.19 IDMEF_MSG_ANALYZER_TAG

Classe no RFC	Analyzer
Valor	18

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ANALYZER_ANALYZERID	29	analyzerid	String
IDMEF_MSG_ANALYZER_NAME	30	name	String
IDMEF_MSG_ANALYZER_MANUFACTURER	31	interface	String
IDMEF_MSG_ANALYZER_MODEL	32	model	String
IDMEF_MSG_ANALYZER_VERSION	33	version	String
IDMEF_MSG_ANALYZER_CLASS	34	class	String
IDMEF_MSG_ANALYZER_OSTYPE	35	ostype	String
IDMEF_MSG_ANALYZER_OSVERSION	36	osversion	String

Tabela G.25: Tag IDMEF_MSG_ANALYZER_TAG

G.5.20 IDMEF_MSG_ALERTIDENT_TAG

Classe no RFC	AlertIdent
Valor	19

Texto Interior	**Valor**	**Tipo**
IDMEF_MSG_ALERTIDENT_ALERTIDENT	29	String

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ALERTIDENT_ANALYZERID	30	analyzerid	String

Tabela G.26: Tag IDMEF_MSG_ALERTIDENT_TAG

G.5.21 IDMEF_MSG_IMPACT_TAG

Classe no RFC	Impact
Valor	20

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_IMPACT_SEVERITY	29	severity	Enumeração: -1: "error" 1: "info" 2: "low" 3: "medium" 4: "high"
IDMEF_MSG_IMPACT_COMPLETION	30	completion	Enumeração: -1: "error" 1: "failed" 2: "succeeded"
IDMEF_MSG_IMPACT_TYPE	31	type	Enumeração: -1: "error" 0: "other" 1: "admin" 3: "dos" 3: "file" 4: "recon" 5: "user"
IDMEF_MSG_IMPACT_DESCRIPTION	32	description	String

Tabela G.27: Tag IDMEF_MSG_IMPACT_TAG

G.5.22 IDMEF_MSG_ACTION_TAG

Classe no RFC	Action
Valor	21

Texto Interior	**Valor**	**Tipo**
IDMEF_MSG_ACTION_DESCRIPTION	30	String

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ACTION_CATEGORY	29	category	Enumeração: -1: "error" 0: "other" 1: "block-installed" 2: "notification-sent" 3: "taken-offline"

Tabela G.28: Tag IDMEF_MSG_ACTION_TAG

G.5.23 IDMEF_MSG_CONFIDENCE_TAG

Classe no RFC	Confidence
Valor	22

Texto Interior	**Valor**	**Tipo**
IDMEF_MSG_CONFIDENCE_CONFIDENCE	30	Float

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_CONFIDENCE_RATING	29	rating	Enumeração: -1: "error" 0: "numeric" 1: "low" 2: "medium" 3: "high"

Tabela G.29: Tag IDMEF_MSG_CONFIDENCE_TAG

G.5.24 IDMEF_MSG_ASSESSMENT_TAG

Classe no RFC	Assessment
Valor	23

Tabela G.30: Tag IDMEF_MSG_ASSESSMENT_TAG

G.5.25 IDMEF_MSG_TOOL_ALERT_TAG

Classe no RFC	ToolAlert
Valor	24

Sub-Tag	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_TOOL_ALERT_NAME	29	name	String
IDMEF_MSG_TOOL_ALERT_COMMAND	30	command	String

Tabela G.31: Tag IDMEF_MSG_TOOL_ALERT_TAG

G.5.26 IDMEF_MSG_CORRELATION_ALERT_TAG

Classe no RFC	CorrelationAlert
Valor	25

Sub-Tag	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_CORRELATION_ALERT_NAME	29	name	String

Tabela G.32: Tag IDMEF_MSG_CORRELATION_ALERT_TAG

G.5.27 IDMEF_MSG_OVERFLOW_ALERT_TAG

Classe no RFC	Overflow Alert
Valor	26

Sub-Tag	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_OVERFLOW_ALERT_PROGRAM	29	program	String
IDMEF_MSG_OVERFLOW_ALERT_SIZE	30	size	Unsigned Int 32 bit
IDMEF_MSG_OVERFLOW_ALERT_BUFFER	31	buffer	Dados codificados em Base64

Tabela G.33: Tag IDMEF_MSG_OVERFLOW_ALERT_TAG

G.5.28 IDMEF_MSG_ALERT_TAG

Classe no RFC	Alert
Valor	27

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_ALERT_MESSAGEID	29	messageid	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_ALERT_CREATE_TIME	30	CreateTime	Tempo
IDMEF_MSG_ALERT_DETECT_TIME	31	DetectTime	Tempo
IDMEF_MSG_ALERT_ANALYZER_TIME	32	AnalyzerTime	Tempo

Tabela G.34: Tag IDMEF_MSG_ALERT_TAG

NOTA: Ao contrário de os demais sub-elementos, os sub-elementos CreateTime, DetectTime e AnalyzerTime para além de incluírem no seu texto interior o tempo extraído em formato para leitura humana, incluem também um atributo "*ntpstamp*" que contém o valor de segundos desde o *UNIX Epoch* seguido do resto deste valor em microsegundos, correspondente ao valor do tempo extraído.

G.5.29 IDMEF_MSG_HEARTBEAT_TAG

Classe no RFC	Heartbeat
Valor	28

Atributo	**Valor**	**Nome**	**Tipo**
IDMEF_MSG_HEARTBEAT_MESSAGEID	29	messageid	String

Sub-Tag	**Valor**	**Nome**	**Texto interior**
IDMEF_MSG_HEARTBEAT_CREATE_TIME	30	CreateTime	Tempo
IDMEF_MSG_HEARTBEAT_ANALYZER_TIME	31	AnalyzerTime	Tempo
IDMEF_MSG_HEARTBEAT_HEARTBEAT_INTERVAL	32	HeartbeatInterval	Unsigned Int 32 bit

Tabela G.35: Tag IDMEF_MSG_HEARTBEAT_TAG

NOTA: Ao contrário de os demais sub-elementos, os sub-elementos CreateTime, DetectTime e AnalyzerTime para além de incluírem no seu texto interior o tempo extraído em formato para leitura

humana, incluem também um atributo "*ntpstamp*" que contém o valor de segundos desde o *UNIX Epoch* seguido do resto deste valor em microsegundos, correspondente ao valor do tempo extraído.