FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

IPBrick: Segurança em *Cloud*

Hugo Manuel Carvalho Fonseca

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Ricardo Santos Morla

Supervisor Externo: Luís Miguel Ramalhão Ribeiro

Co-supervisor Externo: Eduardo da Costa Maia

25 de Julho de 2016

Resumo

O contexto tecnológico atual tem proporcionado alternativas bastante atrativas ao mundo profissional, possibilitando que as empresas diminuam os gastos a nível da infraestrutura tecnológica que as suporta. Tal deve-se às tecnologias que derivaram do conceito de Computação em *Cloud*, vulgarmente designadas por soluções *Cloud*.

A adoção de uma solução *Cloud* por uma empresa permite que esta terceirize responsabilidades de aquisição, gestão e suporte de uma infraestrutura tecnológica necessária à empresa a, por exemplo, um Provedor de Serviços em *Cloud* (CSP). Desta forma, a empresa pode diminuir os gastos destinados à sua infraestrutura tecnológica ou realocar os recursos libertados para outras áreas.

Por esse motivo, a IPBrick S.A. começou a disponibilização dos seus serviços e *softwares* sobre uma plataforma tecnológica assente no conceito de *Cloud*, cuja administração pertence a um CSP. No entanto, a execução de serviços críticos e armazenamento de dados confidenciais num provedor externo é visto como uma possível vulnerabilidade do sistema para o cliente, uma vez que a segurança e a privacidade do conteúdo da sua área pessoal na *Cloud* não lhe podem ser eficazmente garantidas.

Com este projeto pretende-se suprimir essa ineficácia na garantia de segurança através da manipulação da *firewall* Linux IPtables e de ferramentas de Deteção e Prevenção de Intrusões (IDS/IPS), por forma a proporcionar uma gestão *user-friendly* ao utilizador, mas também tendo em conta o dinamismo caraterístico de um ambiente em *Cloud*.

Desta forma, este projeto teve como principais objetivos a introdução de uma interface simplificada para a gestão da segurança do sistema, mas também que a solução desenvolvida incorporasse a caraterística dinâmica da *Cloud* e, autonomamente, se adaptasse ao desempenho, em tempo-real, do sistema. Tal implicou que o projeto fosse dividido em duas fases principais, a realização de testes de desempenho às ferramentas de IDS/IPS alvo e a implementação de mecanismos que envolvem três componentes: as ferramentas de IDS/IPS, a *Firewall* IPtables e o sistema em si.

Após a finalização desta Dissertação, foi possível concluir que os objetivos propostos foram eficazmente ultrapassados. Por um lado, a segurança ficou otimizada pela complementarização da *Firewall* com o sistema de IDS/IPS, por outro também foi dotada de inteligência com a implementação automática de novas regras a partir dos *logs* do sistema de IDS/IPS. Além disso, o impacto da introdução do sistema de IDS/IPS foi igualmente otimizado, quer pela geração e implementação de regras na *Firewall* (eliminando a necessidade da análise do tráfego de rede pelo sistema de IDS/IPS), como também da auto-adaptação do sistema de IDS/IPS ao desempenho e caraterísticas do sistema.

Abstract

The current technological context has provided very attractive options to the professional world, providing to companies a way to minimize their spendings with technological infrastructures which support them. This is due to technologies that derivates from Cloud Computing concept, also known as Cloud-based solutions.

Adopting a Cloud solution by a company allows it to outsource multiple responsabilities about having a technological infrastructure like acquisition, management and support, and designate a Cloud Service Provider (CSP) to take care of it. This way, a company should be able to save costs on technology infrastructure or reallocating resources to another company's area.

Considering this, IPBrick S.A. began to provide its services and software over a technology platform based on the concept of Cloud whose administration belongs to a CSP. However, the execution of critical services and need of saving confidential data in an external location has been seen as a possible system vulnerability to the customer since security and privacy of his personal area in the Cloud can not be effectively guaranteed.

This project is intended to eliminate this inefficiency in the security by manipulating the Linux IPtables firewall and Detection and Intrusion Prevention tools (IDS/IPS), in order to provide a user-friendly management to the user but also having in consideration the dynamism of a Cloud environment.

Therefore, this project has as main objectives the introduction of a simplified interface for system security management, but also the solution should be able to lead with dynamic characteristic of Cloud and autonomously adapt to the real-time performance of the system. These meant that the project was divided into two main phases, the fulfillment of performance tests on IDS/IPS tools and implementation of mechanisms involving three components: IDS/IPS tools, IPtables Firewall and the system itself.

After finishing the development of this thesis, it was possible to conclude that the proposed main goals were effectively overcome. The existent security was optimized by adding the IDS/IPS system to work with Firewall and by endowing Firewall with automatic implementation of new rules from IDS/IPS system's logs. Furthermore, the impact of the introduction of IDS/IPS system was also optimized by generation and implementation of rules on Firewall (which eliminates its necessity to analyze the traffic), as well as the self-adaptation of IDS/IPS system to system performance and characteristics.

Agradecimentos

Ao longo da realização desta Dissertação, muitas foram as pessoas que, direta e indiretamente, me apoiaram e ajudaram a realizar a sua concretização. Por esse motivo, não podia deixar esta Dissertação sem uma palavra de apreço a todos aqueles que tornaram possível a realização deste trabalho, e que me levou à concretização de um dos meus objetivos de vida: a conclusão do Mestrado com sucesso.

Ao professor Ricardo Morla, orientador desta Dissertação, agradeço-lhe as suas valiosas sugestões de desenvolvimento e de melhorias do projeto desenvolvido, bem como o seu incentivo e persistência nas inúmeras horas de análise ao longo do desenvolvimento da Dissertação.

Ao Eng.º Miguel Ramalhão e Eng.º Eduardo Maia, supervisor e co-supervisor externos desta Dissertação, agradeço-lhes pelas excelentes sugestões de trabalho e esclarecimentos de dúvidas no contexto da análise do problema e implementação do projeto inerente a esta Dissertação, bem como pelo apoio na integração no contexto profissional.

À minha família, em especial aos meus pais, agradeço-lhes a enorme compreensão e apoio incondicional numa fase da minha vida em que não pude lhes dar o tempo e dedicação que os sempre habituei.

Aos meus amigos, em especial aos meus amigos mais próximos, que, sempre que necessário, deram a sua opinião, ajuda e principalmente motivação.

E, por último, mas não menos importantes, agradeço a todos aqueles que não foram mencionados, mas que de alguma forma também contribuíram para a elaboração deste trabalho.

Hugo Manuel Carvalho Fonseca

"The precondition to freedom is security."

Rand Beers

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	*Application Programming Interface*
CPU	*Central Processing Unit*
CSP	*Cloud Service Provider*
E/S	Entrada/Saída
HIDS	*Host-based Intrusion Detection System*
HIPS	*Host-based Intrusion Prevention System*
IaaS	*Infrastructure as a Service*
IDS	*Intrusion Detection System*
IPS	*Intrusion Prevention System*
NIDS	*Network-based Intrusion Detection System*
NIPS	*Network-based Intrusion Prevention System*
NIST	National Institute of Standards and Technology
PaaS	*Platform as a Service*
RAM	*Random Access Memory*
SaaS	*Softare as a Service*
SLA	*Service-Level Agreement*
SOA	*Service-Oriented Architecture*
TI	Tecnologias da Informação
UI	Interface Gráfica
UPS	*Uninterruptible Power Supplies*

Capítulo 1

Introdução

Neste Capítulo introdutório será abordado o enquadramento desta Dissertação, no que respeita ao tema e ao contexto da mesma, a motivação inerente, os objetivos que visam ser cumpridos na resolução do problema e a forma como a mesma se encontra estruturada.

1.1 Contexto

No âmbito do projeto final de curso, do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, foi desenvolvida a presente Dissertação. Esta Dissertação, cujo tema se denomina "IPBrick: Segurança em *Cloud*", surgiu pela necessidade de suprimir a falta de eficácia na garantia dada aos clientes nos contratos de privacidade e segurança de cada CSP, serviços que se baseiam num conceito cada vez mais emergente: a computação em *Cloud*.

Atualmente, a sociedade encontra-se cada vez mais dependente dos recursos tecnológicos e cada vez mais móvel, pelo que as *Clouds* têm tido um grande impacto, possibilitando a cada utilizador o acesso à sua área na *Cloud* onde e quando pretender, necessitando apenas de uma ligação à rede e/ou Internet. Além disso, o impacto das *Clouds* tem tendência em ser cada vez maior, fruto da evolução dos *smartphones* e da crescente utilização de ferramentas baseadas na *Web*.

Uma *Cloud* é baseada no conceito de *Cloud Computing*, o qual consiste em conceitos e arquiteturas previamente existentes. São eles os conceitos de computação em rede e computação de utilidade, estruturados numa arquitetura cliente-servidor. Desta forma, uma *Cloud* representa um designado conjunto de recursos físicos (por exemplo, computadores e servidores) que, inseridos num esquema de rede baseado no conceito de computação em rede, proporcionam uma quantidade de memória bastante alargada, bem como um maior poder de processamento e um maior grau de redundância, responsável por satisfazer os pedidos dos clientes de um dado CSP.

As *Clouds* podem ser diferenciadas em três diferentes arquiteturas principais: pública, na qual os recursos físicos ficam alojados nas instalações do fornecedor e cada utilizador partilha recursos computacionais e físicos com outros utilizadores; privada, que pode ter os seus recursos físicos alojados num fornecedor, estando a área dedicada a cada utilizador separada virtualmente das

restantes, ou nas instalações do cliente, em que todo o ambiente *Cloud* é estritamente dedicado a este; por último, a *Cloud* híbrida, cuja arquitetura é dinâmica, podendo o utilizador decidir quais os dados e serviços que estarão alojados numa *Cloud* pública e quais os que estarão numa *Cloud* privada.

Neste sentido, a IPBrick S.A.[1] oferece uma solução que permite a execução de serviços por si desenvolvidos – nomeadamente, o sistema operativo IPBRICK OS – num espaço em *Cloud*, designado por cada cliente. Deste modo, cada cliente possui um espaço devidamente limitado, separado virtualmente de outros utilizadores, sendo baseada numa arquitetura baseada no modelo privado de desenvolvimento de *clouds*.

Desta forma, as empresas podem focar as suas atenções no seu *core business*, permitindo um aumento da produtividade do seu negócio, uma vez que a gestão e utilização de serviços e *softwares* necessários é passível de se realizar de uma forma mais simples e centralizada.

1.2 Motivação

Tal como foi referido na secção 1.1, a constante evolução tecnológica trouxe uma solução que tem um impacto bastante positivo na economia de uma empresa: as *Clouds*. Através das *Clouds*, uma empresa pode minimizar os custos despendidos em licenciamento de *software* e, sobretudo, com *hardware* e infraestruturas de suporte, contratando um espaço em *Cloud* que lhe disponibilize serviço de armazenamento e de processamento. Por esse motivo, a IPBrick S.A. optou por começar a disponibilizar o seu sistema operativo IPBRICK OS[2] num ambiente em *Cloud*.

No entanto, as *Clouds* ainda possuem alguma controvérsia em relação à segurança: a sua forma de partilha e disponibilidade de recursos levanta novas questões ao nível da segurança que têm que ser resolvidas com novos paradigmas de segurança mais vocacionados para a *Cloud*. É, por isso, importante dotar a *Cloud* de mecanismos inteligentes, reagindo automaticamente contra ataques externos ou, inclusive, negligências do próprio utilizador, mas também permitir que este possua um maior controlo e informação, em tempo real, sobre o estado do seu espaço em *Cloud*.

É, neste contexto, que surge a motivação por esta Dissertação, procurando que a segurança em torno da *Cloud* seja melhorada com mecanismos inteligentes e reativos, possibilitando que os serviços IPBrick – nomeadamente, os serviços inerentes ao *software* IPBRICK OS – possam assegurar um maior conforto e confiança ao cliente quanto à questão da segurança.

1.3 Objetivos

Os objetivos que estão na base deste projeto assentam em mecanismos de segurança de perímetro, nomeadamente a *firewall* e os sistemas de IDS/IPS, permitindo a utilização de todas as funcionalidades IPBrick numa plataforma em execução na *Cloud*, mas com um nível de segurança equiparado ao existente num acesso local.

Desse modo, a solução a desenvolver deverá garantir a robustez e segurança de toda a plataforma de comunicações, não só ao nível da *firewall*, mas também de aplicações existentes, utilizando determinados mecanismos de segurança contra ataques externos, mas também de negligências do cliente na configuração dos serviços.

Assim, na abordagem à solução a implementar, o ponto de partida será a utilização de algumas ferramentas de auditoria *open-source*, por forma a identificar possíveis vulnerabilidades no sistema e serviços IPBrick.

Posteriormente, a abordagem incidirá sobre as duas componentes principais desta Dissertação: a *firewall* Linux, IPtables, e as ferramentas de IDS/IPS de código-aberto. A *firewall* deve ser dotada de mecanismos inteligentes e de uma maior robustez a falhas, possuindo uma caraterística reativa e dinâmica, detetando e bloqueando novos ataques em tempo-real. Por si só, a *firewall* não possui uma caraterística dinâmica, apenas reagindo consoante as regras constituintes da mesma, pelo que as ferramentas de IDS/IPS são as responsáveis por complementar a *firewall*.

Por outro lado, tendo em conta a integração da solução IPBrick na *Cloud*, e tendo em conta a utilização de recursos computacionais pelo sistema de IDS/IPS, as estratégias adotadas nesta Dissertação devem convergir para a implementação de métodos que evitem uma elevada carga no sistema pelos serviços não utilizados, diretamente, pelo utilizador, evitando assim latências nas respostas do servidor na *Cloud*.

Por último, tendo em conta um dos princípios básicos da IPBrick S.A. – a simplicidade e facilidade de utilização dos seus serviços –, o desenvolvimento da solução deve convergir para uma implementação *user-friendly*, possibilitando que, qualquer que seja o grau de conhecimento técnico do utilizador, este não sinta dificuldades na perceção de dados e configuração dos serviços.

1.4 Estrutura do Relatório

A estrutura deste relatório consiste em seis capítulos: o Capítulo 1, a Introdução, que retrata a contextualização do problema no seio da empresa e no mundo tecnológico atual; o Capítulo 2, a Revisão Bibliográfica, na qual são abordados os aspetos técnicos relativos ao tema da dissertação e apresentadas algumas tecnologias e soluções em vigor; os Capítulos 3 a 5, relativos ao estudo e planificação, implementação e testes da solução idealizada; e Capítulo 6, nas quais é feito uma análise global do projeto desenvolvido e introduzidos possíveis trabalhos futuros.

No Capítulo 1 é abordado o impacto do problema no seio da empresa, sendo apresentadas brevemente as deficiências que o atual sistema em vigor carece, e no mundo atual, expondo o impacto que o tema deste problema tem no mesmo e quais as consequências que este problema implica.

No Capítulo 2, são aprofundados os conceitos referidos na Introdução, sendo feito a descrição detalhada do conceito e estrutura de Computação em *Cloud*, abordado o capítulo da segurança na *Cloud* e finalizado com a apresentação da solução *Cloud* alvo de integração em ambiente *Cloud*.

No Capítulo 3, são apresentados o problema existente, o estudo das ferramentas de IDS/IPS consideradas e a estrutura e lógica da solução.

No Capítulo 4, são apresentados amostras da solução desenvolvida e que se relacionam com o conteúdo referido na secção 3.3, mas também outros requisitos da solução considerados importantes.

No Capítulo 5, é realizado a comprovação do bom funcionamento da solução, quer ao nível dos mecanismos influenciados diretamente pelo utilizador, como também dos mecanismos de automatização implementados. Além disso, é efetuado o tratamento dos resultados obtidos, procurando se perceber a qualidade do impacto – positivo, neutro ou negativo – desta solução no *software* IPBRICK OS.

Por último, no Capítulo 6, é apresentada uma visão geral sobre todo o trabalho realizado ao longo da Dissertação, através da avaliação dos objetivos considerados e, ainda, são apresentados possíveis trabalhos futuros no contexto desta Dissertação, identificando possíveis melhorias e/ou complementos à solução desenvolvida.

Capítulo 2

Revisão Bibliográfica

Neste Capítulo é exposta a definição e estrutura das *Clouds*, abordando as arquiteturas e modelos de desenvolvimento existentes, e explorados diversos conceitos que marcam a diferença entre o conceito de *Computação em Cloud* e os conceitos previamente existentes. Dado o caráter do tema da Dissertação, é igualmente explorado a segurança na *Cloud*, sendo apresentadas algumas tecnologias existentes que possibilitam a segurança das E/S de um espaço na *Cloud*. Por último, é apresentado o sistema operativo IPBRICK OS, sobre o qual são executados os diversos serviços e aplicações IPBrick.

2.1 Introdução

Nos últimos tempos, tem-se assistido a um novo serviço que providencia armazenamento e poder de processamento, que surge de um conceito emergente: a Computação em *Cloud*. Este novo serviço resultante são as *Clouds*, que proporcionam aos seus utilizadores uma redução económica ao nível de *hardware* e infraestruturas de suporte e uma maior flexibilidade e facilidade de acesso aos recursos alojados no seu espaço na *Cloud*.

Um serviço de *Cloud* pode dividir-se em diversos aspetos, nomeadamente ao nível da arquitetura que estes estão assentes e que providencia um tipo de serviço diferente ao utilizador, mas também quanto ao CSP que fornece o serviço. Assim, ao nível da arquitetura de uma *Cloud*, esta pode ser dividida ao nível de **(i)** modelos de serviço, que identificam o grau do serviço de *Cloud* prestado, e **(ii)** modelos de desenvolvimento, que identificam a arquitetura física e lógica dos recursos prestados por um serviço de *Cloud*. Nas secções 2.2.3 e 2.2.4, respetivamente, serão abordados estas duas caraterísticas da arquitetura da *Cloud* mais exaustivamente.

Por outro lado, ao nível do CSP, a não existência de *standards* universais para todos os CSP's impossibilita uma uniformização do serviço de *Cloud* e que obriga à utilização de API's específicas a cada CSP. Na secção 2.2.7 será abordado mais detalhadamente este aspeto.

2.2 Definição e Estrutura da Computação em *Cloud*

2.2.1 História e Conceito de Computação em *Cloud*

A Computação em *Cloud* é um conceito tecnológico emergente que resulta de conceitos e arquiteturas existentes, providenciando um serviço caraterístico por se adequar à exigência do cliente, não só a nível económico, mas ao nível de recursos alocados. Uma das definições mais aceites globalmente é dada pelo *National Institute of Standards and Technology* (NIST), que define a Computação em *Cloud* como [3]:

> *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.*

Baseada nos conceitos de Computação em Rede e Computação de Utilidade, a Computação em *Cloud* organiza estes dois conceitos numa arquitetura Cliente-Servidor. Deste modo, a Computação em *Cloud* consiste num mecanismo em que **(1)** o cliente faz um pedido à *Cloud* do CSP que está associado, **(2)** e, no centro de dados do CSP, este é responsável pela resposta ao pedido do cliente utilizando os conceitos de Computação em Rede e Computação de Utilidade para reunir o poder de processamento e tempo de resposta adequado, numa proporção que lhe permita evitar gastos desnecessários, retornando a resposta ao cliente. Desta forma, o *trade-off* entre a minimização de custos – um dos principais objetivos delineados desde os primórdios da computação – e a maximização de recursos é otimizado.

A minimização dos custos trata-se de um objetivo delineado desde o surgimento dos computadores. Os primeiros passos para cumprir esse objetivo foram dados por John McCarthy, na década de 1950, que introduziu o conceito de computação por tempo partilhado – um computador pode ser utilizado, simultaneamente, por dois ou mais utilizadores para realização de uma determinada tarefa. Para tal, o conceito baseia-se no intervalo de tempo ocioso entre cada processo, ou seja, através de uma multiplexagem nos tempos, é associado um dado *slot* de execução periódico dessa tarefa a cada utilizador.

No entanto, devido à definição da arquitetura Cliente-Servidor e da inexistência de outros conceitos e componentes necessários, o conceito de computação por tempo partilhado foi evoluindo lentamente. Assim, a implementação das *Clouds* passou por alguns estágios iniciais[4]:

1. nas décadas de 1960 e 1970, foi implementada a troca de informação por pacotes e criada a ARPANET, respetivamente. Desta forma, conseguiu-se a troca de informação, ordenadamente, por pacotes, relativos a diversos utilizadores, numa escala global;

2. introdução do WWW (*World Wide Web*), por Tim Berneers-Lee, funcionário no CERN, na década de 1990. Tal permitiu a globalização da troca de informação entre utilizadores de todo o globo;

3. no início do milénio de 2000, foi implementado o primeiros serviço baseado em Computação em *Cloud* pela Salesforce.com.

2.2.2 Caraterísticas e Termos

A definição de *Cloud*, a nível das componentes que esta integra, não é mais do que aquilo que os servidores e infraestruturas de provedores de serviço de armazenamento oferecem. No entanto, o que permite distinguir a *Cloud* dos demais serviços de armazenamento e de processamento é a forma que os seus componentes estão relacionados. Assim, por forma a se diferenciar o conceito de Computação em *Cloud* e o que torna um dado sistema numa solução *Cloud* é necessário clarificar quais os princípios atributos a si associados.

Segundo [3], existem cinco caraterísticas essenciais que devem existir numa dada infraestrutura de Tecnologias de Informação (TI) para que possa ser classificado como uma implementação de Computação em *Cloud*. São eles: rápida elasticidade, serviços medidos e na exigência do cliente, *pool* de recursos e acesso à rede em qualquer lugar. Seguidamente, serão explorados cada um desses atributos.

Elasticidade. Por forma a que, ao longo do tempo, os recursos alocados sejam ajustados à necessidade, a elasticidade carateriza a habilidade de adicionar ou remover recursos sob exigência, em tempo real. Desta forma, aplicações que possuem cargas de trabalho altamente inconstantes podem tirar proveito desta caraterística. Para tal, a existência de balanceadores de carga na rede de suporte do centro de dados é vital nesta caraterística, gerindo o poder de processamento total e redistribuindo-o consoante as aplicações e pedidos dos cliente assim o exijam: quando um determinando serviço necessita de um maior poder de processamento, estes responsabilizam pela obtenção de processamento de outra máquina (física ou virtual) com excedente de processamento, enquanto que, quando o processamento existente deixa de ser necessário, é mantido em *stand-by* ou realocado para outro serviço crítico. Em casos extremos, quando a capacidade total não é suficiente para satisfazer os pedidos dos clientes ou existe um mau pré-planeamento da capacidade necessário, traduz-se em pedidos não satisfeitos e erros, pelo que é da responsabilidade do CSP lidar com as queixas dos clientes. Por último, geralmente, o tempo de aplicação desta caraterística varia de alguns segundos a alguns minutos, podendo inclusive ser feita automaticamente, não necessitando de um pré-planeamento de capacidade necessário.

Serviços medidos e consoante a exigência do cliente. Tal como a caraterística anterior, estas duas caraterísticas são transparentes ao cliente, que apenas vê os seus pedidos serem satisfeitos. Os serviços medidos são relativos a faturação, controlo de acesso, otimização de recursos e planeamento da capacidade, pelo que se trata de uma tarefa exclusiva do CSP. Por outro lado, a disponibilização de serviços sob exigência é uma caraterística da extremidade do cliente, que pode dispor de qualquer capacidades extra sem que seja necessário uma intervenção humana na

extremidade do CSP. Esta caraterística deve-se à escabilidade da *Cloud*, que facilmente se adapta a modificações na rede e nos recursos atribuídos a um determinado cliente.

***Pool* de recursos.** Traduz-se na capacidade de uma *Cloud* gerir as três caraterísticas anteriores, através de um modelo *multi-tenant*, disponibilizando recursos, físicos e/ou virtuais, aos vários clientes inseridos na *pool* em questão. Desta forma, o CSP consegue otimizar os seus próprios recursos físicos e custos de operação, sendo que o cliente não sente qualquer impacto, uma vez que é transparente ao mesmo. Na figura 2.1 está ilustrado o papel dos balanceadores de carga, redistribuindo os pedidos dos clientes por instâncias com processamento disponível.

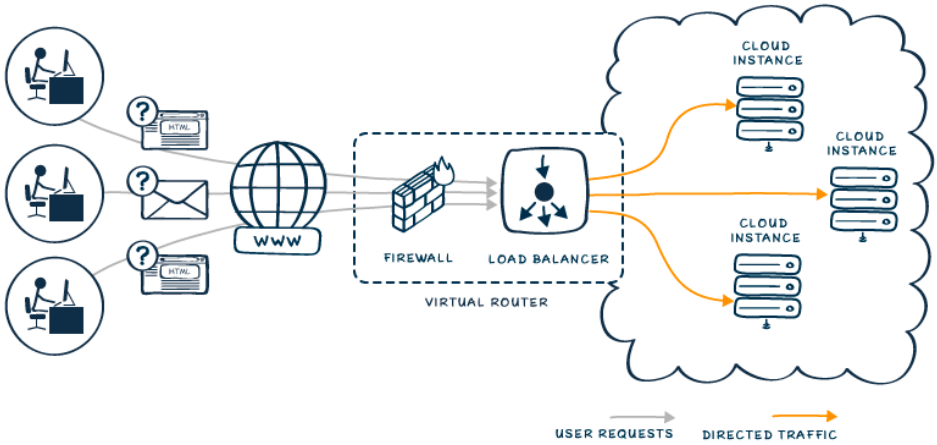


Figura 2.1: Exemplo figurativo do balanceamento de carga existente na rede de uma *Cloud*.

Acesso à Rede em qualquer lugar. A utilização da Internet como meio de transporte de dados entre as extremidades da comunicação providencia que o cliente possa ter um acesso ilimitado ao seu espaço em *Cloud*, em qualquer lugar e a qualquer momento, necessitando apenas de uma ligação válida à Internet. Além disso, o seu acesso também passa a ser independente do dispositivo utilizado pelo cliente, o que, numa época que os *smartphones* adquirem um papel cada vez mais importante no quotidiano humano, se revela como umas das vantagens mais preponderantes das *Clouds*.

Seguidamente, por forma a melhor compreender as caraterísticas e algum do conteúdo exposto ao longo deste relatório, serão abordados alguns termos relativos à Computação em *Cloud*. São eles: *Service-Level Agreement* (SLA), *multitenancy*, interoperabilidade e *Service-Oriented Architecture* (SOA).

SLA. Refere-se ao contrato negociado entre CSP e o cliente, na qual está assente um conjunto de regras e responsabilidades por ambas as partes quanto à utilização e qualidade do serviço prestado. É neste contrato que se encontram os requisitos mínimos do serviço de *Cloud* que um CSP deverá garantir ao cliente, mas também pormenorizado a segurança e privacidade de dados garantida sobre os dados do cliente. Trata-se de um contrato com termos quantitativos, pelo que o CSP deverá ser capaz de definir métricas que o apoiem na avaliação do serviço prestado ao cliente.

***Multitenancy*.** Refere-se a um modo de funcionamento de uma dada aplicação, cuja instância executada num servidor pode ser dividida virtualmente por forma a suportar diferentes utilizadores/*tenants* de diversas organizações, executando uma instância virtual da aplicação para cada um. Por esse motivo, um *tenant* poderá personalizar partes da aplicação, mas nunca caraterísticas principais da mesma. Desta forma, um ambiente *multitenant* torna-se desejável do ponto de vista do CSP pois facilita a implementação de atualizações.

Interoperabilidade. Abordada mais exaustivamente na secção 2.2.7, a interoperabilidade refere-se à relação entre sistemas de diferentes CSP's e à forma como estas trocam informação. No entanto, esta habilidade ainda não é eficazmente cumprida devido à inexistência de *standards* globalmente adotados pelos vários CSP's, sendo necessário recorrer às API's proprietárias.

SOA. Útil para contornar a não existência de *standards* globalmente aceites, trata-se de uma arquitetura de *software* que traduz serviços de um dado *software* para uma forma genérica, possibilitando a comunicação entre diferentes serviços através de mensagens. Um exemplo de arquitetura de *software* é a construção de um *Web Service* baseado em *Representational State Transfer* (REST).

2.2.3 Modelos de Serviço

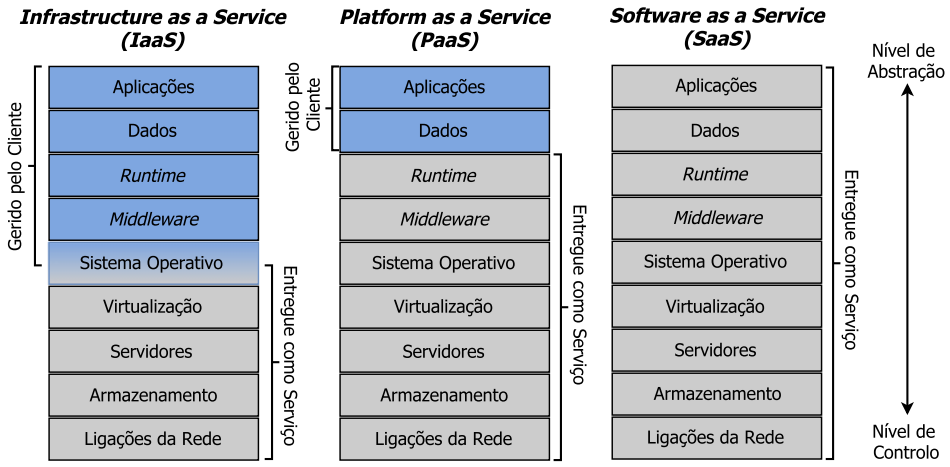


Figura 2.2: Pilha de serviços de um serviço de *Cloud*, consoante o modelo de serviço escolhido.

A responsabilidade pela gestão dos serviços concedida ao cliente pode ser representada em diferentes modelos de serviço. Tal como se pode observar na figura 2.2, um modelo de serviço representa uma dada camada de um todo, o qual poderá representar desde uma gestão nula dos serviços por parte do cliente até uma gestão parcial. Assim, enquanto numa implementação tradicional (nas instalações do cliente) existe um grau de abstração nulo para o cliente – este é,

geralmente, responsável por todos os aspetos intrínsecos à configuração e manutenção das componentes, pelo que possui um controlo total –, ao passo que, nos modelos de serviço da *Cloud*, existe graus de abstração e controlo que variam consoante o modelo aplicado.

Ao nível das soluções tradicionais, o nível de controlo é máximo, uma vez que cabe ao departamento de TI da empresa configurar e gerir as várias camadas representadas, diferindo da *Cloud* pelo controlo configurações de rede, servidores e armazenamento. Com um serviço de *Cloud* é possível uma divisão dos serviços em três diferentes modelos, sendo eles **(a)** o *Infrastructure as a Service* (IaaS), que possibilita ao cliente a gestão das camadas sobre o Sistema Operativo, **(b)** o *Platform as a Service* (PaaS), que disponibiliza uma plataforma de desenvolvimento onde os desenvolvedores podem executar, testar e desenvolver aplicações, e **(c)** o *Software as a Service* (SaaS), sendo o modelo mais comum na maioria das empresas, possibilitando um maior grau de abstração ao cliente. Seguidamente, serão abordados detalhadamente cada um dos três modelos.

Infrastructure as a Service

O IaaS é o modelo que possibilita um maior controlo das camadas nas quais os modelos de serviço estão assentes. Por esse motivo, proporciona um serviço de *Cloud* mais orientado para os responsáveis de TI de uma organização, permitindo que estes manipulem o seu espaço em *Cloud* para construir, por exemplo, um serviço baseado no PaaS ou no SaaS. Por esse motivo, este modelo baseia-se na provisão de recursos de armazenamento, computação e de rede por parte do CSP, sob a forma de serviço, consoante a exigência do cliente e é pago numa modalidade baseada na utilização do serviço.

Tal como se pode observar na figura 2.2, o CSP possui o equipamento e é o responsável pela gestão e manutenção do mesmo. Deste modo, o CSP é responsável por manter serviços e componentes como servidores, redes e armazenamento, que se enquadram nas camadas 4 a 7 do modelo Open Systems Interconnection (OSI) [5], enquanto o cliente gere as camadas de mais alto nível da pilha protocolar do OSI, incluindo o sistema operativo, os dados e as aplicações. É igualmente entre as camadas 4 e 7 (modelo OSI) que estão concentradas os principais componentes que garantem o bom funcionamento de um serviço de *Cloud*, como os balanceadores de carga, e introduzem uma segurança da camada lógica, nomeadamente as *firewalls* e ferramentas de IDS e IPS. Já o cliente tem um controlo na medida em que pode especificar o número de Unidades Centrais de Processamento (CPU), bem como a memória alocada por cada Máquina Virtual (VM), podendo ainda efetuar pedidos para aumentar a Largura de Banda, número de sub-redes e endereços IP (*Internet Protocol*) necessários para o correto funcionamento de uma dada aplicação.

A estruturação e controlo deste modelo permitem que o cliente assegure uma segurança extra, personalizada pelo próprio, o que pode ser visto como uma mais valia perante a incógnita da segurança e privacidade providenciada pelos CSP's. Na elaboração da Dissertação, será explorado este conceito de segurança personalizada baseado no contexto da oferta da IPBRICK.

Platform as a Service

A camada intermédia dos modelos de serviço é identificado pelo PaaS, que providencia uma maior abstração ao cliente, embora este seja responsável pela gestão da plataforma de execução de aplicações e serviços. Tal possibilita aos desenvolvedores a possibilidade executar as suas suas soluções numa plataforma da *Cloud*, evitando custos desnecessários com servidores para diferentes objetivos e replicações das ferramentas de desenvolvimento por cada um deles.

O maior nível de abstração deve-se, tal como se pode observar na figura 2.2, à delegação da responsabilidade de tarefas de baixo nível da pilha protocolar para o CSP, pelo que serviços necessários para o desenvolvimento na plataforma já incluem *middleware*, sistemas de gestão de bases de dados, acesso à Rede, armazenamento e ferramentas úteis para o desenvolvimento e *design* da aplicação. Exemplos de serviços PaaS são os bastante conhecidos Microsoft Azure e Google App Engine.

Um CSP que providencia um serviço baseado em PaaS elimina a necessidade de o cliente investir em infraestruturas físicas, necessitando apenas de pagar a infraestrutura virtual fornecida CSP, proporcionando-lhe balanceamento de carga, rápida escabilidade e propagação de alterações, atualizações automáticas, entre outras. Note-se que, tendo em conta que uma atualizações do sistema operativo no qual a plataforma de desenvolvimento está assente pode influenciar as aplicações desenvolvidas, o cliente tem a opção de desabilitar a automaticidade das atualizações.

Software as a Service

Por último, o modelo SaaS é aquele que possui um maior grau de abstração para o cliente e, consequentemente, um controlo reduzido sob as camadas visualizadas na figura 2.2. Por esse motivo, é aquele que o serviço de *Cloud* é mais barato, estando, geralmente, assente numa modalidade de pagamento mensal ou anual. Em suma, o cliente paga o serviço e não produto.

Através de um serviço de *Cloud* baseado em SaaS, o cliente tem acesso a determinadas aplicações e bases de dados, delegando para os CSP's a responsabilidade de gestão de infraestruturas e plataformas onde as aplicações e bases de dados estão assentes. Assim, a necessidade de infraestruturas de suporte, *hardware* e licenciamento é claramente reduzido.

A centralização de *softwares* numa mesma plataforma e a alocação de instâncias dos mesmos através da Internet, facilita a implementação de atualizações, bem como a rápida propagação das mesmas. No entanto, trata-se de um modelo bastante limitado ao nível de permissões e compatibilidades de *software*, uma vez que o cliente está dependente da infraestrutura e compatibilidades das plataformas que o CSP possui.

2.2.4 Modelos de Desenvolvimento

As necessidades de *Clouds* que fossem ao encontro a determinados requisitos dos utilizadores (por exemplo, um maior grau de segurança mas com um sistema com elevada escalabilidade) proporcionou o desenvolvimento de quatro tipos de *Clouds*: pública, privada, híbrida e comunitária.

Em comum, tem o facto de se basearam no conceito de Computação em *Cloud* [3], divergindo, sobretudo, ao nível de custos para o cliente, arquitetura da rede e nível de segurança. Seguidamente, serão exploradas cada ums dos tipos de *Cloud*, cuja relação pode ser observada na figura 2.3.

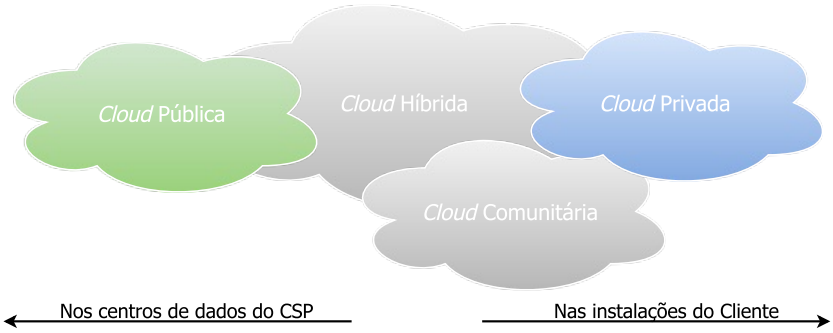


Figura 2.3: Tipos de arquitetura de um serviço de *Cloud*.

Cloud Pública

Uma *Cloud* pública é aquela em que a infraestrutura e os serviços são prestados através da Internet, sobre uma rede que está aberta ao uso público e acessível por qualquer utilizador que pretenda aderir a um serviço de *Cloud* pública. São, por isso, aquelas que tornam um serviço de *Cloud* o mais oposto possível em relação a uma solução tradicional. Além disso, ao contrário das implementações privada e híbrida, uma *Cloud* pública situa-se sempre fora das instalações de um cliente.

Estas *Clouds*, geralmente, apresentam um melhor nível de eficiência de partilha de recursos, uma vez que possuem pessoal mais especializado na sua configuração, e, fornecem os serviços a diversos clientes utilizando a mesma infraestrutura partilhada. Além disso, o facto de se basearam em implementações generalistas, implica que estas já estejam preparadas para o que cliente adira e comece logo a usar as suas funcionalidades. Mesmo em caso de migrações para a *Cloud* a partir de uma solução tradicional ou de um outro serviço de *Cloud*, os CSP's já dispõe de algumas facilidades (geralmente, em forma de tutoriais iniciais) para que o próprio utilizador comum possa ele próprio configurar o serviço.

Atualmente, esta é a implementação de Computação em *Cloud* mais habitual, mas que, a longo prazo, sofrerá uma grande diminuição devido às implementações privadas e híbridas, que contornam melhor o problema da segurança associado a esta tecnologia.

Cloud Privada

Uma *Cloud* privada representa um esforço ao nível da segurança e privacidade em relação às *Clouds* públicas, usando as mesmas tecnologias, contendo um ambiente que apenas clientes específicos conseguem operar. Desta forma, estas *Clouds* oferecem a cada cliente um maior grau de controlo e segurança, o que lhes permite adaptar o seu espaço em *Cloud* às suas especificações e políticas de segurança.

Ao contrário das *Clouds* públicas, as *Clouds* privadas podem operar num ambiente fora das instalações do cliente – na qual o cliente poderá, ou não, compartilhar os recursos existentes com os outros, de uma forma mais segura em relação à *Cloud* pública – ou nas instalações do cliente – semelhante à solução tradicional, mas que engloba um ambiente adequado às caraterísticas da Computação em *Cloud* e com um custo mais elevado em relação às *Clouds* públicas.

Ao nível da segurança, em relação às *Clouds* públicas, uma *Cloud* privada providencia uma separação, física ou virtual, entre os utilizadores envolvidos numa dada *pool* de recursos. Além disso, tal como já foi referido, trata-se de um modelo de desenvolvimento mais ajustável às políticas de uma organização. No entanto, este maior cuidado ao nível da segurança torna este modelo de desenvolvimento mais caro em relação ao modelo público (mesmo para as *Clouds* privadas instaladas fora das instalações do cliente).

Cloud Híbrida

A *Cloud* híbrida representa todo um sistema ágil e interligado, entre diferentes modelos de desenvolvimento, podendo inclusive se incluir o modelo tradicional de TI. Desta forma, uma *Cloud* híbrida pode representar uma combinação de duas, ou mais *Clouds*, podendo cada *Cloud* ter, ou não, o mesmo modelo de desenvolvimento.

Este modelo apresenta o maior *trade-off* entre flexibilidade e custos associados entre os modelos de desenvolvimento apresentados, uma vez que cabe ao cliente decidir a forma como quer dispor os seus recursos (por exemplo, poderá ser do interesse do cliente manter recursos e dados confidenciais e/ou com elevada criticidade nas suas próprias instalações), beneficiando assim da diminuição da carga de trabalhos com aqueles que serão dispostos num dado CSP.

A interligação entre diferentes modelos de desenvolvimento é feita com recurso a *standards* e/ou API's proprietárias (associado ao CSP dessa *Cloud*), que possibilita a portabilidade de dados e aplicações. Na secção 2.2.7 serão abordados algumas das propostas de *standards* existentes.

Cloud Comunitária

Por último, uma *Cloud* comunitária representa um espaço comum entre várias organizações que suporta uma dada comunidade com preocupações comuns, como os objetivos, requisitos de segurança e políticas da organização[3]. À semelhança das *Clouds* privadas, esta pode ser gerida por terceiros ou por uma organização, podendo estar instalada na localização de uma organização ou externamente a esta.

Desta forma, pode-se entender este modelo de desenvolvimento como uma mescla dos modelos público e privado, providenciado uma redução de custos, mantendo um bom nível de segurança e privacidade, mas partilhando recursos computacionais com as organizações envolvidas nessa comunidade.

2.2.5 Ótica de Negócio

A evolução do serviço de *Cloud* tem permitido a que as empresas fixem-se cada vez mais no seu *core business*, uma vez que as preocupações ao nível da infraestrutura de TI, *hardware* e

licenciamento de *software* é largamente minimizado. Além disso, num mundo empresarial onde o acesso a informação em qualquer lugar e a qualquer momento é visto como um objetivo fulcral, a utilização da Internet como meio de transporte entre o utilizador e o seu espaço na *Cloud* permite atingir esse objetivo.

Deste modo, as empresas olham para um serviço de *Cloud* como uma alternativa que lhes irá permitir uma redução drástica no TCO, sobretudo devido à maioria dos custos de compra, operação e manutenção da infraestrutura TI e do *hardware* serem da responsabilidade do CSP. Seguidamente, são visualizadas as caraterísticas de um serviço de *Cloud* que tem impacto na infraestrutura económica e técnica de uma empresa.

Infraestruturas TI e Licenciamento. Um CSP que seja responsável por fornecer grande parte dos serviços intrínsecos a uma empresa – como os serviços de e-mail, armazenamento ou ferramentas de edição de documentos – minimiza a necessidade da existência de infraestruturas e *hardware* responsável por suportar os servidores dos serviços que estarão na *Cloud*. Além disso, a necessidade de aquisição de licenças por cada utilizador é igualmente minimizada, uma vez que existem *softwares* que já estão incluídos no serviço de *Cloud*. Note-se que, em relação às infraestruturas e *hardware*, a sua existência *on-premises* devido à necessidade da existência de serviços de proteção local, tais como serviços de anti-vírus e anti-spam, como também de servidores DNS, responsáveis por fazer o mapeamento da rede local.

Gestão e Manutenção. Dada a delegação da maioria da infraestrutura TI e *hardware* para o CSP, o cliente consegue minimizar os custos de manutenção e operação dos servidores existentes *on-premises*. Além disso, cada componente de uma infraestrutura IT – seja ela uma componente da rede local, como um *switch*, um *router* ou um computador, onde estejam alojados um ou mais servidores – possui um ciclo de vida, pelo que, periodicamente, a empresa necessita de custos elevados na compra de novos componentes. Desta forma, com um serviço de *Cloud*, apenas os recursos mantidos localmente necessitam desses cuidados.

Redundância e Garantia de Serviço. Um serviço de *Cloud*, geralmente, apresenta uma implementação redundante que assegura a integridade de dados dos clientes e, consoante o nível de espalhamento dos centros de dados no globo, proporciona que os atrasos na obtenção de pedidos sejam menores. Além disso, está igualmente assegurado a segurança física contra falhas de eletricidade, através de sistemas como *Uninterruptible Power Supplies* (UPS).

Produtividade. Um CSP pode providenciar *softwares* que possibilitam a edição partilhada de documentos e, desta forma, proporciona um elevado impacto na elaboração de determinadas tarefas associados ao *core business* de uma empresa. Além disso, tal como referido no ponto anterior, a existência de sistemas que asseguram a disponibilidade da *Cloud* (por exemplos, as UPS) também contribuem para que a produtividade não seja afetada.

Facilidade de Inovação. A escalabilidade introduzida por um serviço de *Cloud* possibilita que um cliente possa gerir o seu negócio, manobrando facilmente os recursos que necessita, o que possibilita uma maior facilidade de inovação, com custos bem mais reduzidos. Além disso, a centralização de um *software* num CSP proporciona a facilidade de atualizações do mesmo, também proporcionando um melhor desempenho na produtividade e viabilizando a inovação.

No entanto, além das caraterísticas supracitadas, o cliente deve avaliar previamente a adequação do seu negócio a um serviço de *Cloud*, projetando qual o tipo de serviço mais adequado. Segundo [6], devem ser tidas em conta as seguintes caraterísticas:

- existência de período de experimentação do serviço de *Cloud*, por forma a que o cliente consiga ter uma visão mais concreta do serviço que está a contratar;
- grau de segurança e privacidade de recursos do cliente: se a perda e/ou exposição de um determinado recurso for vista como um elevado risco para o bom funcionamento da organização, o cliente deve optar bem se pretende correr o risco de o colocar na *Cloud*;
- existência de serviços e *softwares* cruciais: apesar da alta disponibilidade da *Cloud*, a impossibilidade do CSP a comprovar eficazmente deve pesar na avaliação de o cliente passar a usar o serviço/*software online*;
- desenvolvimento e testes: para uma empresa cujo *core business* passa pelo desenvolvimento de *software*, existe a necessidade de existirem servidores dedicados para produção, desenvolvimento e testes, que possuem as mesmas ferramentas de desenvolvimento. Com um ambiente de desenvolvimento na *Cloud*, tal pode ser aglutinado num só ambiente, facilitando a vida aos desenvolvedores e, claro, à economia da empresa. No entanto, a criticidade dos resultados do desenvolvimento deve ser tido em conta, uma vez que estará alojado na *Cloud*.

2.2.6 Tecnologias Relevantes

A implementação de um serviço de *Cloud* engloba uma implementação e gestão lógica que engloba muitos componentes por forma a cumprir os requisitos definidos para o conceito de Computação em *Cloud*. Seguidamente, por forma a se perceber quais as componentes e tecnologias relevantes no *back-end* de um serviço *Cloud*, serão apresentadas cada uma delas.

Infraestrutura da Rede. No centro de dados de um CSP, a infraestrutura da rede deve estar configurada por forma a que as caraterísticas de Computação em *Cloud* sejam passíveis de se aplicar. Assim, a existência de componentes como balanceadores de cargas, *firewalls*, *routers* e *switches* torna-se essencial para providenciar o acesso correto de cada utilizador à sua área pessoal na *Cloud*, mantendo a essência de um serviço de *Cloud*. Além disso, a própria configuração da rede deve ser tal que evite a quebra de serviço através de uma estrutura redundante nas ligações entre componentes. Por último, deve ser implementada tal que permita o monitoramento e gestão dos acessos ao sistema [7][8].

Infraestrutura do Centro de Dados. A provisão de segurança no centro de dados dos CSP's deve ter em conta dois níveis de segurança: físico e lógico. A segurança física implica **(i)** controlo do ambiente físico, nomeadamente a prevenção e monitorização contra sismos e controlo de humidade e temperatura nas salas onde a infraestruturas se encontra instalada, **(ii)** prevenção contra falhas elétricas, através de UPS's e geradores elétricos de recurso para todos os sistemas, e **(iii)**

controlo de acessos, através de uma monitorização contínua através de câmaras e seguranças, e de acesso limitado a utilizadores autorizados [9][10].

Virtualização. Segundo [11], a virtualização diz respeito à criação de uma versão virtual de um dado sistema, tal como um *hardware*, sistema operativo, dispositivo de armazenamento ou recursos de uma rede. Desta forma, é providenciada uma maior flexibilidade operacional e aumento a taxa de utilização dos recursos físicos disponíveis, abstraindo os utilizadores da natureza física dos recursos. Num serviço de *Cloud*, não se pode falar do mesmo sem a existência de um sistema baseado em virtualização, enquanto o inverso é passível de se dizer. A virtualização pode ser dividida em diversos tipos, tais como **(a)** virtualização do servidor, que possibilita a existência de múltiplas instâncias servidoras (servidores virtuais) num dado servidor físico, **(b)** virtualização de redes, que proporciona o isolamento e segmentação de determinadas redes físicas, **(c)** virtualização do armazenamento, proporcionando a aglutinação de múltiplos recursos de armazenamento físicos num só, aumentando assim a capacidade de provisionamento, **(d)** virtualização do bloco de armazenamento, utilizada na virtualização do armazenamento, possibilitando a manutenção de aplicações *online* enquando os dados são transferidos, **(e)** e virtualização de ficheiros, possibilitando a transferência de dados sem interrupção da aplicação. Desta forma, o impacto da virtualização verifica-se na otimização dos recursos físicos existentes, maximizando a sua utilização, enquanto proporciona uma redução de custos. Note-se, por último, na figura 2.4 os tipos de um Gestor de Máquinas Virtuais (VMM), *Hosted* e *Bare-Metal*, cuja principal diferença baseia-se, respetivamente, na existência, ou não, de um sistema operativo no sistema físico que serve como *host*.

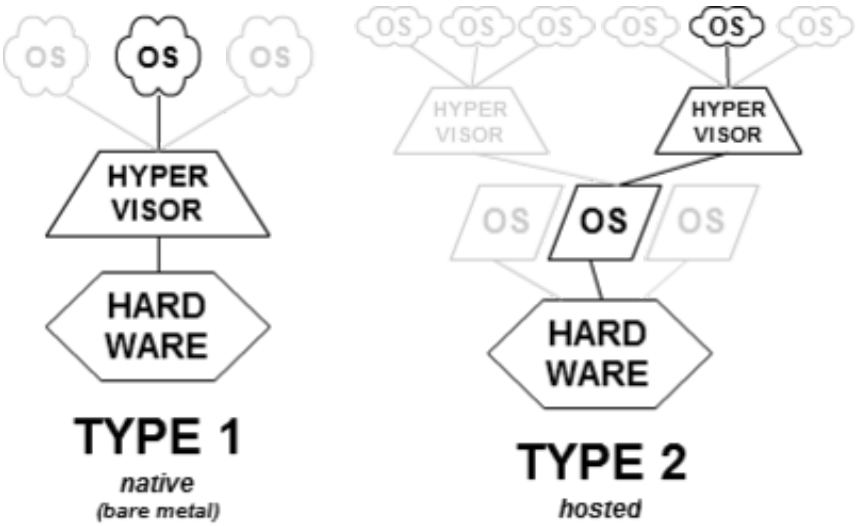


Figura 2.4: Tipos *Hosted* e *Bare Metal* do gestor de Máquinas Virtuais.

2.2.7 Normalização na *Cloud*

A evolução dos serviços de *Cloud* tem implicado uma constante refinação nas comunicação entre diferentes *Clouds*, bem como ao nível da segurança da *Cloud*. Por forma a que se obtenha tais objetivos, é necessário a definição de normas gerais de implementação de um serviço de *Cloud*. No entanto, atualmente, ainda não existem quaisquer *standards* globalmente aceites e que provoquem a implementação de serviços de *Cloud* de cada CSP similarmente, recorrendo-se regularmente a propostas de *standards* para determinadas funcionalidades ou a API's específicas de um CSP.

Segundo [3], pode definir-se três áreas que o desenvolvimento de *standards* deve cobrir: interoperabilidade entre serviços de *Cloud*, portabilidade de dados e segurança da *Cloud*. Seguidamente, serão exploradas cada uma delas.

Interoperabilidade entre serviços de *Cloud*. Esta propriedade permite a interligação entre diferentes CSP's sem que as suas aplicações e dados estejam em risco. No entanto, atualmente, existe uma carência de *standards* que possibilitem implementações similares entre diferentes CSP's, sendo apenas disponibilizadas API's específicas a cada CSP que induzem o utilizador a um maior grau de dificuldade na interligação. Por forma a contornar esse problema, algumas organizações estão já a cooperar na criação de *standards* para determinados serviços[12][13], nomeadamente o **(a)** *Open Cloud Computing Interface*[14] (OCCI), desenvolvido pela Open Grid Forum[15], **(b)** *Cloud Data Management Interface*[16] (CDMI), desenvolvido pela Storage Networking Industry Association[17] e **(c)** *IEEE P2301*[18] e IEEE P2302[19], desenvolvidos pelo IEEE Standards[20], são as principais iniciativas de estandardização para a interoperabilidade de *Clouds*.

Portabilidade de Dados. Esta propriedade permite a transferência/migração de dados e/ou sistemas entre diferentes sistemas (sejam, ou não, baseados em Computação em *Cloud*), utilizando alguns dos *standards* descritos no parágrafo anterior na seguinte medida: para portabilidade de dados, é utilizado o CDMI, enquanto para portabilidade de sistemas, é utilizado o IEEE P2301. Para além destes, para a portabilidade de sistemas, é utilizado o *Open Virtualization Format*[21] (OVF), desenvolvido pela Distributed Management Task Force[22].

Segurança da *Cloud*. Uma das organizações sem fins lucrativos mais importantes no desenvolvimento de *standards* a nível da segurança é a Cloud Security Alliance (CSA)[23]. Esta procura normalizar as implementações e políticas de segurança que respeitam algumas das recomendações definidas em [24], na qual podem ser consultados alguns dos *standards* já existentes.

2.3 Segurança na *Cloud*

A segurança nas *Clouds* é a principal incógnita quanto ao serviço de *Cloud*. Tal deve-se a duas componentes inerentes a um serviço de *Cloud*, nomeadamente **(i)** a utilização da Internet como meio de transporte na comunicação entre centro de dados e cliente e **(b)** a incapacidade de demonstrar ao cliente a segurança e privacidade dos seus dados, mesmo tendo em conta o SLA estabelecido.

Nas secções 2.3.1 e 2.3.2 são apresentadas as ameaças e vulnerabilidades existentes num serviço de *Cloud* e tecnologias de proteção E/S, respetivamente. Por último, na secção 2.3.3 será feita uma apresentação e análise das principais ferramentas de IDS e IPS, gratuitas e de código aberto, existentes no mercado.

2.3.1 Ameaças e Vulnerabilidades

Um solução *Cloud* está exposta a diversas ameaças, geralmente resultantes da existência de vulnerabilidades num serviço englobado na solução. Deste modo, podemos definir os conceitos de *ameaça* e *vulnerabilidade* como o seguinte:

- **Ameaça**: geralmente, representa algo que não pode ser controlado, explorando as vulnerabilidades de um sistema, podendo resultar na danificação ou destruição de dados ou do bom funcionamento de um sistema;
- **Vulnerabilidade**: ao contrário da ameaça, representa que algo que pode ser controlado e solucionado. Trata-se de uma debilidade existente num dado sistema que possibilita a sua exploração por atacantes ou, simplesmente, por negligência do próprio utilizador.

Na tabela 2.1 pode-se observar ameaças e vulnerabilidades associadas a um serviço de *Cloud*, bem como uma breve descrição para cada uma, consultadas em [25][26]. Uma das principais ameaças é a necessidade de uma ligação à Internet, algo que, caso falhe no centro de dados do CSP, impossibilita a ligação do cliente ao seu espaço na *Cloud*.

Tipo	Designação	Descrição
Ameaça	Violação de dados	Possibilidade de acesso de uma máquina virtual a outra, pertencente ao mesmo *host*, sobretudo quando os dados envolvidos são de clientes diferentes
Ameaça	Perda de dados	Inexistência de serviços de *backup* de dados, prevenindo a perda definitiva por neglicência do CSP ou devido a atacantes
Ameaça	Invasão da conta	Obtenção de acesso à conta de um cliente do serviço de *Cloud* por parte de um atacante
Ameaça	API's inseguras	Aproveitadas por atacantes para atacar a confidencialidade e integridade dos dados de utilizadores
Ameaça	Bloqueio do Serviço	Inexistência de mecanismos que evitem geração de grande tráfego devido a pedidos aleatórios de um dado atacante
Ameaça	Intrusos maliciosos	Pessoal com acesso autorizado ao centro de dados pode ter acessos a dados de um cliente

Ameaça	Problema de partilha da tecnologia	Má gestão da caraterística *multitenancy* poderá possibilitar acesso indevido de um cliente a dados de outros
Ameaça	Roubo de identidade	Personificação de um dado cliente por parte de um atacante que possui informação acerca do mesmo devido à não existência de mecanismos de certificação de identidade
Ameaça	Perfis de ataque desconhecidos	Possibilidade de existência de diversos ataques ainda não identificados e que, por isso, não podem ser reconhecidos numa primeira instância
Vulnerabilidade	Aproveitamento e Roubo de sessão	Uso de uma chave de sessão válida ou de um *cookie* roubado para obter acesso não autorizado a informação ou a um sistema ou a um servidor remoto, respetivamente
Vulnerabilidade	Encriptação insegura	Aplicação de métodos que possibilitam a desencriptação indesejada do conteúdo, tal como ataques de força-bruta, de tempo ou de dicionário
Vulnerabilidade	Proteção e Portabilidade de Dados	Existência de debilidades nos SLA's estabelecidos entre cliente e CSP (por exemplo, situação dos dados enquanto um contrato expirado não é renovado)
Vulnerabilidade	Ataque à máquina virtual	Existência de vulnerabilidades numa VM pode facilitar o acesso a máquinas virtuais
Vulnerabilidade	Ataque de injeção de *malware*	Inexistência de ferramentas de deteção de *malware* poderá permitir o *upload* de *malware* para a *Cloud* e, desta forma, afetaro bom funcionamento da mesma
Vulnerabilidade	Aprisionamento ao CSP	Contrato com CSP's que dificultam a migração para outros CSP's ou na interligação com um diferente CSP

Tabela 2.1: Vulnerabilidades e ameaças passíveis de existir num serviço de *Cloud*.

Note-se que, em [26], são definidos alguns ataques que resultam de ameaças definidas na tabela 2.1. Ao longo do desenvolvimento da Dissertação, a solução implementada passará pela resolução de alguns dos problemas referidos, de forma pró-ativa ou apenas com recurso a alertas. Os principais problemas visados para resolução são **(i)** violação de dados, **(ii)** invasão de conta, **(iii)** bloqueio do serviço, **(iv)** ataques de injeção de *malware*, **(v)** perfis de ataque desconhecidos.

2.3.2 Soluções de Segurança do Perímetro

A segurança de um espaço em *Cloud* pode ser vista como um modelo de duas camadas, similar ao que existe numa intranet[27]: uma camada exterior, assente nos protocolos de comunicação e nos diversos campos de um datagrama trocado numa comunicação, cujos constituintes variam consoante a camada da pilha protocolar TCP/IP; e uma camada interna, responsável pela proteção dos dados transmitidos, e que varia consoante um conjunto diverso de algoritmos de encriptação.

A camada exterior, na qual esta Dissertação tem o seu principal foco de incidência, possibilita por isso o controlo de acessos ao espaço em *Cloud* do cliente, através de regras e/ou algoritmos de aprendizagem que têm como parâmetros campos caraterísticos dos datagramas trocados na comunicação (por exemplo, os endereços IP de origem e destino). Ao longo desta secção, serão apresentadas duas ferramentas que possibilitam a gestão da segurança a nível da camada externa referida, nomeadamente as *firewalls* e os sistemas de IDS/IPS.

2.3.2.1 *Firewall*

Uma *firewall* é um sistema de segurança da rede, podendo ser baseado num *hardware* ou *software*, que controla o tráfego de entrada e saída da rede, segundo um determinado conjunto de regras. Esta atua como um obstáculo entre uma rede (ou sistema) segura e outras inseguras (como a Internet), permitindo ou bloqueando o acesso aos recursos da mesma[28]. A existência de uma *firewall* não implica, necessariamente, que **(i)** a rede ou sistema se encontra totalmente protegido, devendo ser acompanhada com outros mecanismos de segurança, e que **(ii)** esta não necessite de configurações, pois uma má configuração da *firewall* poderá trazer piores consequências em relação àquelas que a não-existência de uma *firewall* implica.

Segundo [29], uma *firewall* pode englobar um dado conjunto de tecnologias (por exemplo, filtragem de pacotes ou inspeção do estado de uma ligação), e que possibilitam caraterizar quais as camadas da pilha protocolar TCP/IP[30] que o raio de ação da *firewall* engloba. Na tabela 2.2 podem-se observar as quatro camadas do modelo TCP/IP, com a descrição das caraterísticas que um tipo de *firewall* possui para ser contemplar funcionalidades dessa camada. Note-se que, consoante as caraterísticas de uma *firewall*, esta poderá ser colocada num dispositivo de encaminhamento da rede (como um *router*) ou num dispositivo terminal (como uma estação de trabalho).

A IPtables[31] é um exemplo de uma *firewall* cujo raio de ação assenta nas quatro camadas do modelo TCP/IP. Esta é parte constituinte do módulo de segurança *Netfilter*, que fornece funcionalidades de *firewall* e de tradução de endereços de rede (geralmente, designado como NAT) ao nível do *kernel* Linux, sendo atualmente parte integrante de qualquer sistema Linux. Por conseguinte, a IPtables possibilita a manipulação e filtagem de tráfego, qualquer que seja a camada referida na tabela 2.2, com um fluxo de execução constituído por diferentes tabelas e *chains*, que se podem observar na figura 2.5.

A IPtables é constituído por diversas componentes[32], nomeadamente **(i)** cinco tabelas, consultadas consoante o pacote de entrada, **(ii)** cinco *chains*, constituintes de uma tabela, sendo passível a criação de novas *chains*, **(iii)** ações, que determinam o destino de um pacote, e **(iv)** outros

Camada	Caraterísticas da *Firewall*
Acesso à Rede	Decisões baseadas no estado de saúde dos dispositivos que pretendem aceder à rede, mesmo que autorizados.
Internet	Decisões baseadas nos endereços IP de origem e/ou destino, por cada pacote. Pode ser *stateful* ou *stateless*, usando, ou não, informações de sessões ativas e/ou recentes, respetivamente.
Transporte	Decisões baseadas nos protocolos de transporte envolvidos, mas também das portas utilizadas e do estado da ligação, por cada pacote.
Aplicação	Decisões baseadas nos protocolos de aplicação envolvidas, mas também de sistemas, locais ou remotos (como *proxies*), que façam a seleção do tráfego.

Tabela 2.2: Caraterísticas de um tipo de *firewall*, por camada do modelo TCP/IP.

parâmetros, caraterizadores específicos para um dado pacote, sendo que estes possuem um caráter opcional consoante a tabela em questão. Quanto às tabelas, a IPtables não possibilita a sua modificação e/ou alteração, sendo estas caraterizadas da seguinte forma:

- tabela *filter*: responsável por filtragens de pacotes, esta tabela é utilizada por defeito e é constituída pelas *chains* INPUT, FORWARD e OUTPUT;
- tabela *nat*: responsável pela tradução de endereços ou portas, esta tabela é geralmente utilizada para reencaminhamentos internos ou quando o sistema é utilizado como um *router*, sendo constituída por todas as *chains* existentes: PREROUTING, INPUT, FORWARD, OUTPUT e POSTROUTING;
- tabela *mangle*: responsável por efetuar alterações especiais em pacotes, esta tabela é constituída pelo mesmo conjunto de *chains* da tabela *nat*;
- tabela *raw*: responsável pela definição de exceções, esta tabela não é, geralmente, muito utilizada, e é constituída pelas *chains* PREROUTING e OUTPUT;
- tabela *security*: responsável pelas regras de Acesso de Controlo Obrigatório (em inglês, *Mandatory Access Control*, implementado pelos módulos de segurança Linux, tal como o SELinux), esta tabela é constituída pelas *chains* INPUT, FORWARD e OUTPUT.

Por outro lado, ao nível das *chains*, a IPtables apenas possibilita a modificação e eliminação de *chains* adicionadas pelo utilizador, sendo que as *chains* existentes, por defeito, podem ser caraterizadas por:

- **PREROUTING**: verificação das políticas de segurança no pacote antes de decisão de encaminhamento ser feita;
- **INPUT**: o pacote é entregue localmente, sendo o controlo desta feito pela tabela de encaminhamento;
- **FORWARD**: verificação de todos os pacotes com encaminhamento definido e que não são para entrega local, passam neste modo;

- **OUTPUT**: pacotes que são enviados da própria máquina passam neste modo;
- **POSTROUTING**: verificação de políticas de segurança, antes da passagem para o *hardware* e após a decisão de encaminhamento estar definida.

Por último, as ações mais utilizadas referem-se à passagem ou bloqueio de pacotes, mudança de *chain* ou redirecionamento de decisão para aplicações executadas no *userspace*.

Note-se que, ao nível da prioridade de regras, a IPtables utiliza a ordem que as regras estão dispostas para efetuar a verificação de cada pacote: assim sendo, um pacote percorre as várias regras existentes numa *chain* até que esta encontra correspondência com uma regra existente. Caso tal não se verifique, é aplicada a política de defeito configurada para a tabela em questão.

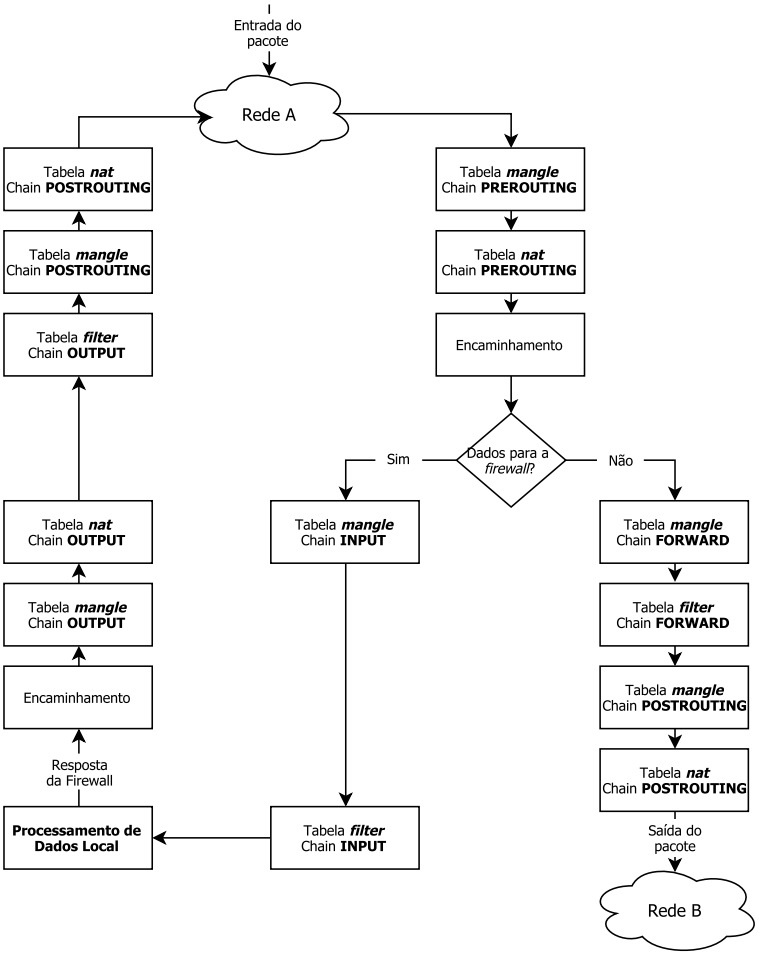


Figura 2.5: Fluxo de execução da *firewall* IPtables, para cada novo pacote de entrada.

2.3.2.2 Sistemas de IDS/IPS

A utilização de sistemas de IDS e IPS tem como finalidade a deteção e prevenção de intrusões em tempo real, respetivamente.

Segundo [33], os sistemas de IDS e IPS distinguem-se pelo tipo de reação a um possível ataque ou vulnerabilidade: enquanto o IDS trata-se de um sistema reativo, que gera alertas relativamente a possíveis falhas na segurança, um sistema de IPS é um sistema pró-ativo, que age com determinadas políticas na existência de uma possibilidade falha de segurança. Por esse motivo, um sistema de IDS é visto apenas como uma ferramenta de análise da rede ou do sistema, necessitando que o utilizador efetue ações com base nos alertas gerados, enquanto um sistema de IPS possui mecanismos automáticos que elimina a necessidade do utilizador executar determinadas ações de segurança.

Os sistemas de IDS e IPS podem ser classificado com base em três critérios: a fonte de eventos (isto é, a localização dos sensores de monitorização), o método de deteção e a resposta. A fonte de eventos pode ser **(a)** as interfaces de rede, baseando-se nos pacotes que fluem no tráfego da rede, sendo estes sistemas designados como NIDS/NIPS (*network-based* IDS/IPS), e **(b)** a informação relativo de um sistema, como os *logs* e os ficheiros críticos de um sistema, sendo estes sistemas designados como HIDS/HIPS (*host-based* IDS/IPS). Por outro lado, os métodos de deteção podem ser baseados em **(a)** padrões de erro, sendo baseado em valores estáticos de campos de um pacote, **(b)** deteção de anomalias, geralmente implementado por algoritmos de *data mining* ou *machine learning* e, por isso, pode possuir uma elevada taxa de falsos alarmes, e **(c)** correlação de eventos, geralmente aplicado num sistema com múltiplos pontos de deteção, e baseia-se nos vários eventos gerados em cada ponto. Por último, a resposta pode ser apenas reativa, através da geração e registo de alertas, ou pró-ativa, agindo com políticas predefinidas.

Um sistema de IPS possui um comportamento equiparado a uma *firewall*, funcionando como uma ferramenta de controlo de tráfego, utilizado para executar determinadas funções na iminência de situações de risco[33]. Executando um sistema de IPS em conjunto com a *firewall*, é possível aumentar a segurança de um sistema, na medida em que o IPS possibilita análise a campos de um datagrama que uma *firewall* não consegue fazer, mas também pelas suas caraterísticas relativos ao método de deteção implementado. Contudo, pela possibilidade de existência de falsos positivos, um sistema de IPS pode representar a perda de acesso a um dispositivo e/ou rede na qual este se encontre a monitorizar.

Na secção 2.3.3 serão expostas e analisadas as ferramentas de IDS e IPS (de código aberto e gratuitas) mais utilizadas no mercado.

2.3.3 Ferramentas de IDS/IPS Existentes

Existem diversas ferramentas de IDS e IPS de código aberto disponíveis, mas, devido à necessidade de aplicar estas para fins comerciais, apenas serão consideradas aquelas que gratuitas e passíveis de serem aplicáveis para fins comerciais, sem que exista a necessidade de exercer algum pagamento por elas. Além disso, por forma a focar nas principais ferramentas existentes no mercado, as ferramentas apresentadas serão o Snort[34], OSSEC[35], Suricata[36] e Bro[37]. Posteriormente, no decurso da Dissertação, serão efetuados testes com cada uma delas por forma a selecionar quais serão as ferramentas de IDS e IPS a integrar na solução proposta.

2.3.3.1 Snort

A ferramenta de IDS mais utilizada globalmente (e que, atualmente, já possui funcionalidades de IPS), desenvolvida pela Sourcefire, em 1998. A sua análise baseia-se no tráfego da rede e no *logging* de pacotes em tempo real, podendo ser executado nos principais sistemas operativos existentes (por exemplo, Linux, Windows e MacOS). A sua metodologia baseia-se num mecanismo de deteção que utiliza um *plug-in* de arquitetura por módulos e numa linguagem flexível de descrição de regras para definir o tráfego a ser capturado.

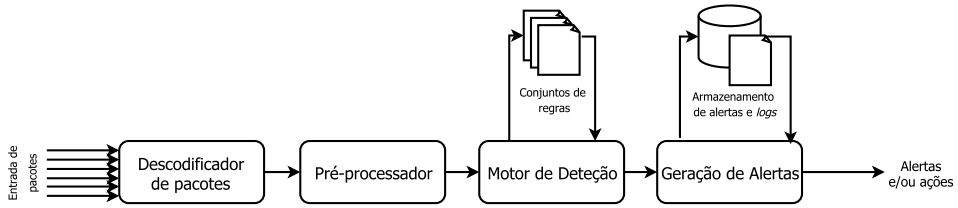


Figura 2.6: Arquitetura modular do funcionamento do Snort.

O Snort usa um mecanismo de *thread* única, pelo que apenas utiliza os recursos de um único processador. No entanto, esta arquitetura está aparentemente ultrapassada, considerando que a existência de *hardwares* com múltiplos CPU's e *cores* é cada vez mais comum. No entanto, por forma a contornar esta limitação, o Snort pode ter em execução múltiplas instâncias, cada uma utilizando um processador diferente[38], com uma mecanismo de execução semelhante ao da figura 2.6.

2.3.3.2 Suricata

Uma ferramenta bastante recente e que, por isso, já se encontra bem preparada para funcionalidades de IDS e IPS. Desenvolvida pelo Open Information Security Foundation (2010), trata-se, atualmente, do principal rival do Snort. Similarmente a este, trata-se de uma ferramenta baseada na análise do tráfego da rede e do *logging* gerado no dispositivo em que se encontra instalada, possuindo um leque de compatibilidades bastante alargado e com modos de operação idênticos.

O fluxo geral dos dados segue a mesma linha do Snort, sendo os pacotes capturados utilizados como objetos de descodificação, processamento e análise, respetivamente. No entanto, ao contrário do Snort, o Suricata utiliza uma metodologia baseada em múltiplas *threads*, sendo que cada uma pode usar um ou mais módulos[38] – ver figura 2.7.

2.3.3.3 OSSEC

Trata-se de uma ferramenta de IDS, desenvolvida por Daniel B. Cid (publicada em 2004), cuja análise se baseia no tráfego interno de um *host*. Tal como o Snort e o Suricata, o OSSEC utiliza padrões de erro (vulgarmente designados por *assinaturas*) como método de deteção. No entanto, esta ferramenta é baseada no *Host*, utilizando os ficheiros e *logs* do sistema na deteção de falhas de segurança.

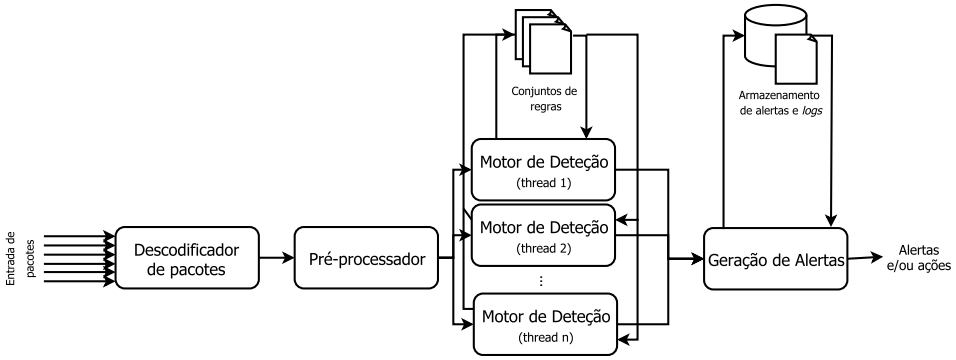


Figura 2.7: Arquitetura modular do funcionamento do Suricata.

O OSSEC possui dois tipos de arquitetura: cliente-servidor, na qual um servidor é responsável pelo tratamento dos *logs* coletados em cada agente/cliente, o que favorece a escalabilidade de uma solução, propagando mais facilmente possíveis alterações nesta ferramenta; e local, cujo procedimento se baseia no que está representado na figura 2.8.

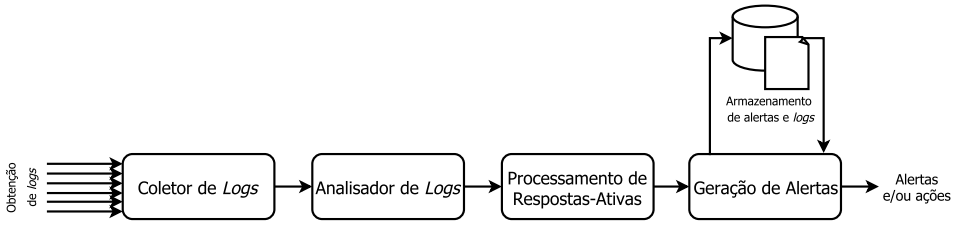


Figura 2.8: Arquitetura modular do funcionamento do OSSEC.

2.3.3.4 Bro

Por último, esta ferramenta de IDS foi desenvolvida por Vern Paxson, encontrando-se agora em constante desenvolvimento pelo mesmo e por uma equipa de investigadores e desenvolvedores. Trata-se de uma ferramenta baseada na segurança da rede, mas que proporciona igualmente uma plataforma para uma análise geral do tráfego na rede.

Além de a nível de licenciamento, possuir um conjunto de políticas menos restrito que o Snort e o Suricata, esta foi desenvolvida numa linguagem única, não providenciando um deteção de assinaturas tradicional. Possibilita mecanismos de computação paralela, providenciado a respetiva análise de rede enquanto efetua a deteção de anomalias no sistema.

Figura 2.9: Arquitetura modular do funcionamento do Bro.

Apesar de não possibilitar uma implementação de funcionalidades de IPS, o Bro possibilita tornar aquilo que um sistema de IDS, por defeito, não é: pró-ativo. Além disso, possui uma ferramenta denominada BroControl, que permite ao administrador a manipulação de múltiplas instância do Bro ao mesmo tempo. Similarmente ao Snort, funciona num mecanismo de *thread* única[38].

2.3.3.5 Comparação das ferramentas

Na tabela 2.3 pode-se observar a comparação das ferramentas exploradas ao longo desta secção. Além disso, em [39] e [40] pode-se visualizar análises técnicas e com recurso a comparações de recursos computacionais entre o Bro e o Snort. Em [41] e [42], são efetuadas análises semelhantes, mas agora com o Suricata e o Snort.

Posteriormente, ao longo da Dissertação, será efetuada uma análise de cada uma das ferramentas com recursos a testes efetuados.

Caraterísticas	**OSSEC**	**Snort**	**Suricata**	**Bro**
Fonte de Eventos	*Host*	Rede		
Método de Deteção	Padrões de Erro			Deteção de Anomalias
Arquitetura de Gestão	Múltiplas (local e cliente-servidor)	Por defeito, local		
Arquitetura de Processamento	*Thread* única		Múltiplas *threads*	*Thread* única
Suporte p/ IPS	Sim			Não
Logging de Alertas	Ficheiros de texto e base de dados	Ficheiros em formato de texto e *unified2* e integração com Barnyard2		
Licenciamento	GNU GPL v.2			BSD

Tabela 2.3: Comparação das ferramentas de IDS/IPS consideradas.

Note-se que o Bro, por não cumprir um dos requisitos principais desta Dissertação – o suporte de resposta ativa (IPS) –, não vai ser considerado no conjunto de ferramentas alvo de estudo nos Capítulos subsequentes.

2.4 Serviços e aplicações IPBrick

O IPBRICK OS é a plataforma de comunicações para sistemas de servidores corporativos desenvolvido pela IPBrick S.A., que utiliza o mais conhecido *Software Open-Source* para serviços de Intranet, comunicações unificadas e sistemas de segurança. Esta plataforma é baseada nas distribuições Linux Debian, sendo que a IPBrick S.A. baseia as versões *major* do IPBRICK OS nas versões *major* do Linux Debian [2].

A disponibilização do IPBRICK OS em *Cloud* visa que este se torne uma alternativa a soluções corporativas semelhantes, tal como o Microsoft Office 365 e Google Apps for Work, soluções disponibilizadas pela Microsoft e Google, respetivamente. No entanto, a IPBrick S.A. apresenta

como mais valia a utilização de serviços e *softwares* de código-aberto, mas também de serviços e aplicações desenvolvidas pela empresa, o que representa uma elevada minimização no custo da solução oferecida.

A plataforma IPBRICK OS integra três tipos de servidores numa só solução, unificando e simplificando a sua gestão a um único administrador. Os três tipos de servidores são **(a)** servidor de Intranet, **(b)** servidor de Comunicações Unificadas e **(c)** Servidor de Segurança, estando os serviços deste último integrado nos dois primeiros tipos de servidor.

O servidor de Intranet é orientado para os serviços básicos existentes numa solução tradicional, integrando os seguintes servidores e aplicações:

- servidor de *e-mail*, integrando os protocolos relacionados (POP, IMAP e SMTP), mas também ferramentas como agenda e livro de endereços. Este servidor é a base da aplicação *web* Groupware, baseada no *software* de código-aberto Horde;
- servidor de ficheiros, utilizado para o armazenamento de dados, utilizando protocolos como o FTP;
- servidor de domínio e serviços de suporte à rede, utilizando protocolos como o LDAP, DHCP e DNS, mas também contém um servidor Radius e possibilita integração com o *Active Directory*;
- servidores de impressão, de bases de dados e de segurança de dados – este último possibilita serviço de *backup* das diversas áreas de trabalho e contempla um anti-vírus para áreas de trabalho e *e-mail*;
- por último, suporta diversas aplicações de negócio, quer estas sejam desenvolvidas por terceiros (por exemplo, utiliza o Bacula para servidor de *backup* e o Nagios para serviço de monitorização), quer pela IPBrick S.A. (tal como a rede social empresarial, IPBRICK.CAFE, ou o iPortalDoc, orientado para a gestão documental).

Por último, o servidor de Comunicações Unificadas é, tal como a sua designação indica, orientado para os serviços que possibilitam a comunicação numa solução tradicional e integra os seguintes servidores e aplicações:

- contém um servidor de *e-mail*, que atua como *relay*, e serviço de consulta de *e-mail* baseado no *webmail*;
- serviço de tradução de *e-mails* para fax, e vice-versa;
- serviços de telefonia, suportando diversos protocolos inerentes ao serviço VoIP (por exemplo, o protocolo SIP), e que oferece a possibilidade de execução do servidor como um *gateway*, PBX ou uma *proxy* VoIP;
- serviços *web*, incorporando um servidor *web* (sobre HTTP e HTTPS), *webphone* e *proxy* e cache HTTP e FTP;

- sistema de mensagens instantâneas, nomeadamente o WebChat e o servidor de *Instant Messaging*;
- serviços de segurança do servidor e comunicações, nomeadamente a *firewall*, servidor VPN de diferentes protocolos (SSL, IPSec, PPTP e GRE), sistema de IDS, anti-vírus para *e-mail* e *proxy* e anti-spam.

2.5 Conclusões

O conteúdo apresentado neste Capítulo teve como objetivo a exposição de três conteúdos principais, nomeadamente **(i)** a apresentação da arquitetura, caraterísticas e ameaças de segurança num ambiente baseado na Computação em *Cloud*, **(ii)** os serviços e *softwares* utilizados para a assegurar segurança do perímetro e **(iii)** os serviços e aplicações da IPBrick.

Este três conteúdos relacionam-se pela disponibilização do IPBRICK OS na *Cloud*, pelo que se pretende garantir a segurança do mesmo quanto a possíveis ataques maliciosos e negligências de utilizadores autorizados – alguns desses problemas estão contidos na tabela 2.1.

Atualmente, tal não é possível apenas com o recurso às ferramentas de segurança do perímetro referidas na secção 2.3. Por um lado, a *firewall* IPtables é ineficaz na verificação dos campos de um datagrama e possui caraterísticas estáticas, pelo que a prevenção de novos ataques apenas resultam da interação direta com o utilizador. Por outro lado, as ferramentas de IDS/IPS possíveis de integração revelam algumas insuficiências no que respeita aos objetivos desta Dissertação quanto à fonte de eventos – esta baseiam-se apenas numa fonte, o *Host* ou a Rede, e não ambos, o que seria otimamente desejável. Além disso, tendo em conta a sua implementação num ambiente *Cloud*, a elevada utilização de recursos computacionais destas ferramentas pode deteriorar acessos a serviços do servidor.

Assim, no contexto da IPBrick, o projeto inerente a esta Dissertação contempla a otimização destas ferramentas de segurança do perímetro, e introduzir o sistema de IPS – atualmente, o IPBRICK OS apenas se encontra dotado do sistema de IDS. Deste modo, e no contexto das soluções *user-friendly* da IPBrick, a solução desenvolvida deve contemplar **(a)** um módulo de gestão dedicado ao sistema de IDS/IPS e as funcionalidades da *firewall* devem ser implementadas com um caráter mais *user-friendly* e **(b)** a implementação de mecanismos de auto-adaptação aos recursos computacionais do servidor. Posto isto, a solução IPBrick será dotada de uma maior transparência, pela maior gama de funcionalidades de gestão da segurança, mas também melhorada a experiência de utilizador, pela existência de métodos de configuração mais *user-friendly*.

Capítulo 3

Solução Dinâmica para a Segurança em *Cloud*

Neste Capítulo está representada a idealização e especificação da solução para o problema desta Dissertação. Por esse motivo, o Capítulo está dividido em três secções principais, **(a)** a contextualização do Problema inerente ao desenvolvimento da solução, sendo identificadas as várias falhas atuais e os objetivos para a solução, **(b)** o estudo das ferramentas de IDS/IPS abordadas na secção 2.3.2, com a finalidade da seleção da(s) ferramenta(s) a integrar o IPBRICK OS, e **(c)** a solução proposta, caraterizada pela arquitetura e mecanismos de interação, ou não, com o Utilizador final.

3.1 Contextualização do Problema

Esta Dissertação apresenta uma proposta para a solução de dois objetivos principais, os quais dizem respeito às aplicações e serviços IPBrick, que serão executadas num ambiente em *Cloud* e disponibilizados sobre a Internet, e ao dinamismo de recursos na *Cloud*, cuja gestão é interessante pela elevada necessidade de recursos consumidos pelo módulo de IDS/IPS.

Por esse motivo, nas secções 3.1.1 e 3.1.2 são apresentados os problemas e identificados os requisitos principais que o projeto desenvolvido no contexto desta Dissertação tem como objetivo resolver, do ponto de vista do interesse da IPBrick S.A., mas também do cliente.

3.1.1 Impacto nos serviços e aplicações IPBrick

A disponibilização dos diversos serviços e aplicações referidos na secção 2.4 sobre a Internet, ao invés destes apenas estarem contidos numa Intranet, é visto como um catalisador para o aumento de ataques aos mesmos. Além disso, a possibilidade de negligências na utilização e configuração de serviços e aplicações, por parte dos utilizadores, também representa um risco para ao servidor e o seu correto funcionamento (por exemplo, uma má configuração pode representar a perda de acesso ao servidor na *Cloud*).

Por outro lado, o modelo de negócio da IPBrick S.A. não soluciona a ineficácia da garantia de segurança contemplada no SLA estabelecido com o CSP, dado que, tanto a IPBrick S.A., como o cliente não conseguem provas tangíveis sobre a segurança nos centros de dados do CSP, a qualquer momento.

Por esse motivo, a minimização do impacto dos serviços e aplicações IPBrick, bem como a eficácia de garantia de segurança à IPBrick S.A. e ao cliente passa por:

- aprimoramento de módulo de segurança, ao nível da *Firewall* e do sistema de IDS/IPS, cuja gestão é realizada através de uma UI de administração, acessível a partir da *web*;
- otimização do sistema de IDS/IPS, com uma configuração orientada para os serviços e aplicações IPBrick.

Estes dois objetivos possuem a sua arquitetura e delineação de requisitos ao longo deste capítulo, nas secções subsequentes.

3.1.2 Impacto do dinamismo de recursos na *Cloud*

A integração e execução de ferramentas de IDS/IPS num servidor assente num ambiente em *Cloud* apresenta, geralmente, um maior impacto comparativamente à sua execução numa solução tradicional, baseada numa Intranet. Tal deve-se, geralmente, a duas causas principais, nomeadamente **(a)** latências intrínsecas à comunicação com a rede do centro de dados do CSP e **(b)** natureza de um solução *Cloud*, que possui dinamismo de recursos disponíveis, não existindo nenhum conjunto de recursos estritamente definido e dedicado a cada utilizador.

Desta forma, ao longo deste Capítulo, procurou-se efetuar uma análise do impacto das ferramentas de IDS/IPS expostas na secção 2.3.3, por forma a optar pela estratégia de utilização que melhor se adapta e minimiza o impacto do acesso remoto aos serviços e aplicações IPBrick e que será o *core* do *back-end* do projeto desenvolvido.

3.2 Estudo das ferramentas de IDS/IPS

Nesta secção encontram-se expostos os testes de impacto e funcionais realizados às ferramentas referidas na secção 2.3.3. Estes testes tem como objetivos **(a)** a perceção do impacto de cada ferramenta no sistema, consoante diversos ambientes de teste, **(b)** a quantificação da eficácia de deteção de ataques por parte das ferramentas de NIDS/NIPS, utilizada para avaliar o tipo de padrões de erro para o Snort, e **(c)** a identificação da estratégia dos mecanismos de auto-adaptação.

Note-se que os testes realizados encontram-se segundo recomendações de testes referidas em [33], para ferramentas de IDS/IPS cuja fonte de eventos se baseia no *Host* e na Rede.

3.2.1 Métricas de Avaliação

A avaliação das ferramentas de IDS/IPS consideradas englobou um conjunto de diversas métricas, utilizadas consoante a fonte de eventos, mas também a disponibilidade de ferramentas e *frameworks* para testes de penetração da ferramenta de IDS/IPS em avaliação.

Por forma a avaliar o impacto de ferramenta de IDS/IPS no sistema, foram consideradas as seguintes métricas:

- **memória RAM**: transversal a todas as ferramentas de IDS/IPS em teste, esta métrica representa, teoricamente, o grau de manobra que uma ferramenta tem para executar as suas funcionalidades na sua plenitude, e é apresentada em *bytes*.

- **número de *cores* e processadores**: esta métrica é importante pela utilização de ferramentas que utilizam mais que uma *thread*, o que, teoricamente, quanto maior o número de processadores e/ou *cores*, mais beneficia a capacidade de deteção da ferramenta.

- **número de padrões de erro da ferramenta de IDS/IPS**: por último, esta métrica traduz-se numa procura de correspondências mais extensa, pelo que, teoricamente, quanto maior o número de padrões, maior a necessidade de recursos computacionais.

Em relação à memória RAM, quando o sistema está em produção, esta divide-se em quatro tipos de memória distintos, e que serão igualmente tidos em conta na avaliação das ferramentas de IDS/IPS. Os quatro tipos de memória referidos, ambos apresentados em *bytes*, são:

- **memória livre**: este tipo de memória está disponível para a sua utilização por qualquer processo, a qualquer momento;

- **memória usada**: este tipo de memória encontra-se a ser utilizada por processos em execução, podendo englobar a memória em *cache* e em *buffer*. Deste modo, à exceção da quantidade de memória adicionada pelas memórias em *cache* e em *buffer*, esta memória não pode ser alocada a nenhum outro processo;

- **memória *buffer***: este tipo de memória possui um caráter temporário e é utilizada para otimizar a obtenção de recursos, evitando assim operação de entrada e saída que, geralmente, implicam maiores tempos de latência;

- **memória *cache***: este tipo de memória assemelha-se ao conceito geral de *cache*, sendo igualmente utilizada para utilizações futuras.

Por outro lado, ao nível do processador, este é caraterizado por três diferentes parâmetros, apresentados em percentagem quanto ao poder de processamento total: disponível, utilizado pelo sistema e utilizado pelo utilizador.

Por último, na avaliação da eficácia de uma ferramenta de IDS/IPS, foram consideradas as seguintes métricas:

- **grau de deteção**: esta métrica identifica o nível de deteção do ataque em três níveis (total, parcial ou nula), refletindo, de certa forma, a taxa de falsos negativos.
- **tipo de teste de penetração**: esta métrica referencia o tipo de teste de penetração, que procura representar algumas das ameaças e vulnerabilidades representadas na tabela 2.1.

3.2.2 Monitorização do sistema de IDS/IPS no sistema

3.2.2.1 Ambientes de Teste

Na realização dos vários testes ao impacto das ferramentas de IDS/IPS no desempenho do sistema, foi utilizado um servidor virtual, cujas caraterísticas – variáveis e estáticas – são as seguintes:

- memória RAM: esta caraterística será variada ao longo dos testes realizados, possuindo os valores de 1 GB, 2 GB e 4 GB;
- poder de processamento: depende do n.º de *cores* e processadores atribuídos ao servidor virtual e, tal como a memória RAM, será variado ao longo dos testes, possuindo os valores de 1, 2 e 4 processadores/*cores*;
- sistema operativo: IPRICK OS, baseada na distribuição Linux Debian, na versão *wheezy* (7);
- Espaço em Disco: 50 GB.

Além disso, por forma a avaliar o impacto das ferramentas de IDS/IPS consoante diversas configurações da memória, CPU e padrões de erro da ferramenta de IDS/IPS, configurou-se o servidor virtual com as caraterísticas estabelecidas na tabela 3.1.

3.2.2.2 Tipos de Configuração utilizadas

As ferramentas de IDS/IPS consideradas, dado as suas diferentes arquiteturas e tecnologias envolvidas, não possuem uma configuração padrão entre si. Por esse motivo, não é possível uniformizar, por exemplo, a carga que uma dada quantidade de padrões de erros representa no sistema, uma vez que estes possuem configurações orientadas à ferramenta de IDS/IPS em si. Desta forma, apenas se orientaram as ferramentas por forma a terem uma quantidade de padrões semelhantes e, ao nível das suas configurações, foram utilizadas as configurações-base de cada uma delas.

Por outro lado, a ferramenta de HIDS/HIPS, o OSSEC, não foi considerada na realização destes testes de impacto. Tal deve-se à necessidade da sua integração no sistema, uma vez que não existe uma alternativa viável ao nível de ferramentas de IDS/IPS com fonte de eventos no *Host*.

Em suma, por forma a avaliar os possíveis ambientes de utilização no *software* IPBRICK OS, foram configurados os seguintes modos de execução das ferramentas:

- execução da ferramenta de NIDS/NIPS em modo IDS;
- execução da ferramenta de NIDS/NIPS em modo IPS.

Ambientes de teste	**Memória RAM**	**CPU**	**#** Padrões de Erro
		#*Cores* x #Processadores	
A	1024 MB	1 *core* x 1 = 1	∼ 1900
B	1024 MB	1 *core* x 2 = 2	∼ 1900
C	1024 MB	2 *core* x 2 = 4	∼ 1900
D	2048 MB	1 *core* x 1 = 1	∼ 1900
E	2048 MB	1 *core* x 2 = 2	∼ 1900
F	2048 MB	2 *core* x 2 = 4	∼ 1900
G	4096 MB	1 *core* x 1 = 1	∼ 1900
H	4096 MB	1 *core* x 2 = 2	∼ 1900
I	4096 MB	2 *core* x 2 = 4	∼ 1900
J	1024 MB	1 *core* x 1 = 1	∼ 8600
K	1024 MB	1 *core* x 2 = 2	∼ 8600
L	1024 MB	2 *core* x 2 = 4	∼ 8600
M	2048 MB	1 *core* x 1 = 1	∼ 8600
N	2048 MB	1 *core* x 2 = 2	∼ 8600
O	2048 MB	2 *core* x 2 = 4	∼ 8600
P	4096 MB	1 *core* x 1 = 1	∼ 8600
Q	4096 MB	1 *core* x 2 = 2	∼ 8600
R	4096 MB	2 *core* x 2 = 4	∼ 8600

Tabela 3.1: Ambientes de teste configurados para avaliação do desempenho das ferramentas de IDS/IPS consideradas.

3.2.2.3 Exposição de Resultados

Nas tabelas A.2-A.5, incluídas no Anexo A, estão expostos os resultados médios obtidos para o impacto de ferramenta de IDS/IPS no sistema, consoante as métricas de avaliação especificadas na secção 3.2.1, enquanto na tabela A.1 estão expostos os valores de referência – com o sistema sem qualquer ferramenta de IDS/IPS em execução – para os vários ambientes referidos.

Os resultados expostos são relativos às monitorizações do sistema em si e apenas do sistema de IDS/IPS, tendo sido efetuada a monitorização do desempenho num intervalo de 10 minutos, com recolha de leitura a cada 10 segundos. Note-se que, em relação à memória, os valores encontram-se arredondados às unidades, enquanto os resultados relativos ao CPU possui um arredondamento com uma casa decimal.

Assim, tal como se pode observar, qualquer que seja o modo de execução da ferramenta (IDS ou IPS), pode-se concluir três factos quanto às duas ferramentas de NIDS/NIPS testadas:

- quanto ao impacto no processamento, o Suricata apresenta um impacto mais elevado que o Snort, causado pela arquitetura *multithreading*, que faz uma maior utilização dos recursos disponíveis nos processadores existentes;

- quanto à memória, o Snort apresenta uma maior utilização de recursos que o Suricata que, possivelmente, deve-se a uma compensação pela arquitetura *single-threading*, utilizando uma maior quantidade recursos da memória na análise dos pacotes;

- por último, tal como era expectável, o número de padrões de erro em execução tem um impacto negativo na utilização de recursos computacionais – sobretudo, na memória –, pelo que, quanto maior o número de padrões de erro, maior é a utilização de recursos.

Deste modo, pode-se concluir que o Snort é uma ferramenta de IDS/IPS mais orientada para sistemas cujas recursos computacionais disponíveis sejam baixos, enquanto o Suricata é orientada para ambientes de elevado desempenho. Note-se que os valores obtidos dizem respeito a um ambiente de teste semelhante ao que se encontra em laboratório, pelo que é expectável observar-se um aumento da utilização do processamento e memória, por ambas as ferramentas, quando executadas num ambiente real.

Na secção 3.2.4 será realizada o tratamento destes resultados por forma a traduzir os desenvolvimentos que a solução desta Dissertação tem por objetivo.

3.2.3 Eficácia contra Ataques

3.2.3.1 *Framework* Pytbull

O Pytbull[43] é um *framework* que utiliza múltiplas ferramentas de penetração da segurança para efetuar diversas tipos de ataques. Este *framework* é assenta numa arquitetura cliente-servidor (o cliente comporta-se como atacante, enquanto o servidor é o sistema-alvo) e possui determinadas dependências quanto a ferramentas de penetração e testes de vulnerabilidades, maioritariamente satisfeitas pela gama de ferramentas que, por defeito, constituem a distribuição Kali Linux[44]. Por esse motivo, foi utilizado um servidor com o Kali Linux para efetuar os ataques ao servidor-alvo, no qual se encontravam as ferramentas de IDS/IPS em execução.

Posto isso, segue-se um funcionamento bastante simples e automatizado, gerando um relatório final com base em gráficos estatísticos e dados técnicos sobre as respostas do NIDS/NIPS ao pedido do cliente, expostos na secção 3.2.3.2.

Note-se que, este *framework* possui como limitações a sua aplicabilidade somente em ferramentas de IDS/IPS cuja fonte de eventos é a rede e, dentro da gama das ferramentas consideradas, apenas é possível de testar o Snort e o Suricata.

Seguidamente, estão expostos os vários módulos de testes efetuados com o recurso a este *framework*, bem como uma breve descrição do objetivo de cada um deles:

- Mau Tráfego: envio de pacotes não aprovados pelos RFCs para o servidor e perceção da forma como estes são processados;
- Força Bruta: testa a eficácia do servidor em detetar ataques de força-bruta;
- Ataques *Client-side*: utiliza *reverse-shell* para providenciar instruções ao servidor para *download*de ficheiros maliciosos remotos. Testa, por isso, a eficácia e habilidade do servidor conta ataques do lado do cliente;
- Negação do Serviço: tal como se pode induzir, este módulo é responsável por testar a proteção contra tentativas de negação do serviço do servidor;

- Técnicas de Evasão: múltiplas técnicas utilizadas com o intuito de verificar se a ferramenta de IDS/IPS as consegue detetar;
- Pacotes Fragmentados: envio de múltiplos *payloads* fragmentados, cujo objetivo passa por perceber se a ferramenta é capaz de recompôr o *payload* e identificar o ataque;
- Utilização Normal: envio de *payloads* que correspondem a uma utilização regular, por forma a detetar más interpretações do servidor;
- Repetição de *pcaps*: repetição de envio de ficheiro *.pcap*;
- *Shellcodes*: envio de diversos *shellcodes* ao servidor, através do porto 21 (geralmente, caraterizado pelo acesso via FTP), para testar a habilidade do servidor detetar/rejeitar *shellcodes*;
- Teste de Regras: por último, este módulo visa o teste de regras básicas, as quais são, por norma, constituintes do conjunto de regras carregadas na ferramenta de IDS/IPS.

A utilização deste *framework* visa a seleção de tipo de padrões de erro do Snort, *VRT Rules* ou *Emerging Threats*, através de testes aos padrões de erro associados às *VRT Rules* e *Emerging Threats*. Note-se que estes testes apenas visam o Snort devido à incompatibilidade do Suricata em relação às *VRT Rules*. Na secção 3.2.3.2 pode-se observar os resultados para testes realizados no Snort, com variância do tipo de padrões de erro.

3.2.3.2 Exposição de Resultados

Na seleção do tipo de padrões de erro, *VRT* ou *Emerging Threats*, foi configurado um servidor, no qual o Snort se encontrava em execução, com 1024 MB de memória RAM e 1 processador no total.

Os resultados obtidos podem ser observados seguidamente, para os padrões de erro *VRT* e *Emerging Threats*, nas figuras 3.1-3.2 e 3.3-3.4, respetivamente.

Tal como se pode observar, os padrões de erro *VRT*, designados *VRT Rules*, possuem uma maior taxa de fiabilidade: ao nível da deteção total e parcial, estas possuem uma taxa de deteção igual e maior, respetivamente, o que é o desejável; por outro lado, ao nível da incapacidade de deteção, este possui uma taxa mais baixa que os padrões de erro *Emerging Threats*, o que beneficia os padrões de erro *VRT*, dado que se pretende minimizar a não-deteção de ataques por uma ferramenta de IDS/IPS.

Por esse motivo, para utilização como base dos padrões de erro em execução no Snort, os padrões de erro *VRT Rules* serão adotados para utilização nos desenvolvimentos e implementações realizadas.

Note-se que seria desejável a utilização de mais que uma ferramenta que permitisse testar as ferramentas de NIDS/NIPS. No entanto, atualmente, o *Pytbull* é a ferramenta disponível e automatizada, de código aberto, mais completa entre as poucas opções disponíveis.

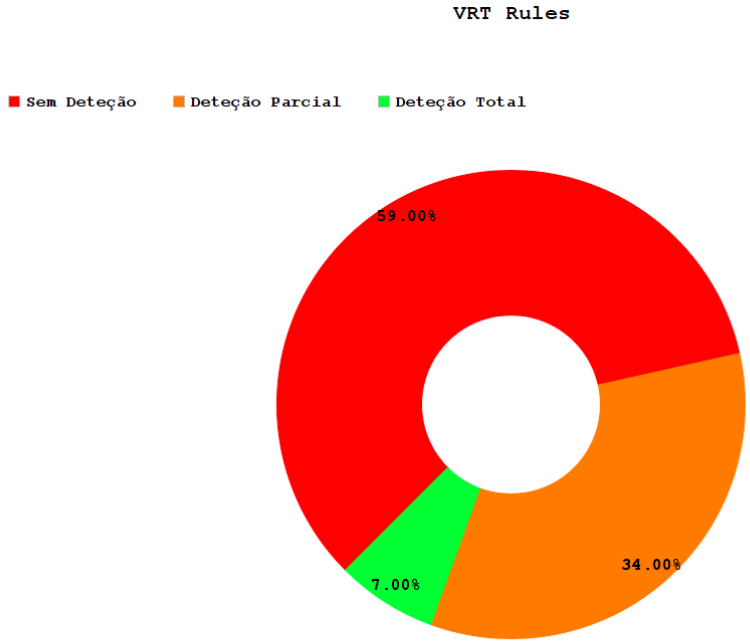


Figura 3.1: Algoritmo de atualização automática da quantidade de padrões de erro (definidos por níveis), consoante a ferramenta de NIDS/NIPS.

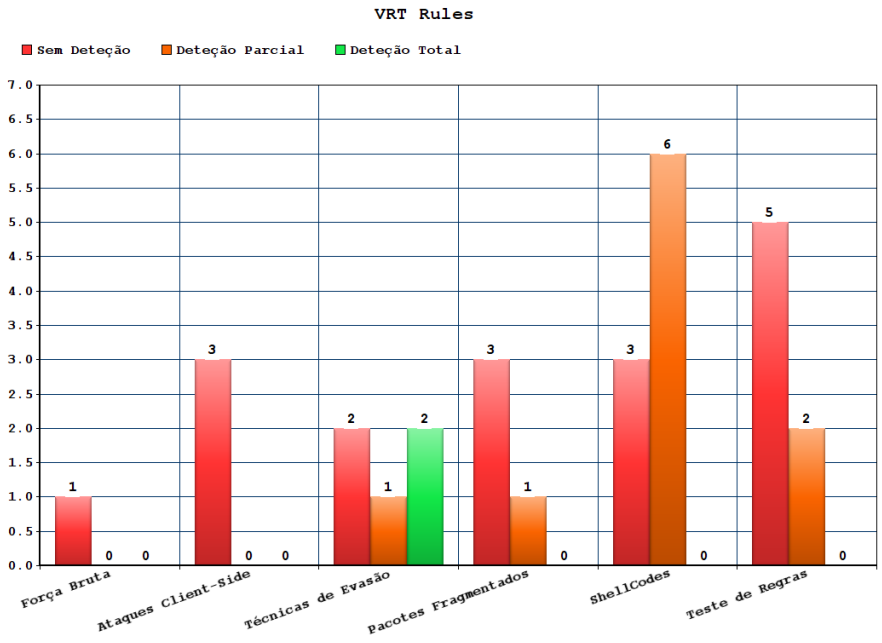


Figura 3.2: Algoritmo de atualização automática da quantidade de padrões de erro (definidos por níveis), consoante a ferramenta de NIDS/NIPS.

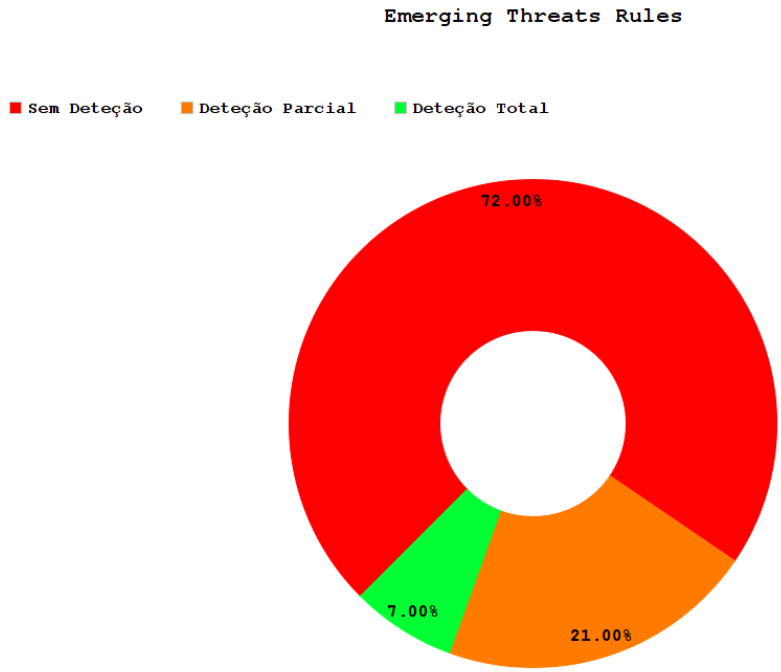


Figura 3.3: Algoritmo de atualização automática da quantidade de padrões de erro (definidos por níveis), consoante a ferramenta de NIDS/NIPS.

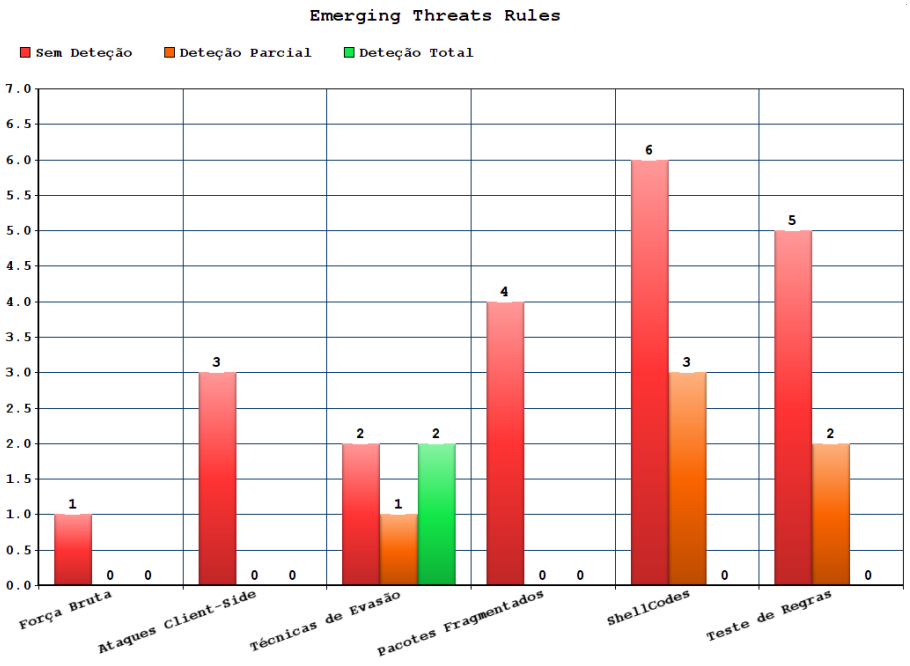


Figura 3.4: Algoritmo de atualização automática da quantidade de padrões de erro (definidos por níveis), consoante a ferramenta de NIDS/NIPS.

3.2.4 Seleção do Dinamismo das Ferramentas de IDS/IPS

Nos diversos testes de monitorização inerentes à secção 3.2.2.3 não foi possível configurar os ambientes de teste por forma a que se assemelhassem aos acessos que este tem quando se encontra exposto aos inúmeros utilizadores, sobretudo quando o servidor se encontra acessível através da Internet. Por esse motivo, foi descrito um perfil de utilização média do servidor pelos utilizadores a que este diz respeito:

- em média, os valores da memória livre e utilizada encontra-se na ordem dos 10% e 70%, respetivamente, para uma dada quantidade de memória RAM alocada;
- em média, os valores do poder de processamento disponível encontra-se na ordem dos 50%, para uma dada quantidade de processadores alocados.

Desta forma, e refletindo o tratamento de resultados das secções 3.2.2.3 e 3.2.3.2, pode-se concluir o seguinte:

- uma vez que o Snort é mais orientado para ambientes com recursos computacionais limitados, a sua utilização encontra-se orientado para **(a)** servidores com caraterísticas limitadas, nomeadamente o número de processadores igual ou inferior a 2 e a memória RAM igual ou superior a 2 GB, e **(b)** poder de processamento disponível igual ou inferior a 25%;
- o Suricata é executado nas condições contrárias às expostas no ponto anterior, uma vez que carateriza-se por ser uma ferramenta orientada a elevados desempenhos;
- por último, foram definidos dois níveis de regras que serão alternados com base na memória RAM disponível, qualquer que seja a ferramenta em execução. Assim, durante o *boot* do sistema, o segundo nível de padrões de erro será executado consoante a memória RAM seja igual ou superior a 2 GB. Por outro lado, se o sistema se encontrar em produção, o segundo de nível de padrões de erro apenas será executado se a memória livre for igual ou superior a 10% ou a memória utilizada for igual ou inferior a 70%.

3.3 Estrutura e Lógica da Solução

A solução desenvolvida assenta em três componentes principais da solução IPBrick: a *Firewall* IPtables, o módulo e ferramentas de IDS/IPS e o sistema em si. Cada uma dessas componentes interagem entre si, quer por interação direta da gestão do utilizador, a partir da aplicação *web* para administração do servidor, quer por mecanismos automáticos, baseados nas caraterísticas do sistema e nos *logs* das ferramentas de IDS/IPS.

Assim, ao longo desta secção, é exposta a estrutura e lógica para o desenvolvimento e implementação da solução, nomeadamente ao nível da arquitetura e requisitos da solução, protocolos idealizados e esquemas de bases de dados criados.

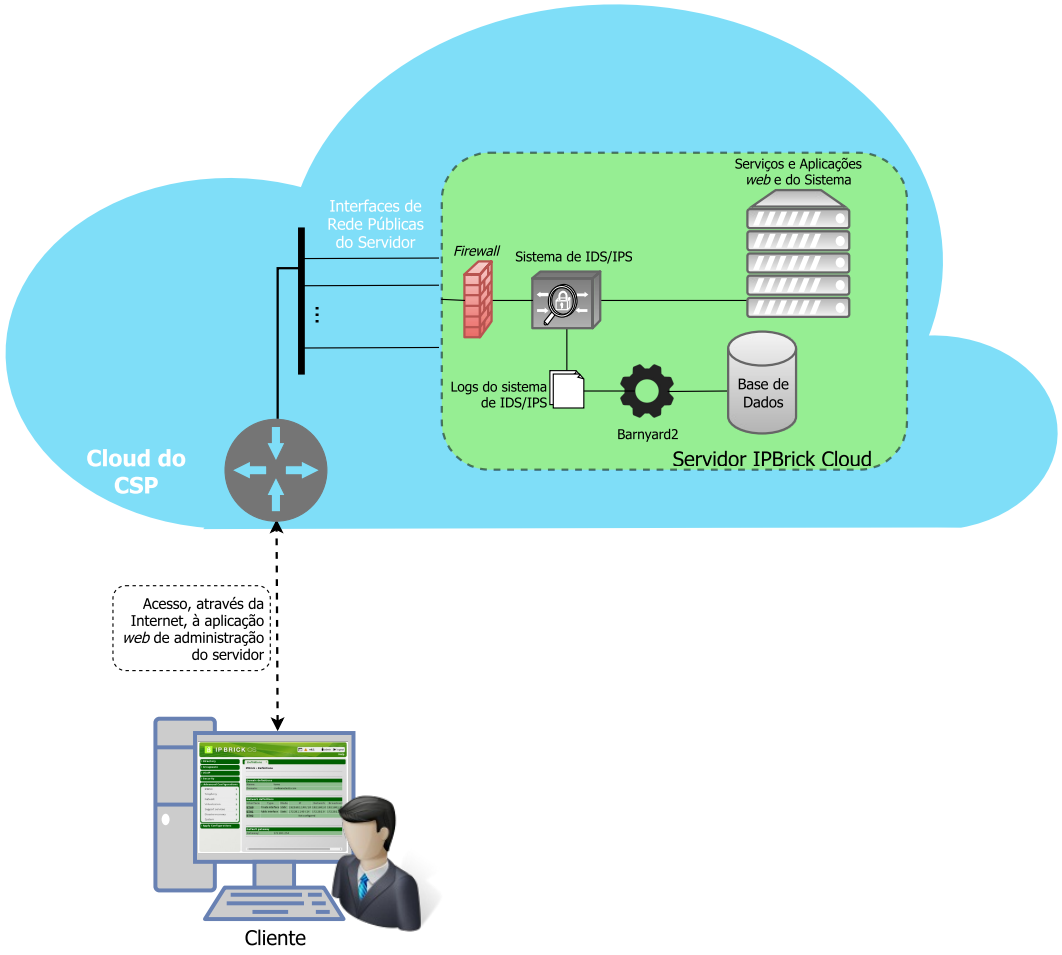


Figura 3.5: Arquitetura, de alto nível, da solução desenvolvida.

3.3.1 Arquitetura da Solução

Na figura 3.5 está representada a arquitetura, de alto nível, para a solução do projeto desenvolvido no contexto desta Dissertação. Esta solução é composta pela manipulação de diversos componentes, resultantes da interação, direta ou indireta, com o utilizador. Seguidamente, estão expostos cada um dos componentes, bem como o motivo e impacto da sua integração na solução desenvolvida.

Aplicação Web de Administração. Esta componente é a mais próxima da interação direta com o utilizador, a qual, com a integração do IPBRICK OS na *Cloud*, estará acessível a partir da Internet. Esta aplicação é responsável pela gestão e configuração da *Firewall* e do sistema de IDS/IPS, com um elevado nível de simplicidade e de abstração do *back-end* para o utilizador.

Interfaces de Rede Públicas. O número de pontos de acesso a um servidor remoto é diretamente proporcional ao número de interfaces de rede públicas que este possua – uma interface de rede pública tem um endereço IP público associado. A utilização de múltiplas interfaces de rede públicas dota o servidor de redundância de acessos, mas também possibilita a gestão e

disponibilização de serviços personalizada a cada interface (por exemplo, a interface pública **X** disponibiliza o serviço de *e-mail*, enquanto a interface pública **Y** disponibiliza aplicações *web*, mapeadas por *nameservers*). Por esse motivo, torna-se essencial uma solução que, não só possibilite a configuração da *Firewall* dedicada a cada interface, mas também a execução de múltiplas instâncias do sistema de IDS/IPS, a cada interface pública.

FIREWALL. Uma das principais soluções de segurança do perímetro, esta componente possui como funcionalidade adicional o redirecionamento do tráfego de rede do *kernel* para o *userspace*, quando o sistema de IDS/IPS encontra-se em modo de IPS. Por outro lado, dadas as suas limitações ao nível de estaticidade e de verificação de uma boa parte dos campos constituintes de um datagrama, a *Firewall* possui uma interação com o sistema de IDS/IPS, por forma a que este integre dinamismo e cubra uma maior verificação de campos do datagrama.

SISTEMA DE IDS/IPS. Esta componente é o *core* do projeto desenvolvido nesta Dissertação, proporcionando um aumento da segurança do perímetro com a sua capacidade reativa e/ou pró-ativa, mas também com o tratamento do *logging* gerado pelas ferramentas integrantes. Por forma a aumentar o raio de ação do sistema de IDS/IPS e, tendo em conta as diferentes fontes de eventos na qual uma ferramenta de IDS/IPS pode se basear, serão utilizadas, em qualquer momento, duas ferramentas de IDS/IPS: uma com fonte de eventos baseada na Rede e outra baseada no *Host* – na secção 3.3.2.

BARNYARD2. A estrita utilização desta componente deve-se à necessidade de armazenamento de *logs* das ferramentas de NIDS/NIPS na base de dados, algo que Snort e Suricata optaram por abandonar o suporte para um armazenamento direto. Além disso, o Barnyard2 possui a capacidade de leitura de ficheiros no formato *unified2*, um dos tipos de formatos dos *logs* gerados pelas ferramentas de NIDS/NIPS selecionadas. Note-se que o Barnyard2 já disponibiliza, por defeito, um *schema* para a base de dados, que automatiza a criação das tabelas necessárias.

BASES DE DADOS. Além da sua necessidade na aplicação *web* de administração, explorada no Capítulo 4, as bases de dadas criadas e configuradas possuem uma enorme importância, tal como referido no parágrafo anterior. Por outro lado, além das necessidades de *logging*, esta revela uma importância vital no tratamento dos *logs* em si, proporcionando a interligação do sistema de IDS/IPS com a *Firewall*.

3.3.2 Interação entre ferramentas de IDS/IPS de diferentes fontes de eventos

As ferramentas de IDS/IPS selecionadas possuem diferentes caraterísticas no que respeita às fonte de eventos. Enquanto o Snort e o Suricata são ferramentas que monitorizam a rede, utilizando as interfaces de rede públicas como sensores, o OSSEC baseia-se nos *logs* e ficheiros críticos do sistema, agindo na existência de modificações destes ficheiros. Por esse motivo, o sistema de IDS/IPS desenvolvido possui uma arquitetura híbrida, o que torna a solução IPBrick mais eficiente na proteção dos seus serviços, eliminando algumas das limitações que cada um dos tipos de ferramentas teriam se fossem executados individualmente, tornando o sistema mais robusto.

Desta forma, é possível cobrir dois objetivos principais relativamente à otimização da segurança no IPBRICK OS, nomeadamente ao nível de **(a)** ataques maliciosos de intrusos, cuja maior

responsabilidade recai nas ferramentas de NIDS/NIPS, e **(b)** negligências do utilizador e existência de *software* maliciosos que permitem acessos desautorizados (vulgarmente designados por *rootkits*), em que a responsabilidade recai na ferramenta de HIDS/HIPS.

3.3.3 Mecanismos de auto-adaptação

Enquanto a implementação de um maior nível de segurança numa solução tecnológica representa um dos requisitos de maior prioridade, na *Cloud*, este requisito adquire uma maior proporção. No entanto, se, por um lado, pretende-se aumentar o nível de segurança devido à execução dos serviços na *Cloud*, por outro este aumento da segurança traduz-se num impacto negativo ao nível do desempenho do sistema e latências na resposta por parte deste.

Essa degradação ao nível do desempenho e latências no servidor na *Cloud* são, no contexto das soluções da segurança introduzidas, muito por culpa do sistema de IDS/IPS, tal como se pode visualizar nos resultados obtidos na secção 3.2. Por esse motivo, uma gestão inteligente destas ferramentas é a estratégia ideal para lidar com os problemas acima identificados.

A estratégia de gestão inteligente das ferramentas contempla os seguintes mecanismos:

- definição de níveis de padrões de erro, para cada conjunto respeitante a um dado serviço;
- seleção das ferramenta de NIDS/NIPS executada a cada *boot* do sistema, consoante as caraterísticas do servidor;
- modificação da ferramenta de NIDS/NIPS em execução, consoante o desempenho médio do servidor;
- e adição de regras na *Firewall*, automática e manualmente, consoante o nível de perigo de avisos gerados pelo sistema de IDS/IPS.

A modularização dos padrões de erro utilizados pelas ferramentas de NIDS/NIPS em diferentes níveis permite otimizar o impacto provocado na memória e processamento pelos motores de deteção destas ferramentas, uma vez que, para uma mesma quantidade de memória e disco alocado na comparação entre padrões de erro existentes, o tempo de utilização desses recursos serão, expectavelmente, menores. Na figura 3.6 pode-se observar o algoritmo responsável pela execução deste mecanismo, cuja implementação será explorada e exemplificada no Capítulo 4.

A seleção e modificação da ferramenta de NIDS/NIPS em execução permite, por sua vez, consoante as caraterísticas definidas para o servidor e o desempenho médio do sistema, respetivamente, alterar a ferramenta de NIDS/NIPS para aquela que permitir minimizar o impacto no desempenho do servidor – esta perceção advém do estudo das ferramentas de IDS/IPS, realizado na secção 3.2. Nas figuras 3.7a e 3.7b podem-se observar os algoritmos representativos destes dois mecanismos similares, mas com momentos de execução diferentes, cuja implementação será explorada e exemplificada no Capítulo 4.

Por último, a geração de regras da *Firewall* com base nos *logs* do sistema de IDS/IPS visa diminuir a carga que este sofre na entrada de novos pacotes. Tal acontece, sobretudo, quando o

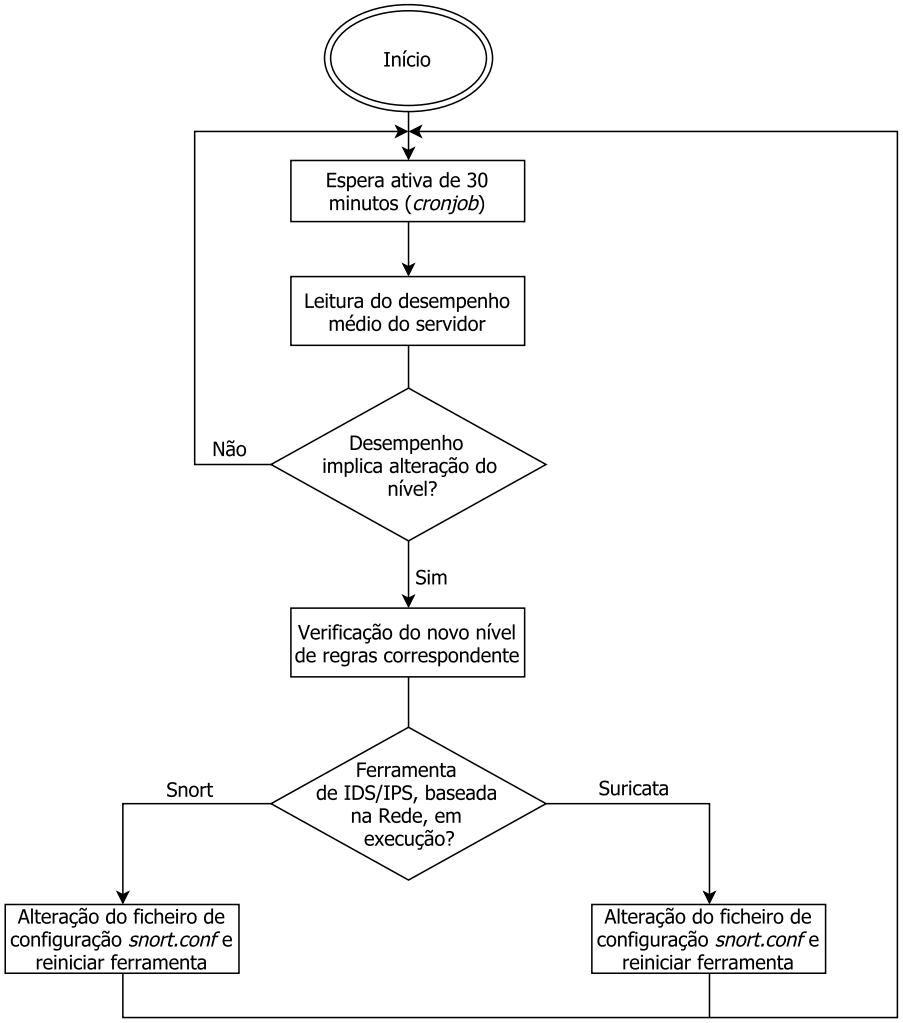
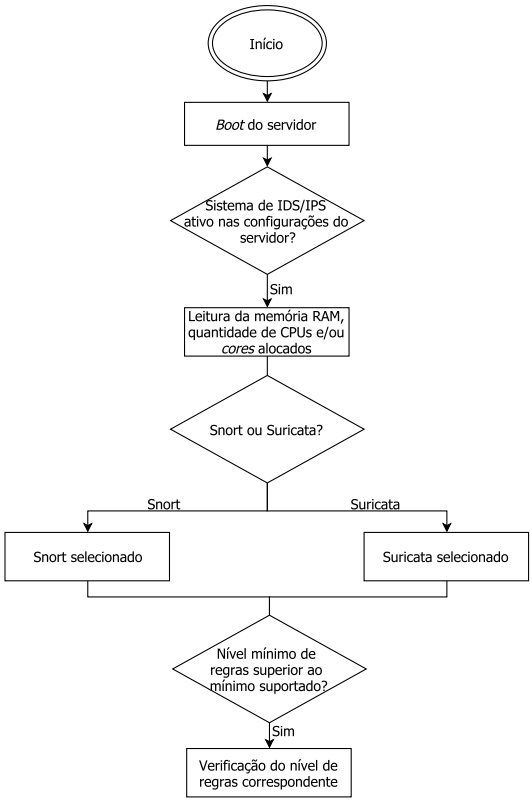
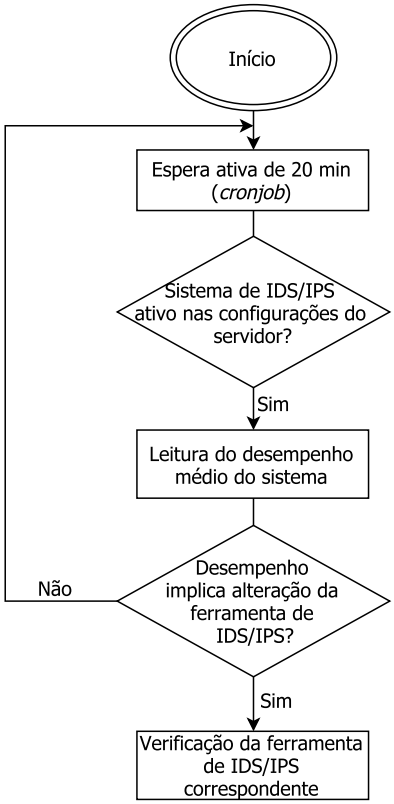


Figura 3.6: Algoritmo de atualização automática da quantidade de padrões de erro (definidos por níveis), consoante a ferramenta de NIDS/NIPS.

sistema de IDS/IPS está configurado em modo de IPS, pelo que a *Firewall* redireciona o tráfego de entrada e saída do *kernel* para o *userspace*, no qual está a ferramenta de NIDS/NIPS responsável por decidir o destino deste tráfego. Deste modo, este redirecionamento deixa de ser necessário, evitando **(a)** a latência da transação de pacotes *kernel-to-userspace* e **(b)** recursos necessários para sistema de IDS/IPS verificar o destino do pacote de chegada. Na figura 3.8 está representado o algoritmo relativo a este mecanismo, sendo que, posteriormente, nos Capítulos 4 e 5 são explorados a implementação e o impacto no desempenho do sistema deste mecanismo.



(a) Algoritmo de seleção automática da ferramenta de NIDS/NIPS, a cada *boot* do sistema.



(b) Algoritmo de modificação automática da ferramenta de NIDS/NIPS.

3.3.4 Módulo de Gestão da Segurança

3.3.4.1 *Firewall*

A *Firewall* atualmente existente possui um nível de abstração muito baixo, encontrando-se a um nível semelhante ao da configuração da IPtables através da linha de comandos. Consequentemente, atualmente não existe qualquer suporte à criação massiva de novas regras. Por forma a integrar a *Firewall* de maior facilidade de compreensão e utilização, é necessário que dividir a *firewall* em três camadas:

- camada de alto-nível: situada ao nível da UI, esta camada contém os dados que possibilitam uma fácil compreensão de políticas de segurança existentes na *Firewall*, por defeito ou executadas pelo sistema, mas também aquelas que o utilizador define;

- camada de nível intermédio: transparente ao utilizador, esta camada efetua a tradução entre as configurações existentes nas camadas adjacentes (alto-nível e baixo-nível), podendo ser definida como uma *biblioteca* de funções de ambas as camadas adjacentes;

- camada de baixo-nível: possui um grau de configurações semelhante ao que se efetua na IPtables, através da linha de comandos. Note-se que, dada a complexidade de funcionalidades da IPtables, esta camada continua a necessitar de uma UI, uma vez que não é possível

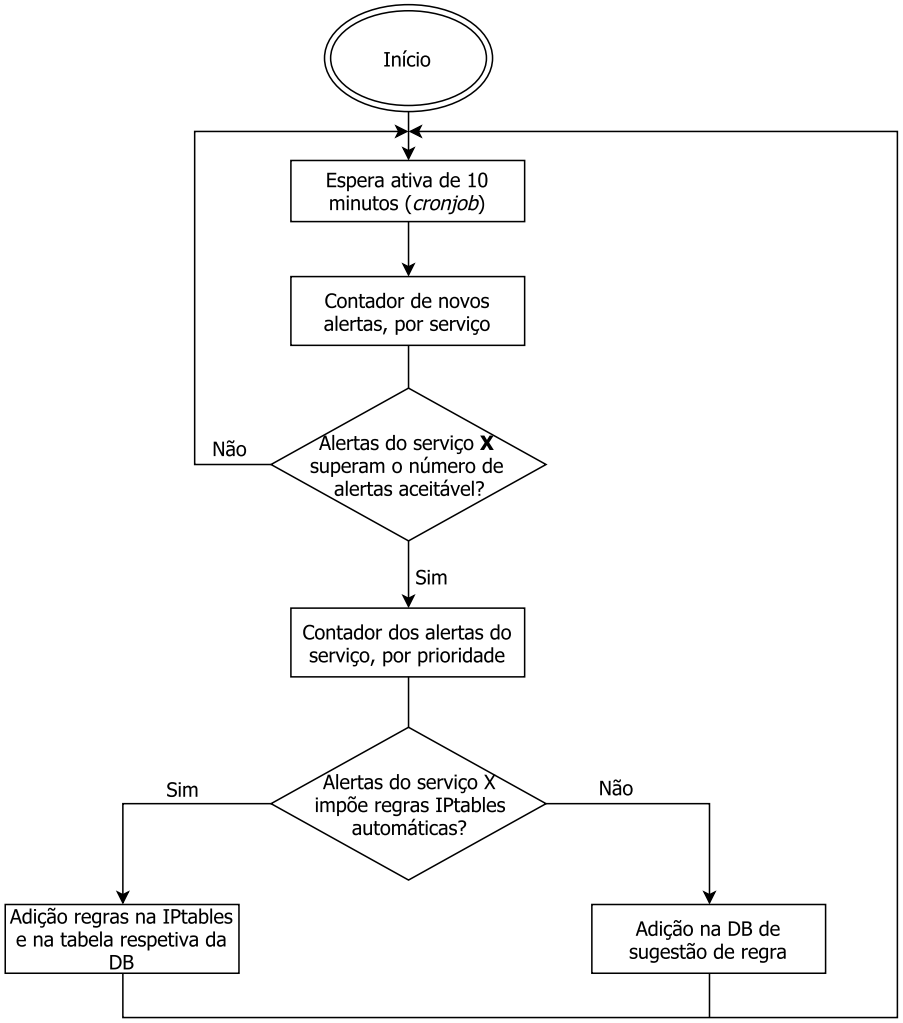


Figura 3.8: Algoritmo de geração automática de regras da *Firewall*, baseado nos *logs* da ferramenta de NIDS/NIPS.

mapear todas as funcionalidades da IPtables numa forma *user-friendly*, sem limitar as possibilidades de configuração existentes.

Assim, no contexto desta Dissertação, por forma a integrar as regras geradas automaticamente pelos *logs* do sistema de IDS/IPS, este deverá utilizar as funções definidas na biblioteca da camada do nível intermédio, por forma a propagar as alterações efetuadas aos níveis deste modelo de três níveis, possibilitando a implementação das regras ao nível da *Firewall* IPtables, mas também para compreensão pelo utilizador. No Capítulo 4 serão explorados e exemplificados a implementação desta interligação do sistema de IDS/IPS com a *Firewall*.

3.3.4.2 Sistema de IDS/IPS

Tal como a *Firewall*, o sistema de IDS/IPS deve contemplar um nível de abstração elevado, pelo que as configurações principais do utilizador apenas se baseiem no modo que este sistema

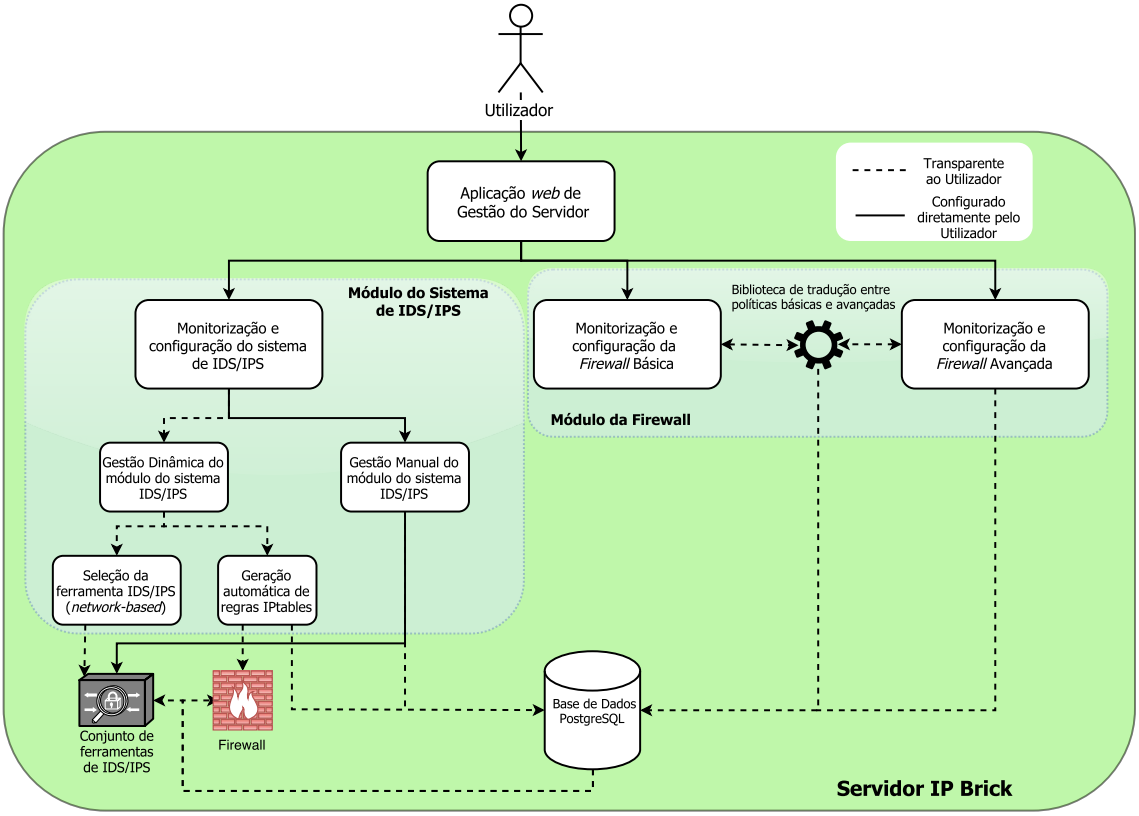


Figura 3.9: Sub-módulos do módulo de segurança da solução desenvolvida, com e sem interação direta do utilizador no sistema e na base de dados.

esteja a atuar (em modo de IDS ou IPS) e quais os serviços que pretende monitorizar na interface de rede selecionada. Por esse motivo, este submódulo pertencente ao módulo de segurança contempla duas componentes paralelas, resultantes da interação com o utilizador, nomeadamente:

- componente dinâmica, transparente ao utilizador, que resulta na seleção da ferramenta de NIDS/NIPS, bem como do nível de regras;
- componente estática, diretamente utilizada pelo utilizador, e baseia-se ao nível dos serviços monitorizados, modo de execução do sistema e interface de rede utilizada como sensor.

Por outro lado, por forma a possibilitar a implementação dos mecanismos definidos na secção 3.3.3, é necessário efetuar as seguintes configurações ao nível das ferramentas de IDS/IPS:

- separação de regras, por cada serviço, em vários ficheiros, cada qual correspondente ao nível de regras estabelecido;
- definir gama de inteiros para cada serviço, a qual será dedicada aos identificativos das regras relativas a um dado serviço.

3.3.4.3 Bases de Dados

As bases de dados possuem uma enorme utilidade no armazenamento e apresentação de dados na aplicação *web* de administração, mas também na execução de tarefas a cada *boot* do sistema e periodicamente, enquanto o sistema se encontra em produção. Por outro lado, tendo em conta um dos objetivos da solução desta Dissertação – a geração de regras da *Firewall* baseado nos *logs* do sistema de IDS/IPS –, a base de dados adquire um papel preponderante na apresentação ao utilizador de informação executada sem interação do mesmo.

Assim, por forma a possibilitar a manipulação das ferramentas de IDS/IPS através da aplicação de administração, mas também possibilitar armazenar as execuções automáticas do sistema, as tabelas criadas seguem os modelos relacionais apresentados nas figuras 3.10-3.12. Note-se que os modelos relacionais expostos nas figuras 3.10 e 3.11 são relativos ao sistema de IDS/IPS, enquanto o modelo relacional exposto na figura 3.12 é relativo à *Firewall*.

As tabelas integrantes no modelo da figura 3.10 devem-se a dois objetivos principais, nomeadamente **(a)** manipulação alto-nível das ferramentas de IDS/IPS, **(b)** implementação de um sistema de alarmística, configurado pelo utilizador e baseado nos *logs* do ficheiro e **(c)** interação com a *Firewall*. Por outro lado, a existência de múltiplas tabelas deve-se ao tipo de relação estabelecido entre as principais entidades – o sistema de IDS/IPS e a alarmística e serviços monitorizáveis associados ao mesmo. Assim, por exemplo, deve-se ser possível criar múltiplas definições de alarmística que notifique, no máximo, o mesmo número de serviços monitorizados pelo sistema.

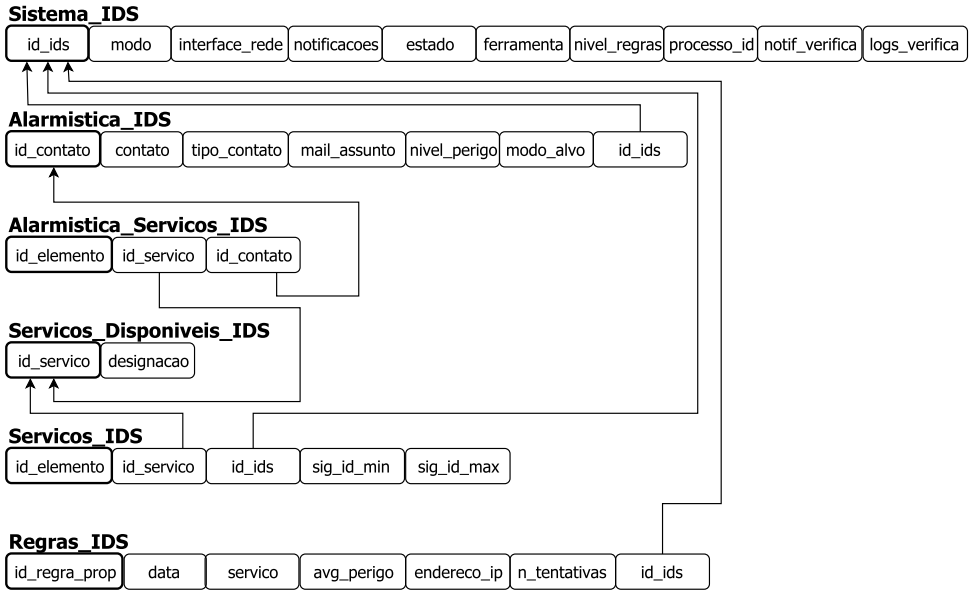


Figura 3.10: Modelo relacional relativo ao sistema de IDS/IPS, ao nível da apresentação de dados na aplicação *web* de administração, mas também de utilização e atualização dos mesmos no mecanismos de auto-adaptação.

Por outro lado, o modelo relacional exposto na figura 3.11 é parte integrante do modelo relacional incluído no Barnyard2. No entanto, apenas está exposto o conjunto de colunas e tabelas úteis

no desenvolvimento da solução desta Dissertação.

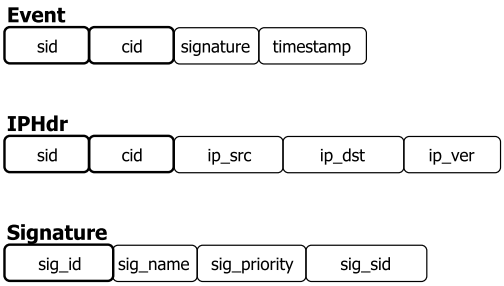


Figura 3.11: Modelo relacional relativo aos *logs* das ferramentas de NIDS/NIPS, para utilização na geração de regras da *Firewall*, mas também na apresentação na aplicação *web* de administração.

Por último, na figura 3.12 encontra-se o modelo relacional relativo às colunas e tabelas úteis no contexto da solução, pelo que este não se encontra na sua versão completa. Em relação ao exposto, note-se a interligação entre as tabelas *Firewall_Basica* e *Firewall_IPtables*, concretizada pela utilização de uma chave estrangeira nesta última tabela, que referencia a *Firewall_Basica*. Tal como se pode induzir, a tabela *Firewall_Basica* encontra-se ao nível da camada de alto nível da *Firewall*, enquanto a tabela *Firewall_IPtables* enquadra-se ao nível da camada de baixo nível, camadas exploradas na secção 3.3.4.1. Por esse motivo, um tuplo da *Firewall_Basica* pode ser referenciado por múltiplos tuplos da tabela *Firewall_IPtables*.

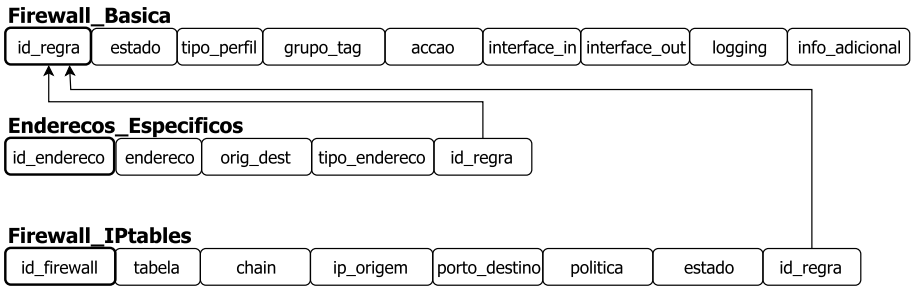


Figura 3.12: Modelo relacional da *Firewall*, utilizado na interação com o sistema de IDS/IPS.

No Capítulo 4 será exposto, mais detalhadamente e com recursos a exemplos práticos, a interação entre as diversas tabelas representadas em cada modelo relacional, bem como a necessidade das suas colunas constituintes.

Capítulo 4

Desenvolvimento e Implementação

Neste Capítulo são abordados os desenvolvimentos e a implementação da solução para o projeto desta Dissertação, referidos no Capítulo 3.Trata-se, por isso, de uma contextualização e um maior aprofundamento de conceitos e requisitos abordados na secção 3.3, através da exposição de excertos do desenvolvimento da solução.

Assim, ao longo deste Capítulo, pode ser consultado **(a)** a metodologia utilizada, com a exploração de tecnologias de desenvolvimento envolvidas e especificações implementadas e **(b)** o *front-end* e *back-end* do Módulo de Segurança desenvolvido, no que diz respeito não só à aplicação *web* de administração, mas também ao sistema IPBRICK OS.

4.1 Introdução

4.1.1 Tecnologias e Ferramentas

No desenvolvimento da solução para esta Dissertação, foram utilizadas diversas tecnologias e ferramentas para as implementações ao nível da aplicação *web* de administração, como também do sistema em si.

A aplicação *web* de administração possui dois níveis de implementações: um alto-nível, o *front-end*, com o qual o utilizador possui interação, e um baixo-nível, o *back-end*, no qual assentam as bases de dados e o sistema. Em relação ao *front-end*, foram utilizadas diversas tecnologias, nomeadamente:

- HTML: linguagem de estruturação de páginas *web*, utilizada para o fornecimento das UI com as quais o utilizador se depara e interage, tem como responsabilidade principal a construção e disposição dos diversos elementos constituintes de uma UI;

- CSS: tal como o HTML, trata-se de um linguagem de estruturação e é responsável por fornecer uma customização dos diversos elementos inerentes ao HTML, sendo responsável pelo *design* dos elementos HTML;

- JavaScript: linguagem de programação, interpretada pelos vários navegadores *web*, é responsável por efetuar verificações dos vários campos submetidos – evitando, assim, uma carga desnecessária no servidor –, bem como de mostrar, ou não, elementos HTML.

Por outro lado, quanto ao *back-end*, foram utilizadas as seguintes tecnologias:

- PHP: linguagem de programação, utilizada para efetuar pedidos ao servidor, é responsável pela obtenção de dados das bases de dados, disponibilizados na aplicação *web* de administração, mas também pela execução de *scripts* PHP, utilizados pelos mecanismos de auto-adaptação;
- *Scripts* Shell: conjunto de comandos, utilizados indiretamente pelas ações na aplicações *web* de administração, mas também pela execução de *cronjobs* inerentes aos mecanismos de auto-adaptação;
- PostgreSQL: sistema de gestão de bases de dados na qual as bases de dados utilizadas foram criadas e configuradas;
- Crontab: *software* do sistema que permite a execução periódica de tarefas definidas.

4.1.2 Metodologias e Estratégias adotadas

A implementação da estrutura e lógica da solução, apresentada na secção 3.3, foi constituída por diversas etapas, nomeadamente:

- otimização da UI do sub-módulo da *Firewall*, constituinte da aplicação *web* de administração;
- desenvolvimento da UI do novo sub-módulo dedicado ao sistema de IDS/IPS;
- criação das bases de dados para as ferramentas de IDS/IPS e das tabelas necessárias para armazenar os dados inseridos na aplicação *web* de administração;
- configuração das ferramentas de IDS/IPS, por forma a suportar os modos de IDS e IPS e armazenar os *logs* nas bases de dados criadas para cada uma delas;
- criação de *cronjobs* para executar os *scripts* Shell e PHP, utilizados nos mecanismos de auto-adaptação.

FIREWALL. Os desenvolvimentos realizados tiveram como intuito principal a simplificação da UI da *Firewall* e a introdução massiva de regras na *Firewall*. Na simplificação da UI os desenvolvimentos principais situaram-se sobre novas tabelas criadas na base de dados (ver figura 3.12) e na criação de uma biblioteca, *LibFirewall.php*, que efetuam o mapeamento entre funcionalidades de alto-nível, despoletadas ao nível da UI, e funcionalidades baixo-nível, com um nível técnico semelhante ao da IPtables. Por outro lado, a geração massiva de regras na *Firewall* foi obtida pela

introdução do conceito de *Grupos de Redes*, no qual estão contidos múltiplos endereços IP e de redes e que pode ser associado a uma nova regra da *Firewall*, sendo a biblioteca criada responsável por desmembrar os elementos do grupo e criar uma nova regra da *Firewall*.

SISTEMA DE IDS/IPS. As UI desenvolvidas, integradas na aplicação *web* de administração, tiveram como objetivo principal a abstração do utilizador quanto à seleção e configuração de uma ferramenta de IDS/IPS, através da linha de comandos, bem como dotar estas ferramentas de funcionalidades de alarmísticas – algo que, por defeito, apenas o OSSEC possui integrado. Por outro lado, tendo em conta a possibilidade de existência de múltiplas interfaces de rede públicas, foi necessário associar o identificativo do processo relativo à ferramenta de NIDS/NIPS que está à escuta em cada interface pública.

CONFIGURAÇÕES IDS/IPS. Os desenvolvimentos realizados situam-se em três configurações principais, nomeadamente **(a)** a configuração dos ficheiros de configuração de cada ferramenta de IDS/IPS, **(b)** a criação de um ficheiro de configuração e de diretórios independentes, para cada nível de padrões de erro de uma ferramenta de NIDS/NIPS, **(c)** a alteração dos identificativos dos padrões de erro, por serviço, e **(d)** a adição de regras na *Firewall*, no caso do sistema estar em execução no modo IPS.

Os ficheiros de configuração possuem variáveis estáticas e dinâmicas, sendo que as estáticas apenas podem ser comentadas, enquanto as dinâmicas podem também sofrer mudanças nos valores atribuídos. As variáveis estáticas são relativas às localizações dos ficheiros de configuração respeitantes aos vários níveis de padrões de erro, à configuração do *logging* da ferramenta e a configurações de utilização *softwares* não incluídos, por defeito. Por outro lado, as variáveis dinâmicas dizem respeito às informações da interface de rede pública, nomeadamente o nome e endereço IP.

Por outro lado, por forma a implementar o dinamismo quanto ao número de padrões de erro em execução, foi necessário criar diretórios exclusivos para estas – por exemplo, */etc/snort/rules/level/level1/* –, bem como um ficheiro de configuração para cada conjunto de padrões de erro. Além disso, por forma a que fosse possível identificar mais facilmente o identificativo de um alerta, foi atribuído uma gama de identificativos a cada serviço (por exemplo, o serviço FTP tem identificativos compreendidos entre 1000 e 4999, enquanto o serviço VoIP possui identificativos entre 5000 e 9999). Desta forma, na geração de regras da *Firewall*, quer automática, como manualmente, facilmente se consegue identificar o serviço em causa.

Por último, quando o sistema de IDS/IPS está em execução com o modo de IPS, é necessário que o tráfego de rede seja redirecionado para o *userspace*, onde se encontra em execução a ferramenta de NIDS/NIPS. Para tal, é necessário adicionar os comandos expostos seguidamente, por cada interface de rede pública, após as regras existentes na *Firewall*, quer sejam por defeito ou adicionadas pelo utilizador. Desta forma, o tráfego correspondente a essas regras não necessitará de ser redirecionado para o *userspace*, evitando custos computacionais adicionais e a possibilidade de ser rejeitado, consoante os padrões de erro existentes no sistema de IDS/IPS. Note-se que o parâmetro <*n_queue*> é específico à instância da ferramenta de NIDS/NIPS que se encontra em execução numa dada interface de rede pública, pelo que devem possuir diferentes valores entre si.

```
$ iptables -t filter -A INPUT -j NFQUEUE --queue-num=<n_queue>
$ iptables -t filter -A OUTPUT -j NFQUEUE --queue-num=<n_queue>
```

CRONJOBS. Por último, os *cronjobs* configurados são utilizados para a leitura periódica do desempenho do sistema e para a mudança da ferramenta de NIDS/NIPS e nível de regras em execução.

4.2 Módulo de Segurança

Nesta secção encontram-se expostos as interações existentes, ao nível do módulo de segurança desenvolvido, quanto ao *front-end*, como também ao *back-end*. É, por isso, descrito os desenvolvimentos efetuados sobre a aplicação *web* de administração, como também sobre o sistema em si.

4.2.1 Manipulação das ferramentas de IDS/IPS

4.2.1.1 Configuração das ferramentas de IDS/IPS utilizadas

As ferramentas de IDS/IPS incluídas no sistema de IDS/IPS desenvolvido possuem duas modificações essenciais, nomeadamente **(i)** a modificação dos subdiretórios relativos aos padrões de erro existentes nas ferramentas Snort e Suricata e **(ii)** a modificação do local de armazenamento dos *logs*.

MODIFICAÇÃO DE SUBDIRETÓRIOS. Esta modificação, visualizada nos Anexos B.1.2 e B.2.2, proporciona o carregamento de diferentes quantidades de padrões de erro e será explorada na secção 4.2.1.3.

MODIFICAÇÃO DO LOCAL DE ARMAZENAMENTO. Estas modificações são constituídas por etapas diferentes, consoante se trata de uma ferramenta de NIDS/NIPS ou de HIDS/HIPS. Uma vez que Snort e Suricata já não possuem suporte para armazenamento direto na base de dados, foi necessário uma ferramenta de interligação: o Barnyard2. Assim, por forma a que Snort e Suricata efetuassem o *logging* na base de dados, foi necessário efetuar as seguintes alterações nos ficheiros de configuração respetivos:

```
# Snort
output alert_unified2: filename snort.alert, limit 128, nostamp

# Suricata
- unified2-alert:
  enabled: yes
  filename: unified2.alert
```

Por outro lado, na configuração do OSSEC, ao nível do ficheiro de configuração, esta revelou-se mais simples, bastando adicionar o excerto exposto seguidamente. No entanto, uma vez que

o sistema IPBRICK OS dispõe de duas interfaces de acesso ao sistema de gestão de bases de dados, foi necessário alterar o código-fonte do mesmo antes de se proceder à instalação – ver Anexo B.3.1.

```
<ossec_config>
 <database_output>
  <hostname>localhost</hostname>
  <username>sistema_ids</username>
  <password>sistema_ids</password>
  <database>ossec</database>
  <type>postgresql</type>
 </database_output>
</ossec_config>
```

No Anexo B podem ser encontradas as configurações realizadas, numa versão mais completa, para as ferramentas constituintes do sistema de IDS/IPS – Snort, Suricata, OSSEC e Barnyard2.

4.2.1.2 Seleção de ferramenta de IDS/IPS baseada na Rede

BOOT DO SISTEMA. Este mecanismo utiliza as caraterísticas iniciais do servidor, nomeadamente a memória RAM e número de processadores alocados, para perceber qual a ferramenta de NIDS/NIPS colocada em execução. No entanto, previamente a essa seleção, é necessário implementar o *script* PHP por forma a que seja executado apenas quando o servidor de base de dados está operacional. Tal deve-se à necessidade de verificação do estado do sistema de IDS/IPS, para uma dada interface de rede: se este estiver ativo, o mecanismo é executado; caso contrário, é ignorado e as ferramentas de IDS/IPS permanecem inativas até alteração do estado do sistema de IDS/IPS através da aplicação *web* de administração. Note-se também a necessidade de colocar o Barnyard2 em execução, vital para o *logging* das ferramentas de NIDS/NIPS.

Desta forma, a implementação deste mecanismo foi realizada da seguinte forma:

1. Obtenção do estado do sistema de IDS/IPS, guardado na coluna *estado* da tabela *Sistema_IDS*, através de um *script* PHP, executado a cada *boot*;

2. Se o sistema de IDS/IPS se encontrar ativo, são obtido os valores da memória RAM e número de processadores através de um *script* Shell, que passa os resultados para o *script* PHP;

3. Consoante os valores obtidos, é selecionado o Snort ou o Suricata, bem o nível de padrões de erro ajustado ao sistema, alterando o ficheiro de configuração da ferramenta de NIDS/NIPS selecionada com esse nível de padrões de erro, bem como iniciado o Barnyard2;

4. Por último, é executada a ferramenta de NIDS/NIPS selecionada, bem como o Barnyard2, essencial para efetuar o *logging* da ferramenta em causa.

Note-se que, para o ponto **2.**, os valores da caraterísticas do *CPU* e da memória RAM são obtidos da seguinte forma, ficando os valores guardados nas variáveis *$n_cpu* e *$n_ram*:

```php
<?php

  $command_ram = "awk '/MemTotal/ { print $2; exit; }' /proc/meminfo";
  execAsRoot($command_ram,$n_ram);

  $command_cpu = "nproc";
  execAsRoot($command_cpu,$n_cpu);

?>
```

Por outro lado, por forma a efetuar as alterações no sistema, o *script* Shell (consulte Anexo C.2 para observar o seu código-fonte) é executado da seguinte forma

```
$ /bin/sh /opt/system/scripts/ids_ips/setup_ids.sh -cb 1 <ferramenta_nids> <
    modo_execucao> 0 <interface_rede>
```

SISTEMA EM PRODUÇÃO. Este mecanismo utiliza o desempenho médio do servidor, ao nível da memória e do processamento, para perceber qual a ferramenta de NIDS/NIPS adequada para estar em execução e otimizar a utilização dos recursos do servidor. No entanto, previamente a essa seleção, é necessário implementar o *script* PHP por forma a verificar se o sistema de IDS/IPS encontra-se ativo: se este estiver ativo, o mecanismo é executado; caso contrário, é ignorado e as ferramentas de IDS/IPS permanecem inativas até alteração do estado do sistema de IDS/IPS através da aplicação *web* de administração. Note-se também a necessidade do Barnyard2 estar em execução, vital para o *logging* das ferramentas de NIDS/NIPS.

Desta forma, a implementação deste mecanismo foi realizada da seguinte forma, executada periodicamente com recurso a um *cronjob*:

1. Leitura do valor da coluna *estado* da tabela *Sistema_IDS*, para uma dada instância do sistema de IDS/IPS. Caso o sistema de IDS/IPS esteja ativo, é obtido o desempenho médio do servidor, através dos valores guardados em base de dados;

2. Leitura da ferramenta de NIDS/NIPS em execução, guardada na coluna *ferramenta* da tabela *Sistema_IDS*, para uma dada instância do sistema de IDS/IPS, e avaliada a viabilidade de uma mudança na ferramenta, consoante o desempenho médio calculado;

3. Se for necessário alterar a ferramenta atual, é realizada uma nova leitura à tabela *Sistema_IDS* para obter os valores das colunas *modo* e *processo_id*, por forma a perceber a estratégia de mudança da ferramenta. Se o modo for IDS, o processo com identificativo igual ao valor de *processo_id* é cancelado, e é executada uma nova instância da nova ferramenta de NIDS/NIPS; caso o modo seja IPS, é necessário eliminar as regras IPtables necessárias à execução deste modo e, posteriormente, cancelado o processo com o valor de *processo_id* e executada uma nova instância da nova ferramenta de NIDS/NIPS.

Similarmente à execução da verificação do sistema de IDS/IPS no arranque do sistema, quando este se encontra em produção é utilizado o mesmo *script* Shell para efetuar as alterações ao nível

do sistema, e que é executado a partir do *script* PHP. Assim, a chamada do *script* Shell é realizada com os seguintes parâmetros

```
$ /bin/sh /opt/system/scripts/ids_ips/setup_ids.sh -cb 2 <ferramenta_nids> <
    modo_execucao> <pid_ferramenta> <interface_rede>
```

4.2.1.3 Seleção de nível de padrões de erro

O número de padrões de erro em execução no sistema de IDS/IPS envolve uma maior quantidade de recursos computacionais, utilizados pela ferramentas integrantes do mesmo. Por esse motivo, foi implementado um mecanismo automático e periódico que, baseado no desempenho médio do sistema em cada período, pondera a alteração do número de padrões de erro. Desta forma, consegue-se um equilíbrio entre o tempo de resposta do servidor e a segurança do mesmo: enquanto o desempenho do servidor afeta os tempos de latência a pedidos dos utilizadores remotos, a quantidade de padrões de erro ativo aumenta a probabilidade de detetar um possível ataque ou negligência.

Numa fase inicial, por forma a possibilitar a execução deste mecanismo, foi necessário configurar os diretórios do Snort e Suricata para que os ficheiros de padrões de erro estivessem dispostos por nível e por interface de rede pública – ver Anexos B.1.2 e B.2.2. Por outro lado, em cada ficheiro de configuração do Snort e Suricata, foram adicionadas as seguintes linhas, para inclusão dos ficheiros de configuração de cada nível.

```
# Snort
var RULE_CONFIG_PATH /etc/snort/rules/config_files/eth1/
include $RULE_CONFIG_PATH/rules_level1.conf
include $RULE_CONFIG_PATH/rules_level2.conf

# Suricata
include: /etc/suricata/rules/config_files/eth1/rules_level1.conf
include: /etc/suricata/rules/config_files/eth1/rules_level2.conf
```

Em relação à implementação do mecanismo, esta assenta em duas etapas, nomeadamente **(a)** recolha e tratamento de dados provenientes da base de dados, implementado com recurso a *scripts* PHP, e **(b)** modificação dos ficheiros de configuração das ferramentas do sistema de IDS/IPS, através de *scripts* Shell. Seguidamente, expõe-se o procedimento inerente ao algoritmo exposto na figura 3.6 e que recorre de tabelas do modelo relacional da figura 3.10:

1. Obtenção dos dados relativos ao desempenho do sistema nos últimos 30 minutos, guardados periodicamente através da execução de um *script* PHP num *cronjob*;

2. Obtenção do desempenho médio do sistema e obtenção do nível de regras e ferramenta de NIDS/NIPS em vigor, guardados nas colunas *nivel_regras* e *ferrramenta*, respetivamente, da tabela *Sistema_IDS*, para uma dada instância do sistema de IDS/IPS;

3. Identificação do nível de padrões de erro ideal. Caso seja necessário efetuar uma mudança, este é alterado nos ficheiros de configuração da ferramenta de NIDS/NIPS (através de um *script* Shell) e os valores na base de dados são atualizados (através de um *script* PHP).

Por forma a alterar os níveis de padrões de erro nos ficheiros de configuração, abordado no ponto **3.**, a implementação desenvolvida passa pela chamada de um *script* Shell, executado por um *script* PHP, em que a chamada é realizada com os seguintes parâmetros

```
1 $ /bin/sh /opt/system/scripts/ids_ips/setup_ids.sh -cl 2 <nivel> <
    interface_rede>
```

4.2.2 Interação entre *Firewall* IPtables e ferramentas de IDS/IPS

Um dos objetivos desta Dissertação passava por dotar a *Firewall* de mecanismos inteligentes que permitissem torná-la dinâmica, isto é, fosse capaz de proteger o servidor contra acessos para os quais não possui políticas de ação. A estratégia idealizada para cumprir tal objetivo contemplou a interação do sistema de IDS/IPS com a *Firewall*, o qual é responsável por gerar *logs*. Os *logs* do sistema de IDS/IPS são o *core* desta integração de dinamismo na *Firewall*: através do tratamento destes, consoante os níveis de perigo dos acessos, estes geram regras para a *Firewall*. No entanto, a integração das novas regras na *Firewall* pode ser realizada em dois diferentes modos:

- automático, na qual as regras são automaticamente adicionadas ao nível do sistema e da base de dados, através de *scripts* Shell e PHP;
- manual, na qual as regras são adicionados por ação do utilizador, na aplicação *web* de administração, cuja metodologia – *front-end* e *back-end* – é apresentada na secção 4.2.3.

A geração de regras de *Firewall* é um processo periódico, implementado através de um *cron-job*, sendo verificados todos os novos *logs* gerados nesse período de espera pela ferramenta de NIDS/NIPS em execução. A verificação de novos *logs* é realizada por um *script* PHP, que processa a informação da seguinte forma:

1. Obter as múltiplas instâncias ativas do sistema de IDS/IPS, através da tabela *Sistema_IDS* (ver figura 3.10);
2. Por cada instância, obter o endereço IP da interface pública de rede;
3. Para cada serviço, baseado na gama de identificativos definida para o mesmo, é verificado a existência de novos *logs* para um *timestamp* superior ao guardado na tabela *Sistema_IDS*, correspondente à última geração de regras;
4. Para cada *log* associado a um padrão de erro com uma prioridade correspondente a um risco elevado, é gerado uma regra de *Firewall* para o serviço e endereços IP – de origem e destino, correspondentes ao atacante e à interface de rede pública visada, respetivamente;

5. Caso tal não se verifique, é realizada a contagem de alertas, por endereço IP de origem, serviço e média de prioridade, e armazenada na tabela *Regras_IDS* (utilizada para a sugestão de regras, apresentada na UI da figura 4.5);

6. Por último, para cada sistema de IDS/IPS, é atualizado o valor da coluna *logs_verifica* com o valor do *timestamp* do último *log* tratado.

No Anexo C.1 pode-se observar a forma de adição de regras de *Firewall*, para o protocolo ICMP, implementada no *script* PHP executado periodicamente.

4.2.3 Interface de Utilizador da aplicação *web* de administração

4.2.3.1 *Front-end*

O *front-end* é responsável por apresentar ao utilizador as informações relativas à *Firewall* – como, por exemplo, as regras existentes, quer estas se encontrem ativas, ou não – e ao sistema de IDS/IPS. No entanto, tendo em conta o contexto desta Dissertação, apenas estão apresentadas as UI relativas ao sistema de IDS/IPS e a sua interação na *Firewall*. Note-se que, por forma a possibilitar uma melhor perceção do conteúdo da UI explorado, estas encontram-se representadas parcialmente.

Na figura 4.1 está representada a UI de gestão de regras da *Firewall*. No contexto do sistema de IDS/IPS, esta disponibiliza as informações relativas à adição de novas regras na *Firewall*, realizada diretamente pelo utilizador – através da interface de gestão de alertas do sistema de IDS/IPS –, mas também pelos mecanismos de auto-adaptação. Assim, ao nível da UI, são apresentadas as várias regras geradas pelo sistema de IDS/IPS, constituídas pelos seguintes parâmetros:

- *State*: por defeito, quando a regra é adicionada, esta fica ativa. No entanto, um utilizador pode simplesmente ativá-la ou, de uma forma permanente, eliminá-la;

- *Descriptive*: este parâmetro é visto como uma *tag*, útil para funções de filtragem, e tem, por defeito, atribuído o valor *Default*;

- *Service*: identifica para que serviço a regra encontra-se a filtrar o tráfego;

- *Action*: identifica qual o destino dos pacotes correspondentes a esta regra, sendo, por defeito, bloqueados;

- *Origin*/*Destination*: identifica a origem e destino, respetivamente, dos pacotes contido no tráfego de rede.

- *Priority*: este parâmetro identifica a posição da regra nas tabelas da *Firewall* IPtables;

- *Logging*: possibilita o registo de pacotes correspondentes à regra em causa que chegaram à *Firewall*;

- *Comments*: este campo é puramente ilustrativo, servindo como um auxiliar ao utilizador para mais facilmente identificar determinada regra.

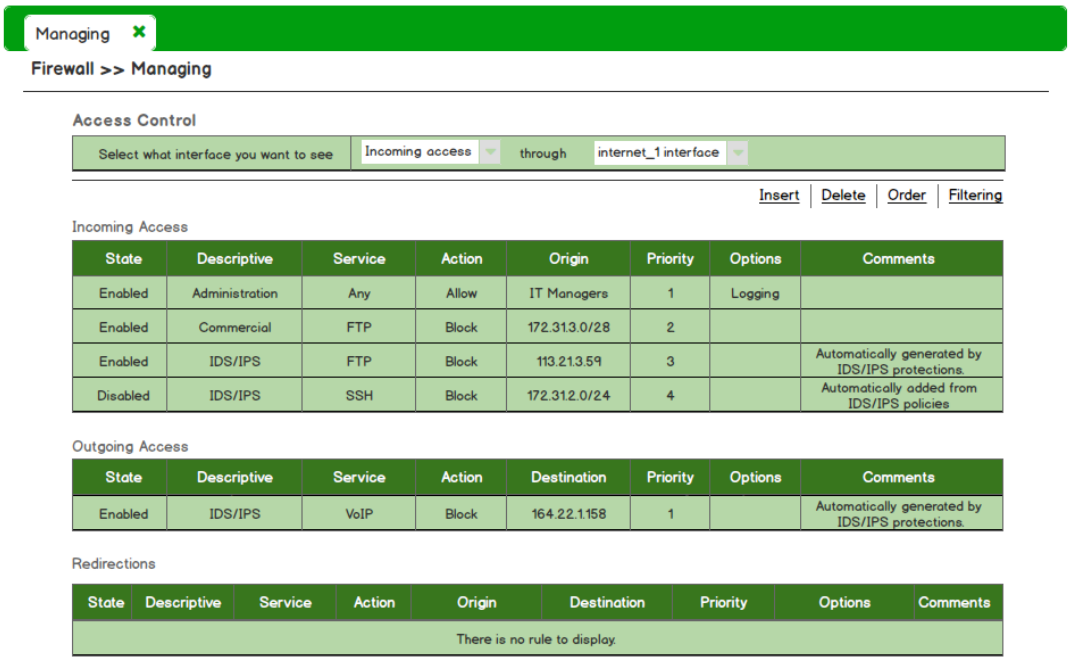


Figura 4.1: Interface de Utilizador de gestão de regras da *Firewall*, na aplicação *web* de administração.

A figura 4.2 é a UI principal do sistema de IDS/IPS, na qual o utilizador pode gerir as suas funcionalidades mais básicas, nomeadamente **(a)** configuração do modo de proteção do sistema de IDS/IPS e serviços monitorizados e **(b)** criar novas alarmísticas, baseadas no modo de proteção, serviços e num nível mínimo de perigo. Em relação aos parâmetros visíveis na figura 4.2, para uma dada interface de rede pública, estes possuem as seguintes funções:

- *State*: identifica o estado de execução do sistema de IDS/IPS, para qualquer uma das ferramentas de IDS/IPS;
- *Mode*: identifica o modo de execução do sistema de IDS/IPS, isto é, *Apenas Alertas* para execução em modo IDS e *Alertas e Prevenção* para execução em modo IPS;
- *Monitored Services*: identifica quais os serviços existentes no servidor são alvos de monitorização por parte do sistema de IDS/IPS;
- *Comments*: este campo é puramente ilustrativo, servindo como um auxiliar ao utilizador para mais facilmente associar as configurações realizadas;
- *Contact*: identifica qual o contato-alvo para envio de notificações de determinada alarmística;

- *Target Mode*: identifica em modo de execução do sistema de IDS/IPS, a alarmística irá enviar notificações;
- *Target Services*: cada alarmística pode notificar até ao máximo de serviços monitorizados pelo sistema de IDS/IPS, pelo que este parâmetro identifica quais são os serviços selecionados para notificação;
- *Minimum Danger Level*: nível mínimo que um dado alerta deve possuir para que seja enviado uma notificação relativa à alarmística em causa.

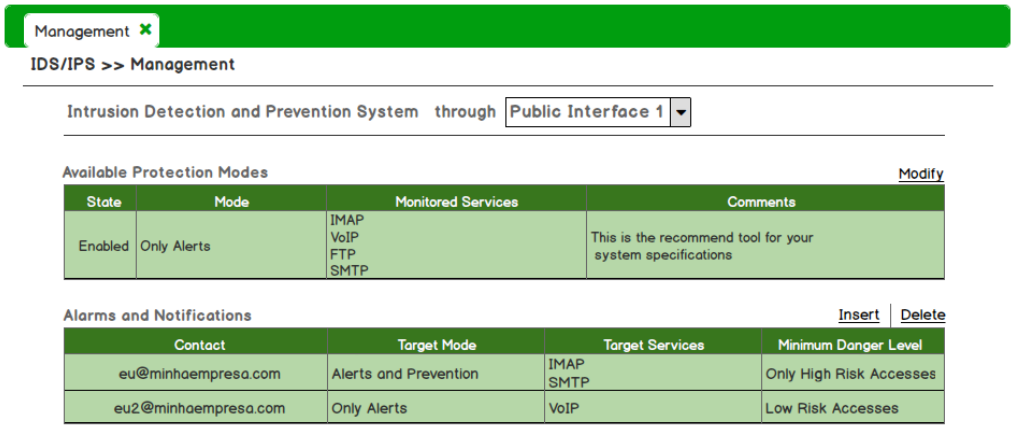


Figura 4.2: Interface de Utilizador principal do sistema de IDS/IPS, na aplicação *web* de administração.

As configurações do sistema de IDS/IPS, apresentadas na figura 4.2, podem ser sujeitas a alterações. Enquanto o sistema de IDS/IPS, para uma dada interface de rede pública, apenas pode ser modificado, as alarmísticas para esse sistema podem ser múltiplas, podendo ser adicionadas, modificadas ou eliminadas. Seguidamente, na figura 4.3, é apresentado o formulário para modificação do *Protection Mode*, no qual estão representados parte dos parâmetros apresentados na UI principal do sistema de IDS/IPS, mas também a adição de um novo parâmetro: *Notifications*. Este parâmetro permite controlar, de uma maneira global, a permissão de envio de notificações definidas pelas alarmísticas quanto à interface de rede pública em causa.

Por outro lado, na figura 4.4, é apresentada a UI relativa à modificação de alarmísticas, baseando num formulário de diversos parâmetros. Esta contém parâmetros apresentados na figura 4.2, apenas sendo adicionado um novo parâmetro: o *Mail Subject*. Este parâmetro é, tal como a sua designação indica, o conteúdo do campo *Assunto* de cada notificação enviada. Note-se que a UI de adição de novas alarmísticas é bastante similar a esta, apenas modificando a identificação da página.

Por último, a UI apresentada na figura 4.5 possibilita a gestão de regras de *Firewall* sugeridas pelos mecanismos de tratamento de *logs* do sistema de IDS/IPS, pelo que o utilizador pode decidir

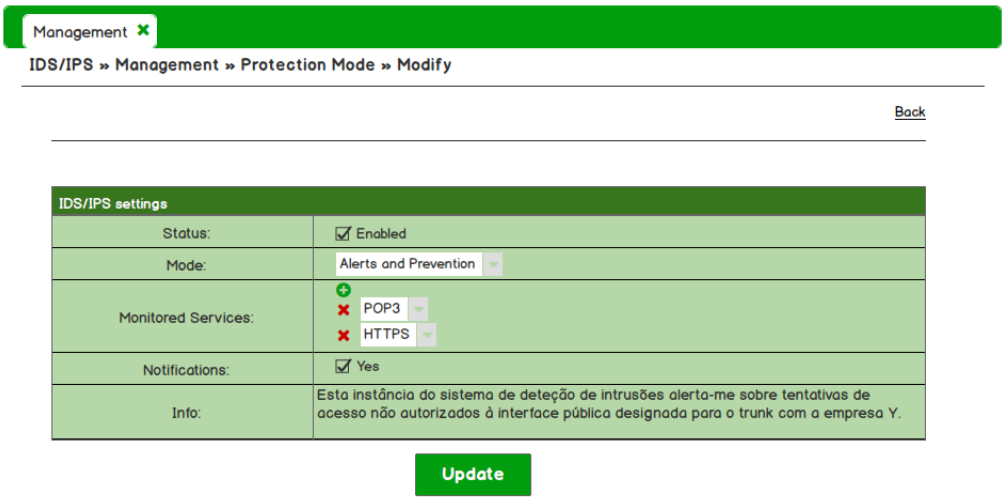


Figura 4.3: Interface de Utilizador para modificação do sistema de IDS/IPS, na aplicação *web* de administração.

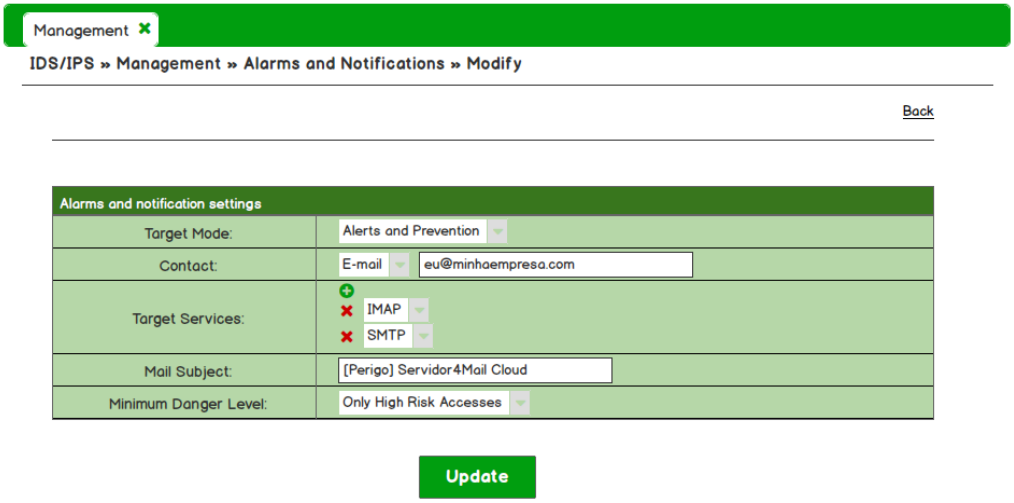


Figura 4.4: Interface de Utilizador para criação ou modificação de alarmísticas do sistema de IDS/IPS, na aplicação *web* de administração.

se pretende adicionar uma nova regra, ou não, à *Firewall*, consoante os parâmetros observados. Esses parâmetros são:

- *Date*: identifica a data e a hora da última tentativa de acesso – com ou sem sucesso;
- *Service*: identifica qual o serviço alvo de tentativas de acessos;
- *Danger Level*: carateriza o grau de perigo que a tentativa de acesso representa, com base na definição de níveis de erro das ferramentas de NIDS/NIPS;

- *Malicious IP*: identifica o endereço IP do atacante, quer este seja um intruso malicioso ou apenas um utilizador negligente;
- *Attemps*: quantifica o número de tentativas de acesso;
- *Actions*: parâmetro que possibilita ao utilizador adicionar, ou não, uma nova regra à *Firewall* que elimina o tráfego relativo ao endereço IP e serviço do *log* em causa.

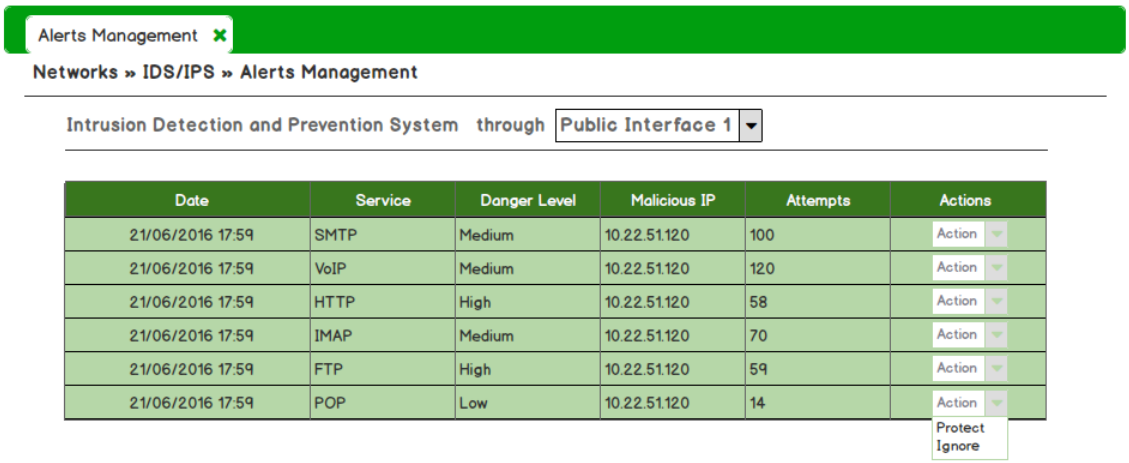


Figura 4.5: Interface de Utilizador para gestão de regras da *Firewall*, sugeridas pelo sistema de IDS/IPS, na aplicação *web* de administração.

4.2.3.2 *Back-end*

O *back-end* é responsável por tratar a informação submetida pelo utilizador, através de formulários existentes na UI da aplicação *web* de administração, mas também de obter a informação a apresentar nesta mesma aplicação. No entanto, tendo em conta o contexto desta Dissertação, apenas estão apresentados conteúdos relativos ao *back-end* do sistema de IDS/IPS e a sua interação na *Firewall*.

INTERLIGAÇÃO ENTRE FIREWALL E SISTEMA DE IDS/IPS. Estas duas componentes, responsáveis pela segurança do servidor, tem uma interação uni-direcional: o sistema de IDS/IPS gera regras que, direta ou indiretamente, são adicionadas à *Firewall*. Esta inclusão de regras de *Firewall*, apresentadas na UI de gestão de regras da *Firewall* (ver figura 4.1) assenta na utilização da biblioteca *LibFirewall.php*, utilizada para efeitos de *middle-end* entre as camadas de alto e baixo-nível da *Firewall*. Esta biblioteca contém funções específicos a cada umas das tabelas da *Firewall*: *Firewall_Basica*, *Enderecos_Especificos* e *Firewall_IPtables*, apresentados no modelo relacional da figura 3.12. Assim, na adição de novas regras, esta biblioteca possui uma função responsável pela:

1. adição de novo tuplo à tabela *Firewall_Basica*, cujos parâmetros definidos, por defeito, para o sistema de IDS/IPS são *tipo_perfil = 3* e *grupo_tag = "IDS/IPS"*, e receber o *id_regra* gerado;

2. adição de novo tuplo à tabela *Enderecos_Especificos*, cujos parâmetros definidos, por defeito, são o *id_regra*, retornado na adição do novo tuplo na tabela *Firewall_Basica*, e o *tipo_endereco = 1*, que implica que se trata de um endereço IP;

3. mapeamento do número de regras IPtables respeitante ao serviço em causa e, consoante isso, adicionar à tabela *Firewall_IPtables* um, ou mais, tuplos, em que o parâmetro definido, por defeito, é o *id_regra*, retornado na adição do novo tuplo na tabela *Firewall_Basica*.

Desta forma, enquanto o *middle-end* ao nível do PHP é realizado pela biblioteca *LibFirewall.php*, ao nível da base de dados, é a coluna *id_regra* que interliga as tabelas das várias camadas da *Firewall*.

MODIFICAÇÃO DO SISTEMA DE IDS/IPS. A modificação das caraterísticas do sistema de IDS/IPS tem implicações na base de dados, mas também no sistema. Na base de dados em causa, são efetuadas modificações sobre as tabelas *Sistema_IDS* e *Servicos_IDS* (ver modelo relacional da figura 3.10). Por outro lado, nas modificações no sistema, são utilizados *scripts* Shell e PHP para a execução das ferramentas de IDS/IPS e armazenamento de dados retornados na base de dados. Assim, o procedimento adotado na modificação do sistema de IDS/IPS é o seguinte:

1. atualização de valores nas tabelas *Sistema_IDS* e *Servicos_IDS* com os dados recolhidos do formulário de modificação do sistema de IDS/IPS – ver figura 4.3;

2. atualização dos ficheiros de configuração das ferramentas de IDS/IPS e modificação do modo de execução do sistema de IDS/IPS. Para tal, são verificados os serviços selecionados pelo utilizador e, através de um *script* Shell, é feita a modificação nos ficheiros de configuração. Além disso, este *script* é igualmente responsável pela modificação do modo de execução das ferramentas de IDS/IPS.

3. armazenamento, na tabela *Sistema_IDS*, do *id* relativo ao processo da ferramenta de NIDS/-NIPS, útil associar a instância de uma ferramenta de NIDS/NIPS à interface de rede pública em causa.

O ponto **1.** trata-se de um desenvolvimento simples, baseado em *queries* à base de dados, executadas no ficheiro PHP, tendo em vista a atualização do sistema de IDS/IPS, de uma dada interface de rede pública, com um dado *id_ids*. Por outro lado, os pontos **2.** e **3.** não possuem uma implementação tão trivial, uma vez que necessita de uma interação entre a aplicação *web* de administração e o sistema, que é realizado com recurso à execução de *scripts* Shell e PHP.

A atualização dos ficheiros de configuração e a reinicialização do sistema de IDS/IPS envolvem a passagem de parâmetros do formulário, a partir do PHP, bem como a atualização dos ficheiros de configuração das ferramentas de IDS/IPS. Na atualização dos serviços monitorizados,

o *script* Shell, *setup_ids.sh*, recebe como parâmetros os serviços selecionados pelo utilizador e a interface de rede em causa. Seguidamente, está exposto uma chamada do *script* Shell, executado para que modifique os serviços monitorizados pelo sistema de IDS/IPS.

```
$ /bin/sh /opt/system/scripts/ids_ips/setup_ids.sh -cs <flag_comenta_tudo> <
    servico> <interface_rede>
```

Por outro lado, a reinicialização de um sistema de IDS/IPS envolve a passagem de um maior número de parâmetros, armazenados na base de dados, nomeadamente:

- modo de execução atual: se houver mudança no modo de execução, pode implicar a adição, caso o novo modo seja IPS, ou eliminação de regras da *Firewall*, caso o modo em vigor seja o IPS. Note-se que este parâmetro pode ser possuir valor nulo, caso o sistema de IDS/IPS esteja desativado até ao momento da modificação;
- ferramenta de NIDS/NIPS em execução: identifica qual a ferramenta de NIDS/NIPS em vigor e utilizará o comando respetivo para inicialização da mesma. Tal como o modo de execução atual, pode possuir caráter nulo nas mesmas circunstâncias;
- *id* do processo da ferramenta de IDS/IPS em execução: é utilizado para matar o processo. Pode, igualmente, possuir valor nulo nas circunstâncias dos dois parâmetros anteriores;
- designação da interface de rede pública: é utilizada como parâmetro para o comando de inicialização de um novo processo da ferramenta de NIDS/NIPS.

Ao nível do *script* Shell, a sua execução é realizada da seguinte forma

```
$ /bin/sh /opt/system/scripts/ids_ips/setup_ids.sh -ct <modo_atual> <modo_novo>
    <interface_rede> <ferramenta> <pid_ferramenta>
```

Note-se que, após finalização dos comandos do *script* Shell em questão, este executa um *script* PHP, responsável por atualizar a tabela *Sistema_IDS* relativamente ao valor da coluna *processo_id*.

ALARMÍSTICAS DO SISTEMA DE IDS/IPS. A criação de novas alarmísticas para o sistema de IDS/IPS tem implicações nas bases de dados, quer da aplicação *web* de administração, como também dos *logs* das ferramentas de IDS/IPS. No entanto, ao contrário Snort e Suricata não possuem um mecanismo de envio de *e-mails* por defeito, pelo que é necessário comparar os parâmetros guardados para cada alarmística e a existência de novos *logs* das ferramentas de IDS/IPS de uma forma periódica. Para tal, é necessário, em cada alarmística existente:

- comparar o modo de execução do sistema de IDS/IPS com o definido pela alarmística;
- verificar a existência de novos *logs*, ainda não verificados, cujo grau de perigo é maior, ou igual, ao definido no formulário da figura 4.4;
- se as condições anteriores implicarem a necessidade de notificação, é enviado um *e-mail* com o assunto definido para a alarmística em causa;

- atualiza a coluna *notif_verifica* da tabela *Firewall_Basica*, da instância do sistema de IDS/IPS em causa, com o *timestamp* do último *log* verificado.

Note-se que, por forma a verificar a necessidade de notificações de forma periódica, foi inserido um *cronjob* que executa um *script* PHP, útil para fazer as operações ao nível da base de dados, e que executa um *script* Shell para o envio de um novo *e-mail*. A chamada do *script* Shell no PHP é realizada da seguinte forma

```
1 <?php
2   $command = "mail -s '".$mail_assunto."' ".$alarmistica_contacto." <<< '
      There are a possibility of new attacks against your Cloud server.'";
3   execAsRoot($command,$sending_result);
4 ?>
```

GERAÇÃO DE REGRAS NA *Firewall*, SUGERIDAS PELO SISTEMA DE IDS/IPS, POR AÇÃO DO UTILIZADOR. Por último, esta geração acionada pelo utilizador resulta das suas configurações ao nível da UI da figura 4.5, na qual é apresentada um conjunto de campos, retirados dos *logs* do sistema de IDS/IPS, úteis para identificar se as ações são, ou não, maliciosas. Consoante isso, o utilizador pode gerar uma nova regra ou simplesmente ignorar a sugestão.

A geração de uma nova regra na *Firewall* utiliza, uma vez mais, a biblioteca *LibFirewall.php*, que, neste caso, é responsável por efetuar a adição de novas regras nas camadas de alto e baixo-nível da *Firewall*, o que resulta em novos tuplos nas tabelas *Firewall_Basica*, *Firewall_IPtables* e *Enderecos_Especificos*. Por outro lado, se o utilizador optar por não adicionar a regra na *Firewall*, esta sugestão é apagada da base de dados.

Capítulo 5

Testes e Resultados

Neste Capítulo estão expostos os testes realizados à solução desenvolvida no contexto desta Dissertação, bem como a análise de resultados e conclusões sobre estes. Desta forma, procura-se avaliar a solução desenvolvida quanto aos objetivos para os quais esta Dissertação foi proposta.

Por se tratar de uma solução que visa a segurança, mas também a otimização do desempenho do sistema, foram realizados dois tipos de teste, nomeadamente **(i)** testes funcionais, responsáveis por verificar se a solução cumpre os objetivos ao nível da segurança, e **(ii)** testes de impacto, responsáveis por verificar o grau de influência, positivo, neutro ou negativo, dos mecanismos de auto-adaptação introduzidos.

5.1 Introdução

Após a implementação da solução idealizada, é necessária a realização de testes para analisar o grau de satisfação dos objetivos propostos nesta Dissertação. Desta forma, a solução deve ser avaliada quanto a quatro aspetos essenciais: o funcionamento do sistema de IDS/IPS, a viabilidade na interação entre o sistema de IDS/IPS e a *Firewall* e a alternância automática de ferramentas de NIDS/NIPS e da quantidade de padrões de erro ativos.

O funcionamento do sistema de IDS/IPS passa pela verificação de deteção de ataques, de forma controlada, pelas ferramentas de IDS/IPS. Trata-se, por isso, de um teste de caráter funcional, que procura perceber se cada ferramenta de IDS/IPS encontra-se a realizar a deteção e prevenção de ataques.

A viabilidade na interação entre o sistema de IDS/IPS e a *Firewall* engloba dois tipos de teste: testes funcionais e de impacto. Os testes funcionais contemplam a verificação da adição de regras na *Firewall* pelo sistema de IDS/IPS, qualquer que seja a camada da mesma. Por outro lado, os testes de impacto têm por objetivo traduzir o grau de influência que a geração automática de regras na *Firewall* tem no desempenho do sistema.

Por último, a alternância automática de ferramenta de NIDS/NIPS e da quantidade de padrões de erro tem por objetivo perceber se a variação do desempenho do sistema possibilita o aumento ou diminuição da segurança e se implica a alteração da ferramenta de NIDS/NIPS em execução.

Por esse motivo, para as alternâncias da ferramenta de NIDS/NIPS e do nível de padrões de erro serão realizados testes funcionais, comprovando o funcionamento dos mecanismos em causa.

5.2 Testes Funcionais

5.2.1 Fiabilidade das ferramentas de IDS/IPS

As ferramentas de IDS/IPS são um dos objetivos desta Dissertação, pelo que foram realizados testes de ataque controlados ao sistema na qual estas se encontram em execução por forma a comprovar a eficácia das ferramentas na deteção e prevenção de ataques. Uma vez que o sistema de IDS/IPS é composto por duas ferramentas de IDS/IPS em execução – o OSSEC, para garantir a segurança do sistema, e o Snort ou o Suricata, para garantir a segurança do tráfego de rede. Por outro lado, as ferramentas de NIDS/NIPS possuem dois modos de execução: IDS, em que a ferramenta apenas alerta possíveis ataques, e IPS, que complementa a ferramenta com a possibilidade de rejeitar ou aceitar pacotes, com base nos padrões de erro.

Deste modo, para as ferramentas de NIDS/NIPS, foram realizados ataques controlados a serviços específicos, nos dois modos de execução, e quanto ao OSSEC, foram realizados testes de alterações no sistema. Seguidamente, serão expostos cada um deles. Note-se que, para as ferramentas de NIDS/NIPS, por forma a testar vários serviços, nomeadamente o acesso às aplicações *web* e envio/receção de *pings*, foram adicionados os dois padrões de erro, expostos nas secções respetivas às ferramentas referidas.

5.2.1.1 Execução em modo IDS

Por forma a testar a viabilidade do modo IDS do Snort e Suricata, foram adicionados os padrões de erro expostos seguidamente. Como componentes do teste, foram utilizadas duas máquinas: a *cliente*, responsável por envio de pacotes, com endereço IP 192.168.1.154; e o *servidor*, na qual o sistema de IDS/IPS está em execução, responsável por alertar o utilizador através de *logs*.

```
alert tcp 192.168.1.154 any -> $HOME_NET [80,443] (msg:"Testing
Access HTTP"; sid:99990; rev:1;)
alert icmp 192.168.1.154 any -> $HOME_NET any (msg:"Testing Ping
Reception"; itype:8; sid:99991; rev:1;)
```

Para que fossem gerados os alertas especificados, foram utilizados os seguintes comandos, respetivamente:

```
1 $ wget https://192.168.1.199 -t 2 -T 4 --no-check-certificate
2 $ ping -c 5 192.168.1.199
```

Nas figuras 5.1 e 5.2, estão expostos o *output* dos testes no *cliente*, que demonstra, tal como o esperado, o sucesso da comunicação com o *servidor*.

```
hugofonseca@ubuntu:~$ wget https://192.168.1.199 -t 2 -T 4 --no-check-certificate
--2016-06-25 01:28:56--  https://192.168.1.199/
A conectar 192.168.1.199:443... conectado.
AVISO: cannot verify 192.168.1.199's certificate, issued by 'CN=ipbrick.domain.com,O=IPBrick,L=Porto,ST=Porto,C
=PT':
  Encontrado certificado auto-assinado.
    AVISO: certificate common name 'ipbrick.domain.com' doesn't match requested host name '192.168.1.199'.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 1626 (1,6K) [text/html]
Saving to: 'index.html'

index.html                  100%[=======================================>]   1,59K  --.-KB/s    in 0s

2016-06-25 01:28:56 (14,4 MB/s) - 'index.html' saved [1626/1626]
```

Figura 5.1: *Output* do teste realizado às ferramentas Snort e Suricata, em modo IDS, por forma a testar o alerta de pedidos *HTTP*.

```
hugofonseca@ubuntu:~$ ping -c 5 192.168.1.199
PING 192.168.1.199 (192.168.1.199) 56(84) bytes of data.
64 bytes from 192.168.1.199: icmp_seq=1 ttl=64 time=0.410 ms
64 bytes from 192.168.1.199: icmp_seq=2 ttl=64 time=0.832 ms
64 bytes from 192.168.1.199: icmp_seq=3 ttl=64 time=0.275 ms
64 bytes from 192.168.1.199: icmp_seq=4 ttl=64 time=0.565 ms
64 bytes from 192.168.1.199: icmp_seq=5 ttl=64 time=0.336 ms

--- 192.168.1.199 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.275/0.483/0.832/0.200 ms
```

Figura 5.2: *Output* do teste realizado às ferramentas Snort e Suricata, em modo IDS, por forma a testar o alerta de pedidos *ICMP*.

SURICATA. Na figura 5.3 estão expostos os alertas gerados pelo Suricata, que, tal como esperado, apenas gera um alerta sobre o pacote correspondente, permitindo a entrada dos pacotes no servidor.

```
ipbrick:~# tail -f /var/log/suricata/fast.log
06/25/2016-09:29:57.103811  [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Priority: 3]
 {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:29:58.101679  [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Priority: 3]
 {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:29:59.099997  [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Priority: 3]
 {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:30:00.098831  [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Priority: 3]
 {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:30:01.097967  [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Priority: 3]
 {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:30:10.235922  [**] [1:99990:1] Testing Access HTTP/HTTPS [**] [Classification: (null)] [Priority:
 3] {TCP} 192.168.1.154:39392 -> 192.168.1.199:443
06/25/2016-09:30:26.690429  [**] [1:99990:1] Testing Access HTTP/HTTPS [**] [Classification: (null)] [Priority:
 3] {TCP} 192.168.1.154:39394 -> 192.168.1.199:443
```

Figura 5.3: *Output* gerado na ferramentas Suricata, em modo IDS, por forma a comprovar a geração de alertas a pedidos *ICMP* e *HTTP*.

SNORT. Na figura 5.4 estão expostos os alertas gerados pelo Snort, que, tal como esperado e similarmente ao Suricata, apenas gera um alerta sobre o pacote correspondente, permitindo a entrada dos pacotes no servidor.

```
tpbrick:~# tail -f /var/log/snort/alert
[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:51:05.959273 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:63095 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6558   Seq:1  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:51:06.960040 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:63218 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6558   Seq:2  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:51:07.959821 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:63383 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6558   Seq:3  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:51:08.960774 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:63577 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6558   Seq:4  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:51:09.960567 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:63825 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6558   Seq:5  ECHO

[**] [1:99990:1] Testing Access HTTP [**]
[Priority: 0]
06/25-09:51:16.247822 192.168.1.154:39400 -> 192.168.1.199:443
TCP TTL:64 TOS:0x0 ID:61855 IpLen:20 DgmLen:60 DF
******S* Seq: 0x86D76856  Ack: 0x0  Win: 0x7210  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 17471945 0 NOP WS: 7
```

Figura 5.4: *Output* gerado na ferramentas Snort, em modo IDS, por forma a comprovar a geração de alertas a pedidos *ICMP* e *HTTP*.

5.2.1.2 Execução em modo IPS

Por forma a testar a viabilidade do modo IPS do Snort e Suricata, foram adicionados os padrões de erro expostos seguidamente. Como componentes do teste, foram utilizadas as duas máquinas referidas na secção 5.2.1.1.

```
drop tcp 192.168.1.154 any -> $HOME_NET [80,443] (msg:"Testing
Access HTTP"; sid:99990; rev:1;)
drop icmp 192.168.1.154 any -> $HOME_NET any (msg:"Testing Ping
Reception"; itype:8; sid:99991; rev:1;)
```

Para que fossem gerados os alertas especificados, foram utilizados os seguintes comandos, respetivamente:

```
1 $ wget https://192.168.1.199 -t 2 -T 4 --no-check-certificate
2 $ ping -c 5 192.168.1.199
```

Nas figuras 5.5 e 5.6, estão expostos o *output* dos testes no *cliente*, que demonstra, tal como o esperado, o insucesso da comunicação com o *servidor*, uma vez que Snort e Suricata rejeitam os pacotes *ICMP* e *TCP*, nos portos 80 e 443, oriundos do endereço IP do *cliente*.

SURICATA. Na figura 5.7 estão expostos os alertas gerados pelo Suricata, que, tal como esperado, gera um alerta sobre o pacote correspondente e não permite a entrada dos pacotes no servidor.

```
hugofonseca@ubuntu:~$ wget https://192.168.1.199 -t 2 -T 4
--2016-06-25 01:26:40--  https://192.168.1.199/
A conectar 192.168.1.199:443... falhou: A ligação expirou.
A tentar novamente.

--2016-06-25 01:26:45--  (tentativa: 2)  https://192.168.1.199/
A conectar 192.168.1.199:443... falhou: A ligação expirou.
A desistir.
```

Figura 5.5: *Output* do teste realizado às ferramentas Snort e Suricata, em modo IPS, por forma a testar o alerta de pedidos *HTTP*.

```
hugofonseca@ubuntu:~$ ping -c 5 192.168.1.199
PING 192.168.1.199 (192.168.1.199) 56(84) bytes of data.

--- 192.168.1.199 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4032ms
```

Figura 5.6: *Output* do teste realizado às ferramentas Snort e Suricata, em modo IPS, por forma a testar o alerta de pedidos *ICMP*.

```
ipbrick:~# tail -f /var/log/suricata/fast.log
06/25/2016-09:21:12.215204  [Drop] [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Prior
ity: 3] {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:21:13.217574  [Drop] [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Prior
ity: 3] {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:21:14.221380  [Drop] [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Prior
ity: 3] {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:21:15.225920  [Drop] [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Prior
ity: 3] {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:21:16.230578  [Drop] [**] [1:99991:1] Testing Ping Reception [**] [Classification: (null)] [Prior
ity: 3] {ICMP} 192.168.1.154:8 -> 192.168.1.199:0
06/25/2016-09:21:28.412286  [Drop] [**] [1:99990:1] Testing Access HTTP/HTTPS [**] [Classification: (null)] [Pr
iority: 3] {TCP} 192.168.1.154:39374 -> 192.168.1.199:443
06/25/2016-09:21:28.663036  [Drop] [**] [1:99990:1] Testing Access HTTP/HTTPS [**] [Classification: (null)] [Pr
iority: 3] {TCP} 192.168.1.154:39376 -> 192.168.1.199:443
```

Figura 5.7: *Output* gerado na ferramenta Suricata, em modo IPS, por forma a comprovar a geração de alertas a pedidos *ICMP* e *HTTP*.

SNORT. Na figura 5.8 estão expostos os alertas gerados pelo Snort, que, tal como esperado e similarmente ao Suricata, gera um alerta sobre o pacote correspondente e não permite a entrada dos pacotes no servidor.

5.2.1.3 OSSEC

Por forma a testar a viabilidade do OSSEC, foi realizado um teste que resultou na modificação de um ficheiro do sistema, nomeadamente o */var/log/snort/alert*, sendo este monitorizado pelo OSSEC. Além disso, tendo em conta a necessidade de modificação do ficheiro de registo de *logs* do Snort, foram utilizadas as duas máquinas referidas na secção 5.2.1.1.

Seguidamente, na figura 5.9, pode-se observar o alerta gerado após modificação de um ficheiro monitorizado pelo OSSEC.

```
ipbrick:~# tail -f /var/log/snort/alert
[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:57:12.296604 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:37975 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6603   Seq:1  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:57:13.298888 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:38200 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6603   Seq:2  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:57:14.302765 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:38287 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6603   Seq:3  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:57:15.307256 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:38399 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6603   Seq:4  ECHO

[**] [1:99991:1] Testing Ping Reception [**]
[Priority: 0]
06/25-09:57:16.311251 192.168.1.154 -> 192.168.1.199
ICMP TTL:64 TOS:0x0 ID:38432 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6603   Seq:5  ECHO

[**] [1:99990:1] Testing Access HTTP [**]
[Priority: 0]
06/25-09:57:27.111664 192.168.1.154:39404 -> 192.168.1.199:443
TCP TTL:64 TOS:0x0 ID:55811 IpLen:20 DgmLen:60 DF
******S* Seq: 0x4E2BDF36  Ack: 0x0  Win: 0x7210  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 17564981 0 NOP WS: 7
```

Figura 5.8: *Output* gerado na ferramentas Snort, em modo IPS, por forma a comprovar a geração de alertas a pedidos *ICMP* e *HTTP*.

```
home:~# tail -f /var/ossec/logs/alerts/alerts.log
2016 Jun 15 17:34:46 home->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/snort/snort.conf'
Size changed from '26429' to '26498'
Old md5sum was: 'f35d70e8b9fed913341c30aeacc402de'
New md5sum is : '870fde8dc02a92221b708c3c90057242'
Old sha1sum was: 'c67d1761ed0ea0b20904ad83f1ad3b4790cff8c7'
New sha1sum is : 'fd174d039e9ea7dd5a7939a41faf292e71d86824'
```

Figura 5.9: *Output* gerado na ferramenta OSSEC, após a modificação de um ficheiro monitorizado pela mesma.

5.2.2 Geração de Regras na *Firewall* pelo Sistema de IDS/IPS

Os testes realizados à funcionalidade de geração de regras na *Firewall*, automática e manualmente, pelo sistema de IDS/IPS, através dos *logs* e alertas gerados por este, tem como objetivo comprovar a eficácia da sua implementação, descrita na secção 4.2.2. No entanto, esta geração de regras possui dois modos de implementação das regras – nomeadamente, com e sem interação do utilizador – pelo que são necessários testes a cada um destes modos. Para tal, foram configurados os seguintes ambientes de teste:

- Sem interação do utilizador – introdução de um padrão de erro no sistema de IDS/IPS, cuja prioridade é de risco elevado, para um endereço IP e serviço específicos. Para verificação do

sucesso, ou não, deste teste, são consultados os constituintes da *firewall* IPtables e as tabelas de *Firewall* na base de dados;

- Com interação do utilizador – Introdução de um padrão de erro no sistema de IDS/IPS, cuja prioridade é de risco médio-baixo, para um endereço IP e serviço específicos.Para verificação do sucesso, ou não, deste teste, são apenas consultadas as tabelas de *Firewall* na base de dados.

Para realizar este teste, foram necessárias duas máquinas: *máquina_1*, responsável por se comportar como atacante, que possui o endereço IP 172.28.1.154; e a *máquina_2*, que possui o sistema de IDS/IPS ativo, com o endereço IP 172.28.1.140. Por forma a aceder à *máquina_1* e gerar alertas no sistema de IDS/IPS, foi executado o seguinte comando pela *máquina_1*:

```
1 $ ping -f 172.28.1.140
```

Este envia pacotes *icmp* para a *máquina_2*, com um *payload* de 50000 bytes. Seguidamente, são expostas as metodologias e resultados obtidos na *máquina_2*, para cada um dos ambientes de teste referidos.

Por outro lado, por forma a testar este mecanismo de forma controlada, foi executado o *script* PHP, ao invés de o utilizar através de um *cronjob*:

```
$ php5 /opt/system/scripts/ids_ips/generate_firewall_rules.php
```

SEM INTERAÇÃO DO UTILIZADOR.

Por forma a que o sistema de IDS/IPS gerasse e implementasse automaticamente uma regra de *Firewall*, foi adicionado o seguinte padrão de erro às ferramentas de NIDS/NIPS:

```
alert icmp !$HOME_NET any -> $HOME_NET any (msg:"Ping Flood Attack";
 itype:8; classtype:denial-of-service; sid:99999; rev:1;)
```

Este padrão de erro é responsável por detetar um ataque denominado *Ping Flood*, para pacotes oriundos do endereço IP 172.168.1.154 (*máquina_1*), cujo grau de perigo é dado pelo valor de *classtype* (neste caso, é equivalente ao maior grau de perigo, que corresponde a uma prioridade = 1). Deste modo, o sistema de IDS/IPS gera uma regra de *Firewall* para impedir a entrar de pacotes no sistema.

Assim, tal como se pode observar nas figuras 5.10- 5.13, este mecanismo teve impacto no sistema, implementando automaticamente a regra na *Firewall* IPtables, e na base de dados, adicionando a regra às várias camadas da *Firewall*, por forma a ser apresentada na aplicação *web* de administração.

```
srv59:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
DROP       icmp --  172.28.1.154         srv59.srv59.linuxfacil.pt  icmp echo-request
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  172.28.1.0/24        172.28.1.0/24
ACCEPT     tcp  --  anywhere             anywhere            state RELATED,ESTABLISHED tcp spt:ftp
ACCEPT     tcp  --  anywhere             anywhere            state RELATED,ESTABLISHED tcp spt:ftp-data
ACCEPT     tcp  --  anywhere             anywhere            state RELATED,ESTABLISHED tcp spts:1024:65535 dpts:1024:65535
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  10.0.0.253           10.0.0.253

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere            state INVALID
DROP       udp  --  anywhere             anywhere            udp dpt:ms-sql-m
ACCEPT     all  --  172.28.1.0/24        172.28.1.0/24
```

Figura 5.10: Regra inserida diretamente no sistema, nomeadamente na *Firewall* IPtables.

```
 id_regra | estado | tipo_perfil | grupo_tag | accao | interface_entrada | interface_saida | logging |            info_adicional
----------+--------+-------------+-----------+-------+-------------------+-----------------+---------+--------------------------------------
-
       39 | t      |           3 | IDS/IPS   |     1 |                 1 |                 |         | Automatically added by IDS/IPS system.
```

Figura 5.11: Regra inserida na base de dados, na camada de alto-nível da *Firewall*, para posterior visualização na aplicação *web* de administração.

```
 id_endereco |   endereco    | orig_dest | tipo_endereco | regra_basica
-------------+---------------+-----------+---------------+--------------
          62 | 172.28.1.154  |         0 |             0 |           39
```

Figura 5.12: Endereço IP de origem da regra inserida na base de dados, que interliga à camada de alto-nível da *Firewall*, para posterior visualização na aplicação *web* de administração.

```
 idfirewall | tipo | regra | interfacei | interfaceo | protocolo | notorigem |   iporigem   | masorigem | portoorigem | notdestino |  ipdestino   | masdestino | portodestino | polit
ica | redirectporto | outro | estado | ordem | modulo | tabela | notmacsource | macsource | regra_basica
------------+------+-------+------------+------------+-----------+-----------+--------------+-----------+-------------+------------+--------------+------------+--------------+------
----+---------------+-------+--------+-------+--------+--------+--------------+-----------+--------------
        257 |    1 | INPUT | eth1       |            | icmp      |           | 172.28.1.154 |           |             |            | 172.28.1.140 |            |              | ACCEP
T   |               |       | t      |  1023 |        | filter |              |           |          39
```

Figura 5.13: Regra inserida na base de dados, na camada de baixo-nível da *Firewall*, para posterior visualização na aplicação *web* de administração.

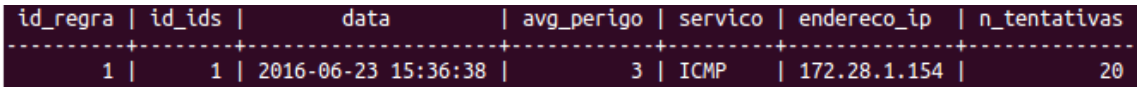
COM INTERAÇÃO DO UTILIZADOR.

Por outro lado, para que o sistema de IDS/IPS gerasse a regra de *Firewall*, sem que a implementasse automaticamente, foi realizada a seguinte alteração no padrão de erro:

```
alert icmp !$HOME_NET any -> $HOME_NET any (msg:"Ping Flood Attack";
 itype:8; classtype:denial-of-service; priority:3; sid:99999; rev:1;)
```

Neste caso, a existência do campo *priority* sobrepõe-se ao *classtype*, pelo que o grau de perigo deste ataque terá valor 3, pelo que não será implementada automaticamente a regra na *Firewall*.

Assim, tal como se pode observar na figura 5.14, este teste resultou na adição dos campos necessários para apresentação na interface da aplicação *web* de administração (ver figura 4.5), na qual o utilizador pode gerir o que pretende adicionar à *Firewall*.



id_regra	id_ids	data	avg_perigo	servico	endereco_ip	n_tentativas
1	1	2016-06-23 15:36:38	3	ICMP	172.28.1.154	20

Figura 5.14: Regra inserida na base de dados, para posterior manipulação (adição à *Firewall* ou eliminação) através da aplicação *web* de administração.

5.2.3 Alternância da ferramenta de NIDS/NIPS

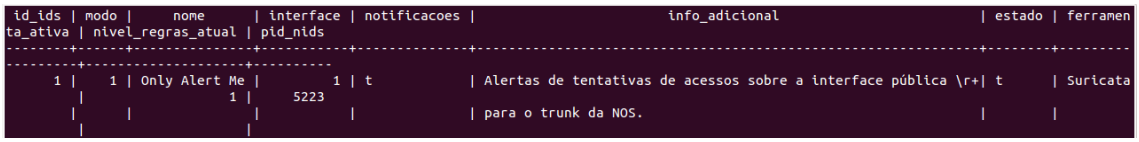
O teste realizado à alteração automática da ferramenta de NIDS/NIPS do sistema de IDS/IPS tem como objetivo comprovar a eficácia do mecanismo implementado, especificado na secção 4.2.1.2.

Para que se observasse este mecanismo em execução, foi configurado um cenário de teste que, de forma controlada, possibilitasse a alteração da ferramenta de NIDS/NIPS em execução. Desta forma, ao invés de colocar o mecanismo por um *cronjob*, foi executado manualmente, pelo comando exposto em seguida, responsável por executar um *script* PHP que verifica o conteúdo guardado na base de dados, obter os dados do sistema e, consoante isso, iniciar ou modificar a ferramenta de NIDS/NIPS em execução.

```
$ php5 /opt/system/scripts/ids_ips/ids_bootsystem.php
```

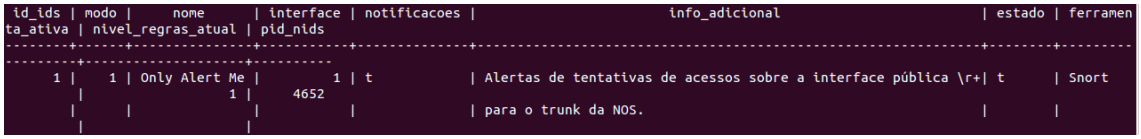
Para além disso, por forma a forçar a obtenção de resultados, foram configurados os seguintes ambientes de teste, com os seguintes resultados esperados:

- Ambiente A: 4096 MB de memória RAM e 4 processadores, na qual é expectável que seja o Suricata a ferramenta de NIDS/NIPS selecionada para execução;
- Ambiente B: 2096 MB de memória RAM e 2 processadores, na qual é expectável que seja o Snort a ferramenta de NIDS/NIPS selecionada para execução.



id_ids	modo	nome	interface	notificacoes	info_adicional	estado	ferramenta_ativa	nivel_regras_atual	pid_nids
1	1	Only Alert Me	1	t	Alertas de tentativas de acessos sobre a interface pública \r+| para o trunk da NOS.	t	Suricata	1	5223

Figura 5.15: Resultados obtidos após execução do *script* PHP, com o ambiente A implementado.



id_ids	modo	nome	interface	notificacoes	info_adicional	estado	ferramenta_ativa	nivel_regras_atual	pid_nids
1	1	Only Alert Me	1	t	Alertas de tentativas de acessos sobre a interface pública \r+| para o trunk da NOS.	t	Snort	1	4652

Figura 5.16: Resultados obtidos após execução do *script* PHP, com o ambiente B implementado.

5.2.4 Níveis de padrões de erro do sistema de IDS/IPS

O teste realizado à alteração automática do nível de padrões do sistema de IDS/IPS tem como objetivo comprovar a eficácia do mecanismo implementado, especificado na secção 4.2.1.3.

Para que se observasse este mecanismo em execução, foi configurado um cenário de teste que, de forma controlada, possibilitasse a alteração do nível de padrões de erro. Desta forma, ao invés de colocar o mecanismo por um *cronjob*, foi executado manualmente, pelo comando exposto em seguida, semelhante ao comando do mecanismo de alternância das ferramentas de IDS/IPS, mas que contém um *script* Shell dedicado à seleção do nível de padrões de regras. Além disso, este *script* PHP é responsável pela reinicialização da ferramenta de NIDS/NIPS, bem como armazenar os novos valores na base de dados.

```
$ php5 /opt/system/scripts/ids_ips/ids_bootsystem.php
```

Por forma a forçar a obtenção de resultados, foram configurados os seguintes ambientes de teste, com os seguintes resultados esperados:

- Ambiente A: 4096 MB de memória RAM, na qual é expectável que seja selecionado ambos os níveis de padrões de erro;
- Ambiente B: 1024 MB de memória RAM, na qual é expectável que seja selecionado o nível mínimo de nível de padrões de erro.

```
 id_ids | modo |     nome      | interface | notificacoes |                      info_adicional                       | estado | ferramen
ta_ativa | nivel_regras_atual | pid_nids
--------+------+---------------+-----------+--------------+-----------------------------------------------------------+--------+---------
---------+--------------------+----------
      1 |    1 | Only Alert Me |         1 | t            | Alertas de tentativas de acessos sobre a interface pública \r+| t      | Suricata
        |                    1 |      5223
        |      |               |           |              | para o trunk da NOS.                                      |        |
        |                    |
```

Figura 5.17: Resultados obtidos após execução do *script* PHP, com o ambiente A implementado, para a verificação e possível alteração do nível de padrões de erro.

```
 id_ids | modo |     nome      | interface | notificacoes |                      info_adicional                       | estado | ferramen
ta_ativa | nivel_regras_atual | pid_nids
--------+------+---------------+-----------+--------------+-----------------------------------------------------------+--------+---------
---------+--------------------+----------
      1 |    1 | Only Alert Me |         1 | t            | Alertas de tentativas de acessos sobre a interface pública \r+| t      | Suricata
        |                    2 |      4958
        |      |               |           |              | para o trunk da NOS.                                      |        |
        |                    |
```

Figura 5.18: Resultados obtidos após execução do *script* PHP, com o ambiente B implementado, para a verificação e possível alteração do nível de padrões de erro.

5.3 Testes de Impacto

5.3.1 Geração e implementação automática de regras de *Firewall*

Por último, por forma a medir o impacto no desempenho do sistema pela geração e implementação automática de regras de *Firewall*, foram comparados os desempenhos do sistema com e sem

geração e implementação automática de regras, com a existência de pontos de ataque. Note-se que, por forma a obter a variância do desempenho quer ao nível da ferramenta de IDS/IPS, mas também do sistema, utilizou-se o desempenho do sistema apenas com a ferramenta de IDS/IPS em execução, sem que exista quaisquer pontos de ataque.

O ambiente de testes configurado é muito similar ao referido na secção 5.2.2, sendo que, por forma a simular múltiplas fontes de ataques, a *máquina_1* (atacante) foi configurada com múltiplos endereços IP, alterados iterativamente. Além disso, tendo em conta o objetivo deste teste, foi apenas considerada a ferramenta de IDS/IPS Snort, executada sobre uma plataforma com 4096 MB de memória RAM e 4 processadores, com apenas um padrão de erro em execução.

Ambiente de teste	**Tipo de Memória**					**Processamento**			
	Sistema Geral				Snort	Sistema Geral			Snort
	Livre	Usada	*Buffer*	*Cache*		Util.	Sist.	Livre	
Base	2923 MB	728 MB	34 MB	269 MB	1.9%	0.0%	0.0%	99.9%	0.0%
S/ geração	2968 MB	753 MB	24 MB	210 MB	1.9%	0.0%	1.3%	98.7%	1.8%
C/ geração	3017 MB	730 MB	27 MB	180 MB	1.9%	0.0%	0.5%	99.4%	0.5%

Tabela 5.1: Desempenho médio na memória e no CPU, com e sem geração e implementação automática de regras na *Firewall*.

5.4 Tratamento de resultados e conclusões

5.4.1 Testes Funcionais

A realização dos testes funcionais permitiu concluir sobre o bom funcionamento do *back-end* da solução desenvolvida, quer ao nível do sistema, quer ao nível da aplicação *web* de administração.

Ao nível das ferramentas de IDS/IPS utilizadas (Snort, Suricata e OSSEC), ambos os modos suportados por cada uma delas ficou comprovado pela geração de *logs* aos testes realizados. Deste modo, pode-se concluir que estas encontram-se operacionais para a verificação do tráfego de rede que entra e sai do servidor, bem como as modificações existentes no sistema.

Posteriormente, verificou-se o sucesso da geração de regras de *Firewall*, quaisquer que seja o modo (implementação manual, através da aplicação *web* de administração, ou automática).

Por último, o dinamismo da ferramenta de NIDS/NIPS e do nível de padrões de erro em execução, foi eficazmente comprovado, indo de encontro às conclusões referidas na secção 3.2.4.

5.4.2 Testes de Impacto

Os resultados obtidos para o teste de impacto, comparativamente aos observados na secção 3.2, refletem uma pequena amostra do que se pode esperar da execução desta solução num ambiente real.

Por outro lado, tal como seria esperado, houve uma diminuição da carga introduzida nos processadores pela ferramenta de IDS/IPS, uma vez que a *Firewall* IPtables não reencaminhou o tráfego de chegada para a ferramenta. Tal deveu-se à introdução da regra gerada previamente às regras de redirecionamento do tráfego para o *userspace*.

Contudo, a diminuição de carga de processamento pela ferramenta de IDS/IPS teria uma consequência que também foi alvo de avaliação: o aumento de carga no processador pela *Firewall* IPtables, uma vez que esta seria responsável pela aceitação ou rejeição dos pacotes que seriam reencaminhada para a ferramenta de NIDS/NIPS. No entanto, tal como se pode observar na tabela 5.1, o poder de processamento utilizada pelo utilizador e sistema foi diminuído pela introdução da geração e implementação de regras de *Firewall*.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Satisfação dos Objectivos

Tendo em conta os objetivos inicialmente propostos, pode-se afirmar que o resultado final reflete um modelo de gestão com duas vertentes: autonomia na gestão da otimização do desempenho do servidor na *Cloud*, providenciando um conjunto de mecanismos que procura reduzir as latências caraterísticas de acessos a serviços remotos; e possibilidade de gestão facilitada do sistema de IDS/IPS implementado pelo utilizador, quer ao nível dos serviços em si, mas também na sua interação com a *Firewall*.

Deste modo, além dos mecanismos de auto-adaptação, o utilizador pode igualmente reduzir o impacto que o sistema de IDS/IPS tem no sistema; através de uma avaliação aos serviços que disponibiliza sobre a Internet, este pode decidir quais aqueles que pretende que sejam monitorizados pelo sistema de IDS/IPS. Deste modo, o impacto no processamento e na memória pode ser maior ou menor: quanto maior o número de serviços críticos monitorizados, maior serão o número de padrões de erro em execução, pelo que, tal como se observou na secção 3.2.4, traduz-se num impacto negativo no processamento e memória do servidor.

Por esse motivo, é possível afirmar-se que os objetivos definidos para esta Dissertação foram ultrapassados com sucesso, na medida em que:

- a *Firewall* ficou dotada de mecanismos inteligentes e reativos, através da geração e implementação automática de regras pelo sistema de IDS/IPS;
- após análise das ferramentas de IDS/IPS introduzidas no Capítulo 2, estas foram definidas para diferentes contextos, nomeadamente um sistema com recursos computacionais reduzidos ou um sistema com recursos que proporcionam elevados desempenhos;
- a Interface de Utilizador de gestão, acessível a partir da aplicação *web* de administração, permite uma gestão das funcionalidades de segurança do servidor de uma forma intuitiva e simples, mas o mais completa possível.

Com esta solução, foi introduzida uma metodologia inovadora na otimização do acesso aos recursos de um utilizador em *Cloud*, enquanto se mantém um nível de segurança do perímetro

elevado e caraterizável pelas configurações do utilizador, o que é importante para garantir eficazmente o que os SLA's estabelecidos entre CSP e cliente não satisfazem. Note-se, no entanto, que dado o perfil de utilização para o qual os mecanismos de auto-adaptação das ferramentas de NIDS/NIPS e de níveis de padrões de erro se encontram orientados, esta solução vai de encontro às necessidades específicas do serviço da IPBrick, não sendo automaticamente extensível a outras situações.

6.2 Trabalho Futuro

As possibilidades de trabalho futuro refletem três objetivos principais, nomeadamente **(a)** a carga de trabalho elevada nas configurações do módulo de IDS/IPS na existência múltiplos servidores (físicos e/ou virtuais), **(b)** a maior fiabilidade na seleção da ferramenta de NIDS/NIPS, consoante o perfil de carga do sistema e de caraterísticas do *hardware* não testadas, e **(c)** a implementação de um modo de aquisição de pacotes com maior fiabilidade para velocidade de tráfego elevadas.

6.2.1 Centralização da Gestão do Módulo de IDS/IPS

A emergente utilização de soluções *Cloud* tem implicado que, à semelhança do que existe numa solução tradicional, um cliente consiga contratar um serviço que contemple múltiplos servidores, físicos e/ou virtuais. Por esse motivo, e enquadrado no contexto desta Dissertação, torna-se interessante, na ótica de um administrador, a centralização da gestão do módulo de segurança, apoiada numa arquitetura cliente-servidor.

Ao contrário do OSSEC, que pode ser configurado numa arquitetura cliente-servidor ou apenas local, Snort e Suricata não possuem a possibilidade de configuração semelhante. Por conseguinte, a principal incidência deste trabalho futuro será sobre estas duas ferramentas de IDS e IPS para que possibilitem uma gestão assente numa arquitetura cliente-servidor.

A proposta de solução para este trabalho futuro contempla um esquema de UIs semelhantes às desenvolvidas nesta Dissertação, sendo apenas diferenciada por:

- ao nível do servidor de gestão, este deve conter um modo de selecionar os servidores-cliente que se pretende, por exemplo, efetuar um conjunto de alterações, visualizar dados de monitorização, gerir sugestões de regras a adicionar na *firewall* respetiva, entre outras possibilidades de gestão;

- por outro lado, ao nível dos servidores-cliente, cada servidor deve possuir uma UI que possibilite, essencialmente, a configuração da ligação ao servidor de gestão.

Por esse motivo, o principal desafio deste trabalho futuro acaba por ser o protocolo de comunicação entre servidor de gestão e servidores-cliente, que pode afetar duas caraterísticas principais: o tráfego da rede, devido ao aumento do tráfego de entrada e saída no servidor de gestão e

servidores-cliente, respetivamente; e a latência da comunicação, que implica atrasos na aplicação de medidas de segurança nos servidores-cliente.

O protocolo de comunicação idealizado engloba componentes do protocolo desenvolvido nesta Dissertação, sendo diferenciado pela localização dos componentes, podendo estar situados no servidor de gestão e/ou nos servidores-cliente. Tal leva às implicações referidas ao nível do tráfego de rede e da latência da comunicação da seguinte forma:

- a execução dos *scripts* de tratamento dos *logs* das ferramentas de IDS/IPS apenas no servidor de gestão para que este retorne a ação para o servidor-cliente respetivo implica uma maior latência na aplicação de regras na *firewall*;

- a execução dos *scripts* de tratamento dos *logs* das ferramentas de IDS/IPS apenas nos servidores-cliente, sendo enviado ao servidor de gestão apenas as regras propostas para adição na *firewall* implica uma menor latência em relação à proposta anterior, mas um aumento no tráfego de rede entre as extremidades da comunicação.

6.2.2 Integração de funções de aproximação para seleção do NIDS/NIPS em execução

Por forma a integrar a solução desenvolvida de um maior dinamismo, seria interessante a implementação de funções de aproximação, baseadas na interpolação dos testes realizados às ferramentas de IDS/IPS. Estas funções teriam como objetivo a seleção da ferramenta de IDS e IPS, baseada na Rede, e do nível de regras de cada serviço, consoante nas caraterísticas do sistema.

As funções de aproximação podem-se basear em até três caraterísticas do sistema, consoante o servidor encontre-se em produção ou a ser iniciado, nomeadamente **(a)** a memória RAM, inicial e a um dado momento, **(b)** o número de CPUs e *cores* do processador, **(c)** a frequência do processador e **(d)** o poder de processamento disponível, a um dado momento. Deste modo, quando o servidor é iniciado, são utilizados como parâmetros para as funções de interpolação os valores da memória RAM e a frequência e quantidade de CPUs e *cores* designados para o servidor, enquanto para um servidor em produção, a função de aproximação utilizará os valores da memória atual – livre, *buffer*, cache e utilizada – e poder de processamento disponível.

Note-se que deverá existir uma função de aproximação específica a cada um dos parâmetros referidos, e que indique a ferramenta de IDS e IPS e nível de regras a ser executado, consoante o valor do parâmetro passado.

Devido à existência de múltiplos parâmetros, o que pode gerar resultados diferentes em cada função de aproximação, devem ser utilizados os resultados obtidos numa outra função, na qual cada parâmetro possui um peso específico na seleção da ferramenta de IDS e IPS e nível de regras em execução.

Deste modo, é dotada à solução desenvolvida uma maior adaptabilidade para especificações de RAM e CPU não testados.

6.2.3 Intervalo dinâmico dos *cronjobs* baseado nos *logs* do IDS/IPS

Tal como na secção 6.2.2, uma possível abordagem futura para aumentar o dinamismo da solução desenvolvida passa pela manipulação dos intervalos de execução dos *scripts* executados por *cronjobs*.

Atualmente, os intervalos de tempo dos *cronjobs* inseridos estão definidos de acordo com um perfil de utilização do sistema. No entanto, à semelhança do que acontece com a geração de regras automáticas na *Firewall*, um procedimento similar pode ser utilizado para redefinição do intervalo de tempo dos *cronjobs*, utilizando duas componentes principais: os *logs* das ferramentas de IDS e IPS e o tráfego de rede no servidor.

Assim, a abordagem idealizada contempla dois passos principais, nomeadamente **(a)** geração de dados estatísticos quanto número de tentativas e tipos de acesso ao servidor, obtidos a partir do tratamento dos *logs* das ferramentas de IDS e IPS, e **(b)** recolha dos valores do tráfego de rede, para se perceber a carga de pedidos e respostas que o servidor na *Cloud* está a processar. Deste modo, consegue-se redefinir um novo perfil de utilização do sistema e ajustar os intervalos de tempo para a execução dos *cronjobs*.

6.2.4 Modo de aquisição de pacotes dinâmico num *end-system* com recurso a interfaces de rede virtuais

Similarmente ao modo *nfq*, o *pf_ring* é um modo de aquisição de pacotes passível de ser utilizado pelas ferramentas de IDS/IPS baseadas na Rede, mas que não utiliza a *firewall* para redirecionar os pacotes para o *userspace*.

Como vantagem em relação ao modo *nfq*, o *pf_ring* é um modo orientado para suportar alto desempenho do servidor-alvo, estando idealizado para velocidades de rede superiores a 1 GB, além de não utilizar a *firewall* como auxiliar para redirecionamento do tráfego. No entanto, tendo em conta as suas caraterísticas, é um modo que necessita de *hardware* de alto-nível e da utilização de duas interfaces de rede.

Por esse motivo, a abordagem deste trabalho futuro é semelhante à utilizada na seleção da ferramenta de IDS/IPS em execução, pelo que serão as caraterísticas iniciais do sistema e, posteriormente, o seu desempenho atual a ditar qual o modo de execução ideal para ser executado. Assim, o protocolo inerente a este trabalho futuro é composto por duas etapas essenciais, nomeadamente **(a)** verificação das caraterísticas de *hardware* do servidor e **(b)** decisão de utilização do modo *pf_ring*, baseado nas caraterísticas recolhidas. Caso a decisão seja favorável à possibilidade de utilização do modo *pf_ring*, o protocolo de execução é composto pelas seguintes etapas:

1. criação de uma interface virtual para a interface pública designada para ser monitorizada;
2. definição da interface virtual como interface de entrada e da interface pública como interface de saída no modo *afpacket*;
3. introdução de regra DNAT na *firewall* IPtables para redirecionar o tráfego destinado à interface pública para a interface virtual.

4. periodicamente, efetuar a verificação do desempenho do servidor e, consoante isso, optar pelo modo de aquisição de pacotes ideal – *pf_ring* ou *nfq*.

Anexo A

Resultados da Monitorização ao impacto do Snort e Suricata

Neste Anexo são apresentados as tabelas que contém os diversos resultados obtidos nas monitorizações realizadas ao impacto das ferramentas de NIDS/NIPS, Snort e Suricata, no sistema em produção.

Ambiente de teste	**Tipo de Memória**				**Processamento**		
	Sistema Geral				Sistema Geral		
	Livre	Usada	*Buffer*	*Cache*	Util.	Sist.	Livre
A e J	75 MB	621 MB	7 MB	258 MB	0.2%	0.2%	99.7%
B e K	79 MB	613 MB	34 MB	264 MB	0.1%	0.3%	99.6%
C e L	78 MB	621 MB	31 MB	259 MB	0.0%	0.1%	99.9%
D e M	1082 MB	618 MB	34 MB	265 MB	0.2%	0.4%	99.4%
E e N	1076 MB	618 MB	37 MB	269 MB	0.0%	0.1%	99.8%
F e O	1069 MB	628 MB	36 MB	267 MB	0.0%	0.0%	100.0%
G e P	3027 MB	627 MB	35 MB	266 MB	0.2%	0.4%	99.3%
H e Q	3026 MB	629 MB	33 MB	267 MB	0.0%	0.2%	99.6%
I e R	3016 MB	639 MB	33 MB	267 MB	0.0%	0.0%	99.9%

Tabela A.1: Desempenho médio do sistema, para utilizar como valores de referência.

Ambiente de teste	**Tipo de Memória**					**Processamento**			
	Sistema Geral				IDS	Sistema Geral			IDS
	Livre	Usada	*Buffer*	*Cache*		Util.	Sist.	Livre	
A	70 MB	726 MB	18 MB	175 MB	11.0%	0.1%	0.2%	99.2%	0.0%
B	70 MB	728 MB	18 MB	173 MB	11.0%	0.0%	0.1%	99.4%	0.0%
C	71 MB	735 MB	17 MB	165 MB	11.0%	0.0%	0.0%	99.5%	0.0%
D	952 MB	734 MB	36 MB	277 MB	5.5%	0.2%	0.0%	99.7%	0.0%
E	957 MB	732 MB	36 MB	274 MB	5.4%	0.0%	0.1%	99.8%	0.0%
F	935 MB	749 MB	40 MB	274 MB	5.4%	0.0%	0.0%	99.9%	0.0%
G	2898 MB	743 MB	42 MB	272 MB	2.8%	0.2%	0.1%	99.7%	0.0%
H	2895 MB	745 MB	41 MB	274 MB	2.8%	0.0%	0.1%	99.7%	0.0%
I	2888 MB	753 MB	40 MB	274 MB	2.8%	0.0%	0.0%	99.9%	0.0%
J	71 MB	843 MB	4 MB	71 MB	30.6%	0.2%	0.3%	98.0%	0.0%
K	70 MB	846 MB	4 MB	70 MB	30.6%	0.1%	0.2%	98.9%	0.0%
L	80 MB	818 MB	5 MB	86 MB	30.6%	0.0%	0.0%	99.8%	0.0%
M	660 MB	951 MB	43 MB	346 MB	15.1%	0.1%	0.2%	99.7%	0.0%
N	746 MB	937 MB	37 MB	278 MB	15.1%	0.0%	0.1%	99.9%	0.0%
O	740 MB	943 MB	42 MB	277 MB	15.1%	0.0%	0.1%	99.9%	0.0%
P	2653 MB	949 MB	43 MB	310 MB	7.7%	0.2%	0.3%	99.5%	0.0%
Q	2684 MB	950 MB	43 MB	278 MB	7.7%	0.0%	0.0%	99.8%	0.0%
R	2677 MB	958 MB	41 MB	278 MB	7.7%	0.0%	0.0%	100.0%	0.0%

Tabela A.2: Desempenho médio do impacto do Snort, executado como IDS, na memória e no CPU.

Ambiente de teste	**Tipo de Memória**					**Processamento**			
	Sistema Geral				IDS	Sistema Geral			IDS
	Livre	Usada	*Buffer*	*Cache*		Util.	Sist.	Livre	
A	80 MB	687 MB	14 MB	208 MB	8.1%	0.4%	0.3%	99.1%	0.5%
B	101 MB	698 MB	32 MB	158 MB	8.1%	0.2%	0.2%	99.1%	1.0%
C	72 MB	702 MB	27 MB	188 MB	8.1%	0.0%	0.0%	99.7%	1.0%
D	980 MB	698 MB	49 MB	271 MB	4.0%	0.2%	0.4%	99.4%	0.5%
E	985 MB	699 MB	43 MB	272 MB	4.0%	0.1%	0.1%	99.7%	0.9%
F	975 MB	709 MB	43 MB	272 MB	4.0%	0.0%	0.0%	99.9%	0.7%
G	2936 MB	708 MB	40 MB	270 MB	2.0%	0.2%	0.3%	99.6%	0.5%
H	2934 MB	710 MB	39 MB	272 MB	2.0%	0.1%	0.1%	99.7%	0.7%
I	2923 MB	719 MB	40 MB	273 MB	2.0%	0.0%	0.1%	99.8%	0.6%
J	120 MB	749 MB	5 MB	115 MB	16.6%	0.2%	0.2%	99.5%	0.6%
K	80 MB	772 MB	14 MB	123 MB	16.6%	0.1%	0.0%	99.3%	0.7%
L	76 MB	776 MB	17 MB	118 MB	16.6%	0.0%	0.0%	99.9%	1.0%
M	882 MB	789 MB	51 MB	277 MB	8.2%	0.2%	0.2%	99.6%	0.6%
N	887 MB	789 MB	45 MB	277 MB	8.2%	0.1%	0.1%	99.7%	0.9%
O	862 MB	808 MB	49 MB	280 MB	8.2%	0.0%	0.0%	100.0%	0.5%
P	2840 MB	799 MB	42 MB	273 MB	4.2%	0.2%	0.1%	99.7%	0.5%
Q	2839 MB	800 MB	41 MB	275 MB	4.2%	0.1%	0.1%	99.8%	0.6%
R	2826 MB	810 MB	41 MB	277 MB	4.2%	0.0%	0.0%	99.9%	0.6%

Tabela A.3: Desempenho médio do impacto do Suricata, executado como IDS, na memória e no CPU.

Ambiente de teste	**Tipo de Memória**					**Processamento**			
	Sistema Geral				IPS	Sistema Geral			IPS
	Livre	Usada	*Buffer*	*Cache*		Util.	Sist.	Livre	
A	343 MB	567 MB	4 MB	75 MB	11.0%	0.2%	0.1%	99.7%	0.0%
B	128 MB	726 MB	15 MB	120 MB	11.0%	0.0%	0.1%	99.8%	0.0%
C	72 MB	732 MB	18 MB	166 MB	11.0%	0.0%	0.0%	99.4%	0.0%
D	957 MB	730 MB	39 MB	272 MB	5.3%	0.1%	0.2%	99.6%	0.0%
E	931 MB	750 MB	39 MB	279 MB	5.3%	0.0%	0.0%	99.9%	0.0%
F	900 MB	778 MB	43 MB	278 MB	5.3%	0.0%	0.0%	100.0%	0.0%
G	2801 MB	763 MB	45 MB	345 MB	2.7%	0.1%	0.1%	99.7%	0.0%
H	2850 MB	766 MB	44 MB	294 MB	2.7%	0.0%	0.0%	99.8%	0.0%
I	2862 MB	772 MB	43 MB	279 MB	2.7%	0.0%	0.0%	99.9%	0.0%
J	75 MB	804 MB	8 MB	102 MB	30.3%	0.2%	0.2%	99.6%	0.0%
K	72 MB	840 MB	3 MB	74 MB	30.3%	0.0%	0.1%	99.6%	0.0%
L	70 MB	844 MB	3 MB	71 MB	30.3%	0.0%	0.0%	99.6%	0.0%
M	704 MB	938 MB	41 MB	317 MB	15.0%	0.2%	0.3%	99.5%	0.0%
N	660 MB	974 MB	42 MB	322 MB	15.0%	0.0%	0.1%	99.7%	0.0%
O	691 MB	981 MB	44 MB	282 MB	15.0%	0.0%	0.0%	100.0%	0.0%
P	2595 MB	967 MB	46 MB	347 MB	7.6%	0.1%	0.1%	99.8%	0.0%
Q	2567 MB	980 MB	41 MB	340 MB	7.6%	0.1%	0.0%	99.9%	0.0%
R	2372 MB	894 MB	79 MB	557 MB	7.6%	0.2%	0.4%	99.4%	0.0%

Tabela A.4: Desempenho médio do impacto do Snort, executado como IPS, na memória e no CPU.

Ambiente de teste	**Tipo de Memória**					**Processamento**			
	Sistema Geral				IPS	Sistema Geral			IPS
	Livre	Usada	*Buffer*	*Cache*		Livre	Sist.	Utiliz.	
A	191 MB	673 MB	8 MB	117 MB	7.9%	0.2%	0.1%	99.5%	0.6%
B	275 MB	613 MB	9 MB	92 MB	7.9%	0.1%	0.1%	99.7%	0.7%
C	154 MB	695 MB	19 MB	121 MB	7.9%	0.0%	0.0%	99.9%	1.0%
D	962 MB	708 MB	52 MB	277 MB	3.9%	0.2%	0.1%	99.7%	0.6%
E	967 MB	708 MB	46 MB	278 MB	3.9%	0.2%	0.0%	99.7%	1.3%
F	941 MB	726 MB	51 MB	280 MB	3.9%	0.0%	0.0%	100.0%	0.7%
G	2919 MB	718 MB	43 MB	274 MB	2.0%	0.2%	0.1%	99.7%	0.6%
H	2919 MB	718 MB	41 MB	276 MB	2.0%	0.0%	0.0%	99.8%	0.6%
I	2905 MB	729 MB	42 MB	277 MB	2.0%	0.0%	0.0%	100.0%	0.6%
J	88 MB	759 MB	8 MB	134 MB	16.3%	0.3%	0.3%	99.1%	0.6%
K	175 MB	705 MB	14 MB	95 MB	16.3%	0.1%	0.0%	99.8%	0.8%
L	75 MB	781 MB	16 MB	117 MB	16.3%	0.0%	0.0%	99.8%	1.0%
M	872 MB	796 MB	53 MB	278 MB	8.1%	0.2%	0.1%	99.7%	0.5%
N	640 MB	520 MB	169 MB	375 MB	8.1%	0.5%	0.5%	98.5%	1.3%
O	851 MB	815 MB	52 MB	281 MB	8.1%	0.0%	0.0%	100.0%	0.7%
P	2828 MB	806 MB	44 MB	277 MB	4.1%	0.2%	0.2%	99.7%	0.5%
Q	2827 MB	807 MB	43 MB	277 MB	4.1%	0.1%	0.0%	99.8%	0.6%
R	2815 MB	818 MB	43 MB	278 MB	4.1%	0.0%	0.0%	100.0%	0.6%

Tabela A.5: Desempenho médio do impacto do Suricata, executado como IPS, na memória e no CPU.

Anexo B

Configuração do Sistema

Neste Anexo são apresentadas as configurações realizadas ao nível do sistema IPBRICK OS, no que respeita às ferramentas de IDS/IPS, como também ao nível de novos diretórios, dedicados à manipulação por níveis de regras.

B.1 Snort

B.1.1 Modificações do ficheiro de configuração

A configuração do Snort incidiu sobre o ficheiro */etc/snort/snort_eth1.conf*, responsável pelas configurações das suas funcionalidades. Seguidamente, são expostas as várias modificações efetuadas, comparativamente ao ficheiro de configuração que, por defeito, vem configurado (note-se que o ficheiro exposto está com uma configuração orientada para a interface de rede pública *eth1*).

```
ipvar HOME_NET <endereco_IP_eth1_servidor>
ipvar EXTERNAL_NET !$HOME_NET

var RULE_CONFIG_PATH /etc/snort/rules/config_files/eth1/
var RULE_CONFIG_DEFAULT /etc/snort/rules/defaults/
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

output alert_unified2: filename snort.alert, limit 128, nostamp

config logdir: /var/log/snort/eth1/

# Apenas ativo no modo IPS
# config daq: nfq
# config daq_dir: /usr/local/lib/daq
# config daq_var: eth1
```

```
include $RULE_CONFIG_DEFAULT/rules_default.conf
include $RULE_CONFIG_PATH/rules_level1.conf
include $RULE_CONFIG_PATH/rules_level2.conf
```

B.1.2 Configuração dos níveis de padrões de erro

Por forma a permitir a manipulação de regras, baseada por níveis, mas também por número de interfaces públicas, foi realizada as seguintes configurações através da linha de comandos. Note-se que apenas está representada para uma interface de rede pública que, por defeito, existe em qualquer servidor IPBrick. A existência de múltiplas interfaces de rede públicas deverá seguir a criação automática de novos diretórios e ficheiros de configuração correspondentes aos últimos três comandos representados seguidamente.

```
# Novos Diretorios
$ mkdir /etc/snort/rules/levels/level1/ /etc/snort/rules/levels/level2/
$ mkdir /etc/snort/rules/defaults/
$ mkdir /etc/snort/rules/config_files/eth1/

# Novos ficheiros
$ touch /etc/snort/rules/config_files/eth1/rules_level1.conf
$ touch /etc/snort/rules/config_files/eth1/rules_level2.conf
$ touch /etc/snort/rules/defaults/rules_default.conf
```

B.2 Suricata

B.2.1 Modificações do ficheiro de configuração

A configuração do Suricata incidiu sobre o ficheiro */etc/suricata/suricata_eth1.yaml*, responsável pelas configurações das suas funcionalidades. Seguidamente, são expostas as várias modificações efetuadas, relativamente ao ficheiro de configuração que, por defeito, vem configurado (note-se que o ficheiro exposto está com uma configuração orientada para a interface de rede pública *eth1*). No entanto, ao contrário do Snort, não foi possível adicionar uma modularização de ficheiros de configuração para os padrões de erro, uma vez que o Suricata não suporta.

```
vars:
 address-groups:
  HOME_NET: "[<endereco_IP_eth1_servidor>]"
  EXTERNAL_NET: "!$HOME_NET"
 outputs:
  - unified2-alert:
   enabled: yes
   filename: unified2.alert
```

```
default-log-dir: /var/log/suricata/eth1/

default-rule-path: /etc/suricata/rules/
include: /etc/suricata/rules/config_files/eth1/rules_levels.yaml
```

B.2.2 Configuração dos níveis de padrões de erro

Tal como o Snort, o Suricata necessitou de um conjunto de configurações idênticas para permitir a manipulação por níveis de regras.

```
# Novos Diretorios
$ mkdir /etc/suricata/rules/levels/level1/ /etc/snort/rules/levels/level2/
$ mkdir /etc/suricata/rules/defaults/
$ mkdir /etc/suricata/rules/config_files/eth1/

# Novos ficheiros
$ touch /etc/suricata/rules/config_files/eth1/rules_levels.yaml
```

B.3 OSSEC

B.3.1 Adaptação do OSSEC ao IPBRICK OS

Por forma a que o OSSEC armazenasse os *logs* na base de dados, foi necessário alterar o código-fonte do mesmo, uma vez que este acede ao porto errado de acesso à base de dados. Tal deve-se à existência de duas interfaces de gestão das bases de dados no sistema IPBRICK OS, acessíveis nos portos 5432 e 5433, para os clientes e para administração pela IPBrick, respetivamente. No entanto, por defeito, apenas o porto 5432 é visível ao OSSEC, pelo que foi necessário introduzir, de forma *hardcoded*, o porto 5433 no código-fonte do OSSEC antes de efetuar a instalação do mesmo, bem como corrigir um *bug* existente quanto ao tipo de variável passada por parâmetro a uma função já existente. Seguidamente, estão expostas as alterações realizadas no ficheiro *ossec-hids-2.8.3/src/os_dbd/db_op.c*.

```
void *postgresql_osdb_connect(char *host, char *user, char *pass, char *db
    , int port, char *sock)
{
  (...)
  char portstr[5];
  sprintf(portstr, "%d", port);

  conn = PQsetdbLogin(host, portstr, NULL, NULL, db, user, pass);
  (...)
}

void osdb_checkerror()
{
```

```
13   (...)
14   db_config_pt->conn = osdb_connect(db_config_pt->host ,db_config_pt->user
       , db_config_pt->pass, db_config_pt->db, 5433, db_config_pt->sock);
15   (...)
16 }
```

B.4 Barnyard2

Por forma a possibilitar tanto o Snort, como o Suricata a efetuar *logging* para a base de dados com recurso ao Barnyard2, foram criados dois ficheiros de configuração distintos, barnyard2_snort.conf e barnyard2_suricata.conf, sendo cada um deles adicionado aos diretórios das ferramentas de NIDS/NIPS em causa.

Seguidamente, estão expostas as configurações realizadas do Barnyard2 para cada ferramenta de NIDS/NIPS.

B.4.1 Snort

Por forma a que o Snort efetuasse o *logging* na base de dados, o ficheiro *barnyard2.conf* foi copiado para */etc/snort/* e foram efetuadas as seguintes modificações ao mesmo:

```
config reference_file:      /etc/snort/reference.config
config classification_file: /etc/snort/classification.config
config gen_file:            /etc/snort/gen-msg.map
config sid_file:            /etc/snort/sid-msg.map

config daemon

output database: log, postgresql, user=sistema_ids password=sistema_ids
 dbname=snort host=localhost port=5433
```

B.4.2 Suricata

Por forma a que o Suricata efetuasse o *logging* na base de dados, o ficheiro *barnyard2.conf* foi copiado para */etc/suricata/* e foram efetuadas as seguintes modificações ao mesmo:

```
config reference_file:      /etc/suricata/reference.config
config classification_file: /etc/suricata/classification.config
config gen_file:            /etc/suricata/gen-msg.map
config sid_file:            /etc/suricata/sid-msg.map

config daemon
```

```
output database: log, postgresql, user=sistema_ids password=sistema_ids
dbname=suricata host=localhost port=5433
```

Anexo C

Código-fonte

Neste Anexo são apresentados excertos do código-fonte desenvolvido no contexto da solução idealizada, ao nível da aplicação *web* de administração, mas também ao nível do sistema.

C.1 Geração de Regras de *Firewall* pelo Sistema de IDS/IPS

O seguinte excerto de código é responsável por adicionar as regras de *Firewall* automaticamente, em tempo-real, ou adicioná-las à tabela *Regra_IDS* (ver modelo relacional da figura 3.10). Trata-se de um exemplo demonstrativo do funcionamento, utilizando o protocolo ICMP.

```
<?php
  (...)
  for ($j=0, $l=0; $j<count($ip_diferentes); $j++){
    $dados_servico = novaRegraAutomaticaManual($ip_diferentes[$j],
    $logs_servico["ICMP"],"ICMP");
    if ($dados_servico["ICMP"]["prioridade"] == 1){
      $command = "iptables -I INPUT -i eth".$ids_instance[$i]->interface."
     -s '$ip_diferentes[$j]' -p ICMP --icmp-type 8 -j DROP";
      execAsRoot($command,$insere_output);
      adicionaRegra ("", $ip_diferentes[$j], $ids_instance[$i]->interface,
     "", 1, 3, "Automatically added by IDS/IPS system.", "IDS/IPS");
      } else {
        $dbidsips->insereSugestaoRegraIptables($data, "ICMP",
    $dados_servico["ICMP"]["prioridade"], $ip_diferentes[$j],
    $dados_servico["ICMP"]["tentativas"], $ids_instance[$i]->id_ids);
      }
  }
  (...)
?>
```

C.2 Modificações de configurações no Sistema

O seguinte *script* Shell, denominado *setup_ids.sh*, é responsável pelas várias configurações no sistema IPBRICK OS, quer ao nível da *Firewall*, como das ferramentas de IDS/IPS utilizadas. Este está modularizado em várias funções, caraterizadas por chamadas específicas do *script*, para as várias funcionalidades implementadas, quer seja uma interação direta ou indireta do utilizador, mas também para a execução dos mecanismos de auto-adaptação implementados.

```
1  #! /bin/sh
2  # $1 modo_execucao_script: selecao servicos ou selecao modo
3
4  # $1: flag_desativa_regras
5  # $2: servico_ativar
6  # $3: interface
7  change_services(){
8  ERRO=0
9
10 # Comenta servicos existentes
11 if [ "$1" -eq 1 ]; then
12 # Snort
13 sed -i "/^#/!s/include \$RULE\_LEVEL1\_PATH/# include \$RULE\_LEVEL1\_PATH/" /
       etc/snort/rules/config_files/"$3"/rules_level1.conf
14 sed -i "/^#/!s/include \$RULE\_LEVEL2\_PATH/# include \$RULE\_LEVEL2\_PATH/" /
       etc/snort/rules/config_files/"$3"/rules_level2.conf
15 # Suricata
16 sed -i "/levels\/level*/s/^#*/#/" /etc/suricata/rules/config_files/"$3"/
       rules_levels.yaml
17 fi
18
19 case "$2" in
20 imap)
21 # Snort
22 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/imap\.rules/include \$RULE\_LEVEL1\
       _PATH\/imap\.rules/" /etc/snort/rules/config_files/"$3"/rules_level1.conf
23 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/imap\.rules/include \$RULE\_LEVEL2\
       _PATH\/imap\.rules/" /etc/snort/rules/config_files/"$3"/rules_level2.conf
24 #Suricata
25 sed -i "s/# - levels\/level1\/imap\.rules/ - levels\/level1\/imap\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml
26 sed -i "s/# - levels\/level2\/imap\.rules/ - levels\/level2\/imap\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml ;;
27 dns)
28 # Snort
29 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/dns\.rules/include \$RULE\_LEVEL1\
       _PATH\/dns\.rules/" /etc/snort/rules/config_files/"$3"/rules_level1.conf
30 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/dns\.rules/include \$RULE\_LEVEL2\
       _PATH\/dns\.rules/" /etc/snort/rules/config_files/"$3"/rules_level2.conf
31 #Suricata
```

```
32 sed -i "s/# - levels\/level1\/dns\.rules/ - levels\/level1\/dns\.rules/" /etc/
       suricata/rules/config_files/"$3"/rules_levels.yaml
33 sed -i "s/# - levels\/level2\/dns\.rules/ - levels\/level2\/dns\.rules/" /etc/
       suricata/rules/config_files/"$3"/rules_levels.yaml ;;
34 ftp)
35 # Snort
36 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/ftp\.rules/include \$RULE\_LEVEL1\
       _PATH\/ftp\.rules/" /etc/snort/rules/config_files/"$3"/rules_level1.conf
37 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/ftp\.rules/include \$RULE\_LEVEL2\
       _PATH\/ftp\.rules/" /etc/snort/rules/config_files/"$3"/rules_level2.conf
38 #Suricata
39 sed -i "s/# - levels\/level1\/ftp\.rules/ - levels\/level1\/ftp\.rules/" /etc/
       suricata/rules/config_files/"$3"/rules_levels.yaml
40 sed -i "s/# - levels\/level2\/ftp\.rules/ - levels\/level2\/ftp\.rules/" /etc/
       suricata/rules/config_files/"$3"/rules_levels.yaml ;;
41 pop3)
42 # Snort
43 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/pop3\.rules/include \$RULE\_LEVEL1\
       _PATH\/pop3\.rules/" /etc/snort/rules/config_files/"$3"/rules_level1.conf
44 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/pop3\.rules/include \$RULE\_LEVEL2\
       _PATH\/pop3\.rules/" /etc/snort/rules/config_files/"$3"/rules_level2.conf
45 #Suricata
46 sed -i "s/# - levels\/level1\/pop3\.rules/ - levels\/level1\/pop3\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml
47 sed -i "s/# - levels\/level2\/pop3\.rules/ - levels\/level2\/pop3\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml ;;
48 voip)
49 # Snort
50 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/voip\.rules/include \$RULE\_LEVEL1\
       _PATH\/voip\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level1.conf
51 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/voip\.rules/include \$RULE\_LEVEL2\
       _PATH\/voip\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level2.conf
52 #Suricata
53 sed -i "s/# - levels\/level1\/voip\.rules/ - levels\/level1\/voip\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml
54 sed -i "s/# - levels\/level2\/voip\.rules/ - levels\/level2\/voip\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml ;;
55 smtp)
56 # Snort
57 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/smtp\.rules/include \$RULE\_LEVEL1\
       _PATH\/smtp\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level1.conf
58 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/smtp\.rules/include \$RULE\_LEVEL2\
       _PATH\/smtp\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level2.conf
59 #Suricata
60 sed -i "s/# - levels\/level1\/smtp\.rules/ - levels\/level1\/smtp\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml
61 sed -i "s/# - levels\/level2\/smtp\.rules/ - levels\/level2\/smtp\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml ;;
62 http)
```

```
63 # Snort
64 sed -i "s/# include \$RULE\_LEVEL1\_PATH\/http\.rules/include \$RULE\_LEVEL1\
       _PATH\/http\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level1.conf
65 sed -i "s/# include \$RULE\_LEVEL2\_PATH\/http\.rules/include \$RULE\_LEVEL2\
       _PATH\/http\.rules/"  /etc/snort/rules/config_files/"$3"/rules_level2.conf
66 #Suricata
67 sed -i "s/# - levels\/level1\/http\.rules/ - levels\/level1\/http\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml
68 sed -i "s/# - levels\/level2\/http\.rules/ - levels\/level2\/http\.rules/" /etc
       /suricata/rules/config_files/"$3"/rules_levels.yaml ;;
69 all)
70 # Snort
71 sed -i "s/# include \$RULE\_LEVEL1\_PATH/include \$RULE\_LEVEL1\_PATH/" /etc/
       snort/rules/config_files/"$3"/rules_level1.conf
72 sed -i "s/# include \$RULE\_LEVEL2\_PATH/include \$RULE\_LEVEL2\_PATH/" /etc/
       snort/rules/config_files/"$3"/rules_level2.conf
73 #Suricata
74 sed -i "/\levels\/level*/s/^#//g" /etc/suricata/rules/config_files/"$3"/
       rules_levels.yaml ;;
75 *)
76 ERRO=1 ;;
77 esac
78
79 if [ "$ERRO" == 1 ]; then
80 echo "ERROR"
81 else
82 echo "SUCCESS"
83 fi
84
85 }
86
87 # $1: modo_atual
88 # $2: modo_novo
89 # $3: interface_rede
90 # $4: ferramenta
91 # $5: pid_ferramenta
92 # $6: ip_interface
93 change_tool(){
94
95 #   sed -i "s/^ipvar HOME_NET.*$/ipvar HOME_NET "$ENDERECO_IP"\/32/" /etc/snort
       /snort.conf
96
97 if [ "$1" -ne 2 ] && [ "$2" -eq 2 ]; then
98 if [ "$4" = "Snort" ]; then
99 sed -i "s/# config daq_dir: \/usr\/local\/lib\/daq/config daq_dir: \/usr\/local
       \/lib\/daq/" /etc/snort/snort_"$3".conf
100 sed -i "s/# config daq: nfq/config daq: nfq/" /etc/snort/snort_"$3".conf
101 sed -i "s/# config daq_var: device = "$3"/config daq_var: device = "$3"/" /etc/
       snort/snort_"$3".conf
```

```
fi
elif [ "$1" -eq 2 ] && [ "$2" -ne 2 ]; then
if [ "$4" = "Snort" ]; then
sed -i "/^#/!s/config daq: nfq/# config daq: nfq/" /etc/snort/snort_"$3".conf
sed -i "/^#/!s/config daq_dir: \/usr\/local\/lib\/daq/# config daq_dir: \/usr\/local\/lib\/daq/" /etc/snort/snort_"$3".conf
sed -i "/^#/!s/config daq_var: device = "$3"/# config daq_var: device = "$3"/" /etc/snort/snort_"$3".conf
fi
fi

if [ "$1" -eq 1 ] && [ "$2" -eq 1 ]; then
if [ -z $5 ] || [ "$5" -ne 0 ]; then
kill "$5"
fi

if [ "$4" = "Snort" ]; then
snort -i "$3" -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/ -D
else
suricata -i "$3" -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips/suricata/suricata.pid -D
fi
elif [ "$1" -eq 1 ] && [ "$2" -eq 2 ]; then
if [ -z $5 ] || [ "$5" -ne 0 ]; then
kill "$5"
fi

if [ "$4" = "Snort" ]; then
snort -Q -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/ -D
else
suricata -q 0 -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips/suricata/suricata.pid -D
fi
iptables -t filter -A INPUT -j NFQUEUE --queue-num 0
iptables -t filter -A OUTPUT -j NFQUEUE --queue-num 0
elif [ "$1" -eq 2 ] && [ "$2" -eq 1 ]; then
iptables -t filter -D INPUT -j NFQUEUE --queue-num 0
iptables -t filter -D OUTPUT -j NFQUEUE --queue-num 0
if [ -z "$5" ] || [ "$5" -ne 0 ]; then
kill "$5"
fi

if [ "$4" = "Snort" ]; then
snort -i "$3" -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/ -D
else
suricata -i "$3" -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips/suricata/suricata.pid -D
```

```
144 fi
145 elif [ "$1" -eq 2 ] && [ "$2" -eq 2 ]; then
146 iptables -t filter -D INPUT -j NFQUEUE --queue-num 0
147 iptables -t filter -D OUTPUT -j NFQUEUE --queue-num 0
148 if [ -z "$5" ] || [ "$5" -ne 0 ]; then
149 kill "$5"
150 fi
151
152 if [ "$4" = "Snort" ]; then
153 snort -Q -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/ -D
154 else
155 suricata -q 0 -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips/
    suricata/suricata.pid -D
156 fi
157 iptables -t filter -A INPUT -j NFQUEUE --queue-num 0
158 iptables -t filter -A OUTPUT -j NFQUEUE --queue-num 0
159 elif [ "$1" -eq 0 ] && [ "$2" -eq 1 ]; then
160 if [ "$4" = "Snort" ]; then
161 snort -i "$3" -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/
    -D
162 else
163 suricata -i "$3" -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips
    /suricata/suricata.pid -D
164 fi
165 elif [ "$1" -eq 0 ] && [ "$2" -eq 2 ]; then
166 if [ "$4" = "Snort" ]; then
167 snort -i "$3" -c /etc/snort/snort_"$3".conf --pid-path /var/run/ids_ips/snort/
    -D
168 else
169 suricata -i "$3" -c /etc/suricata/suricata_"$3".yaml --pidfile /var/run/ids_ips
    /suricata/suricata.pid -D
170 fi
171 iptables -t filter -A INPUT -j NFQUEUE --queue-num 0
172 iptables -t filter -A OUTPUT -j NFQUEUE --queue-num 0
173 elif [ "$1" -eq 1 ] && [ "$2" -eq 0 ]; then
174 if [ -z "$5" ] || [ "$5" -ne 0 ]; then
175 kill "$5"
176 fi
177
178 elif [ "$1" -eq 0 ] && [ "$2" -eq 2 ]; then
179 iptables -t filter -D INPUT -j NFQUEUE --queue-num 0
180 iptables -t filter -D OUTPUT -j NFQUEUE --queue-num 0
181 if [ -z "$5" ] || [ "$5" -ne 0 ]; then
182 kill "$5"
183 fi
184
185 fi
186
187 sleep 2s
```

```
188
189 if [ "$4" = "Snort" ] && [ "$2" -ne 0 ] ; then
190 IDS=1
191 PID=$( cat /var/run/ids_ips/snort/snort_"$3".pid )
192 elif [ "$4" = "Suricata" ] && [ "$2" -ne 0 ]; then
193 IDS=2
194 PID=$( cat /var/run/ids_ips/suricata/suricata.pid )
195 elif [ "$2" -eq 0 ]; then
196 IDS=0
197 PID=0
198 fi
199
200 php5 /opt/system/scripts/ids_ips/atualiza_sistema_ids.php ct "$3" $IDS $PID
201
202 }
203
204 # $1: producao/boot
205 # $2: ferramenta
206 # $3: modo
207 # $4: pid
208 # $5: interface
209 select_tool(){
210
211 if [ "$1" == 2 ] || [ "$4" != 0 ]; then
212 kill "$4"
213 fi
214
215 if [ "$2" = "Snort" ]; then
216 if [ "$3" -eq 1 ]; then
217 snort -i "$5" -c /etc/snort/snort_"$5".conf --pid-path /var/run/ids_ips/snort/
    -D
218 else
219 snort -Q -c /etc/snort/snort_"$5".conf --pid-path /var/run/ids_ips/snort/ -D
220 iptables -t filter -A INPUT -j NFQUEUE --queue-num 0
221 iptables -t filter -A OUTPUT -j NFQUEUE --queue-num 0
222 fi
223 else
224 if [ "$3" -eq 1]; then
225 suricata -i "$5" -c /etc/suricata/suricata_"$5".yaml --pidfile /var/run/ids_ips
    /suricata/suricata.pid -D
226 else
227 suricata -q 0 -c /etc/suricata/suricata_"$5".yaml --pidfile /var/run/ids_ips/
    suricata/suricata.pid -D
228 iptables -t filter -A INPUT -j NFQUEUE --queue-num 0
229 iptables -t filter -A OUTPUT -j NFQUEUE --queue-num 0
230 fi
231 fi
232
233 if [ "$1" -eq 1 ]; then
```

```
234 barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort/"$5" -f snort.alert -w
        /var/log/snort/"$5"/snort.waldo -D
235 barnyard2 -c /etc/suricata/barnyard2.conf -d /var/log/suricata/"$5" -f unified2
        .alert -w /var/log/suricata/"$5"/suricata.waldo -D
236 /var/ossec/bin/ossec-control start
237 fi
238
239
240 }
241
242 # $1: level
243 # $2: interface
244 select_level(){
245
246 # Snort
247 sed -i "/^#/!s/include \$RULE\_CONFIG\_PATH/# include \$RULE\_CONFIG\_PATH/" /
        etc/snort/snort_"$2".conf
248 # Suricata
249 sed -i "/^#/!s/include: \/etc\/suricata\/rules\/config\_files/# include: \/etc
        \/suricata\/rules\/config\_files/" /etc/suricata/suricata_"$2".yaml
250
251 if [ "$1" -eq 1 ]; then
252 # Snort
253 sed -i "s/# include \$RULE\_CONFIG\_PATH\/rules\_level1\.conf/include \$RULE\
        _CONFIG\_PATH\/rules\_level1\.conf/" /etc/snort/snort_"$2".conf
254 # Suricata
255 sed -i "s/# - levels\/level1/ - levels\/level1/" /etc/suricata/rules/
        config_files/"$2"/rules_levels.yaml
256 else
257 # Snort
258 sed -i "s/# include \$RULE\_CONFIG\_PATH\/rules\_level1\.conf/include \$RULE\
        _CONFIG\_PATH\/rules\_level1\.conf/" /etc/snort/snort_"$2".conf
259 sed -i "s/# include \$RULE\_CONFIG\_PATH\/rules\_level2\.conf/include \$RULE\
        _CONFIG\_PATH\/rules\_level2\.conf/" /etc/snort/snort_"$2".conf
260 # Suricata
261 sed -i "s/# - levels\/level1/ - levels\/level1/" /etc/suricata/rules/
        config_files/"$2"/rules_levels.yaml
262 sed -i "s/# - levels\/level2/ - levels\/level2/" /etc/suricata/rules/
        config_files/"$2"/rules_levels.yaml
263 fi
264 }
265
266 [ -z $1 ] && echo "ERROR" && exit 1;
267
268 # Loop for different options
269 while [[ $1 == -* ]]; do
270 case "$1" in
271 -cs|--ch_services) change_services $2 $3 $4; exit 0;;
272 -ct|--ch_tool) change_tool $2 $3 $4 $5 $6 $7; exit 0;;
```

```
273 -cb|--ch_boot_prod) select_tool $2 $3 $4 $5 $6; exit 0;;
274 -cl|--ch_level) select_level $2 $3; exit 0;;
275 --) shift; break;;
276 -*) echo "Invalid option: $1"; exit 1;;
277 esac
278 done
```

Referências

[1] IPBRICK. IPBRICK - Corporate Communication Solutions. Disponível em http://www.ipbrick.com/pt-pt/, acedido a última vez em 5 de Fevereiro de 2016.

[2] IPBRICK. IPBRICK OS. Disponível em http://www.ipbrick.com/pt-pt/ipbrick-ic-2/, acedido a última vez em 5 de Fevereiro de 2016.

[3] Peter Mell e Timothy Grance. The NIST Definition of Cloud Computing, Setembro 2011. Disponível em http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, acedido a última vez em 8 de Fevereiro de 2016.

[4] Richard Thelwell. An Interactive History of Cloud Computing, Setembro 2015. Disponível em http://www.business2community.com/infographics/interactive-history-cloud-computing-01304429#1Rpl5IWGLcmbzhjB.97, acedido a última vez em 8 de Fevereiro de 2016.

[5] Shamus McGillicuddy. Layer 4-7 cloud networking still scarce in IaaS market, Agosto 2012. Disponível em http://searchnetworking.techtarget.com/news/2240161069/Layer-4-7-cloud-networking-still-scarce-in-IaaS-market, acedido a última vez em 9 de Fevereiro de 2016.

[6] Carlos Gonçalves. Cloud service broker, 2014. Disponível em http://hdl.handle.net/10773/12746, acedido a última vez em 9 de Fevereiro de 2016.

[7] Jonathan Appavoo e Waterland *et. al*. Providing a cloud network infrastructure on a supercomputer. Em *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, páginas 385–394. ACM, 2010. URL: http://doi.acm.org/10.1145/1851476.1851534.

[8] dimension data. Network Infrastructure. Disponível em http://cloud.dimensiondata.com/saas-solutions/services/managed-hosting/Network-Infrastructure, acedido a última vez em 9 de Fevereiro de 2016.

[9] Jonathan Appavoo e Waterland *et. al*. Providing a cloud network infrastructure on a supercomputer. Em *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, páginas 385–394. ACM, 2010. URL: http://doi.acm.org/10.1145/1851476.1851534.

[10] dimension data. Data Center Infrastructure. Disponível em http://cloud.dimensiondata.com/saas-solutions/services/managed-hosting/data-center-infrastructure, acedido a última vez em 9 de Fevereiro de 2016.

[11] Yushi Shen *et. al*. *Enable the New Era of Cloud Computing*. Inprint Science Reference, 2014.

[12] N. Loutas, E. Kamateri, F. Bosi, e K. Tarabanis. Cloud computing interoperability: The state of play. Em *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, páginas 752–757, Nov 2011.

[13] István Mezgár e Ursula Rauschecker. The challenge of networked enterprises for cloud computing interoperability. *Computers in Industry*, 65(4):657 – 674, 2014. URL: http://www.sciencedirect.com/science/article/pii/S0166361514000335.

[14] Occi. An Open Community Leading Cloud Standards. Disponível em http://occi-wg.org/, acedido a última vez em 11 de Fevereiro de 2016.

[15] Open Grid Forum. OGF Documents and Standards. Disponível em https://www.ogf.org/ogf/doku.php/standards/standards, acedido a última vez em 11 de Fevereiro de 2016.

[16] SNIA. Cloud Data Management Interface (CDMI). Disponível em http://www.snia.org/cdmi, acedido a última vez em 11 de Fevereiro de 2016.

[17] SNIA. Advancing Storage and Information Technology. Disponível em http://www.snia.org/, acedido a última vez em 11 de Fevereiro de 2016.

[18] IEEE Standards Association. P2301 - Guide for Cloud Portability and Interoperability Profiles (CPIP). Disponível em https://standards.ieee.org/develop/project/2301.html, acedido a última vez em 11 de Fevereiro de 2016.

[19] IEEE Standards Association. P2302 - Standard for Intercloud Interoperability and Federation (SIIF). Disponível em https://standards.ieee.org/develop/project/2302.html, acedido a última vez em 11 de Fevereiro de 2016.

[20] IEEE Standards Association. Página principal. Disponível em http://standards.ieee.org/index.html, acedido a última vez em 11 de Fevereiro de 2016.

[21] DTMF. Open Virtualization Format. Disponível em https://www.dmtf.org/standards/ovf, acedido a última vez em 11 de Fevereiro de 2016.

[22] DTMF. Página principal. Disponível em https://www.dmtf.org/standards/cloud, acedido a última vez em 11 de Fevereiro de 2016.

[23] Cloud Security Alliance. Página principal. Disponível em https://cloudsecurityalliance.org/, acedido a última vez em 11 de Fevereiro de 2016.

[24] Cloud Standard Customer Council. Cloud Security Standards: What to Expect & What to Negotiate, Outubro 2013. Disponível em http://www.cloud-council.org/deliverables/CSCC-Cloud-Security-Standards-What-to-Expect-and-What-to-Negotiate.pdf, acedido a última vez em 11 de Fevereiro de 2016.

[25] A.M. AlZadjali, A.H. Al-Badi, e S. Ali. An analysis of the security threats and vulnerabilities of cloud computing in oman. Em *Intelligent Networking and Collaborative Systems (INCOS), 2015 International Conference on*, páginas 423–428, Setembro 2015.

[26] P. Deshpande, S.C. Sharma, e P.S. Kumar. Security threats in cloud computing. Em *Computing, Communication Automation (ICCCA), 2015 International Conference on*, páginas 632–636, Maio 2015.

[27] LLRX.com. Security for Intranets and Extranets. Disponível em http://www.llrx.com/extras/security.htm, acedido a última vez em 11 de Fevereiro de 2016.

[28] TechTarget. Firewall. Disponível em http://searchsecurity.techtarget.com/definition/firewall, acedido a última vez em 11 de Fevereiro de 2016.

[29] Paul Hoffman e Karen Scarfone. Guidelines on Firewalls and Firewall Policy, Setembro 2009. Disponível em http://csrc.nist.gov/publications/nistpubs/800-41-Rev1/sp800-41-rev1.pdf, acedido a última vez em 8 de Fevereiro de 2016.

[30] Microsoft TechNet. The TCP/IP model. Disponível em http://www.interoute.com/what-saas, acedido a última vez em 9 de Fevereiro de 2016.

[31] Wikipedia. iptables. Disponível em https://en.wikipedia.org/wiki/Iptables, acedido a última vez em 11 de Fevereiro de 2016.

[32] IPTABLES. IPtables Manual. Disponível em http://ipset.netfilter.org/iptables.man.html, acedido a última vez em 30 de Maio de 2016.

[33] H. Mohamed, L. Adil, T. Saida, e M. Hicham. A collaborative intrusion detection and prevention system in cloud computing. Em *AFRICON, 2013*, páginas 1–5, Setembro 2013.

[34] Snort. Página principal. Disponível em https://www.snort.org/, acedido a última vez em 11 de Fevereiro de 2016.

[35] Open Source HIDS SECurity. Página principal. Disponível em http://ossec.github.io/, acedido a última vez em 11 de Fevereiro de 2016.

[36] Suricata. Página principal. Disponível em http://suricata-ids.org/, acedido a última vez em 11 de Fevereiro de 2016.

[37] The Bro Network Security Monitor. Página principal. Disponível em https://www.bro.org/index.html, acedido a última vez em 11 de Fevereiro de 2016.

[38] Mauno Pihelgas. A Comparative Analysis of Opensource Intrusion Detection Systems, 2012. Disponível em http://mauno.pihelgas.eu/files/Mauno_Pihelgas-A_Comparative_Analysis_of_OpenSource_Intrusion_Detection_Systems.pdf, acedido a última vez em 11 de Fevereiro de 2016.

[39] Surya Bhagvan Ambati e Deepti Vidyarathi. A brief study and comparison of open source intrusion detection system tools. Dezembro 2013.

[40] Pritika Mehra. A brief study and comparison of snort and bro open source network intrusion detection systems. Agosto 2012.

[41] Chintan Kacha e Kirtee A. Shevade. Comparison of different intrusion detection and prevention systems. Agosto 2012.

[42] David J. Day e Benjamin M. Burns. A performance analysis of snort and suricata network intrusion detection and prevention engines.

[43] pytbull. Official Documentation. Disponível em http://pytbull.sourceforge.net/index.php?page=documentation, acedido a última vez em 5 de Junho de 2016.

[44] KALI. Página Inicial. Disponível em https://www.kali.org/, acedido a última vez em 23 de Abril de 2016.