UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Gleison de Oliveira Medeiros

Mitigando Ataques de Negação de Serviço em Infraestruturas de Computação em Nuvem

São Luís - MA
2014

Gleison de Oliveira Medeiros

Mitigando Ataques de Negação de Serviço em Infraestruturas de Computação em Nuvem

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Engenharia de Eletricidade.

Orientador: Zair Abdelouahab

Doutor em Ciência da Computação – UFMA

São Luís - MA

2014

Medeiros, Gleison de Oliveira

Mitigando Ataques de Negação de Serviço em Infraestruturas de Computação em Nuvem / Gleison de Oliveira Medeiros. – São Luís - MA, 2014.

102 f.

Orientador: Zair Abdelouahab.

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís - MA, 2014.

1. Computação em Nuvem. 2. Detecção de Intrusão. 3. Sistemas Multiagentes. 4. Virtualização. I. Abdelouahab, Zair, orient. II. Título.

CDU 004.056.53

Gleison de Oliveira Medeiros

Mitigando Ataques de Negação de Serviço em Infraestruturas de Computação em Nuvem

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Gleison de Oliveira Medeiros e aprovada pela comissão examinadora.

Aprovada em 04 de Setembro de 2014

BANCA EXAMINADORA

Zair Abdelouahab (orientador)

Doutor em Ciência da Computação – UFMA

Denivaldo Cicero Pavão Lopes

Doutor em Informática – UFMA

Leila Weitzel Coelho da Silva

Doutora em Ciência da Computação – UNIFESSPA

À mim, por sempre tentar e nunca desistir.

Resumo

A Computação em Nuvem tem sido bastante utilizada como uma solução recente para as demandas crescentes de usuários de serviços de Tecnologia da Informação. Concomitante ao desenvolvimento acelerado desta tecnologia, novas formas de intrusão (e de intrusos) acabam emergindo. Nesse contexto, é proposta nesta dissertação, uma arquitetura de sistema de detecção de intrusão, baseada em sistemas Multiagentes, associadas ao uso de técnicas de criptografia e assinatura digital, para detectar e tratar as tentativas de ataques de DoS (*Denial of Service*) em uma Infraestrutura de Nuvem IaaS (*Infrastructure as a Service*). Os resultados parciais obtidos com a implementação de um protótipo são considerados satisfatórios, tendo em vista o desempenho apresentado pela ferramenta.

Palavras-chaves: Computação em Nuvem, Detecção de Intrusão, Virtualização, Multiagentes, Criptografia, Negação de Serviço.

Abstract

Cloud computing has been widely used as an ultimate solution for the growing demands of users of Information Technology services. Concomitant to the rapid development of this technology, new forms of intrusion (and intruders) have emerged. In this context, it is proposed in this dissertation, an architecture of intrusion detection system based on Multi-agent systems associated with the use of techniques for encryption and digital signature to detect and treat the attempted denial of service attacks in an Infrastructure Service. The partial results obtained with the implementation of a prototype are considered satisfactory in view of the performance presented by the tool.

Keywords: Cloud Computing, Intrusion Detection, Virtualization, Multi-agent, Cryptography, Denial of Service.

Agradecimentos

Agradeço primeiramente a Deus pela oportunidade da vida.

Ao meu orientador Profº PhD. Zair Abdelouahab, pela confiança, ensinamento, paciência e por nunca ter desistido desse projeto.

Ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão por abrir as portas e permitir a expansão do conhecimento.

A minha Mãe Lourdes de Oliveira e meu Pai Raimundo Cabral, que sempre apoiaram minhas decisões.

A minha Irmã Leydiane Oliveira, Minha Tia Elcimeire Oliveira e meu Tio Ribamar Garcia, pelo apoio logístico e moral nas horas difíceis.

Aos meus Primos, Michel Oliveira, Wanessa de Oliveira e Webbert de Oliveira, sem vocês com certeza teria sido muito mais difícil, Obrigado!

Ao Amigo Duandys Ferreira, por sua contribuição na elaboração do projeto.

Aos Amigos do Laboratório de Segurança e Arquiteturas Computacionais - LABSAC, em especial à Amiga Dhileane Rodrigues e ao Amigo Vladimir Oliveira.

Aos Amigos de trabalho da Faculdade de Computação e Engenharia Elétrica da Universidade Federal do Sul e Sudeste do Pará, em especial a Profª Drª Leila Weitzel por sua contribuição.

À Todos que torceram por mim.

"E quem um dia irá dizer que existe razão nas coisas feitas pelo coração? E quem irá dizer que não existe razão?"

Renato Russo.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AA	*Analyzer Agent*.
ACK	*Acknowledgement*.
ARE	*Agente Runtime Environment*.
BDT	*Behavioral Data Table*.
CARE	*Client Agent Runtime Environment*.
CERT.br	*O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil* .
CPU	*Central Processing Unit*.
DDoS	*Distributed Denial of Service* .
DIDS	*Distributed Intrusion Detection System*.
DoS	*Denial of Service* .
DPI	*Deep Packet Inspection* .
EICIDS	*Elastic and Internal Cloud-based Intrusion Detection System*.
ESA	*Event Signaling Agent*.
FIPA	*Foundation for Intelligent Physical Agents*.
GAE	*Google App Engine*.
GCCIDS	*Grid and Cloud Computing Intrusion Detection System*.
GPL	*General Public License*.
GUEST	*Guest Operating System*.
HIDS	*Host-based Intrusion Detection*.

HTTP	*Hypertext Transfer Protocol*.
IaaS	*Infrastructure as a Service*.
ICMP	*Internet Control Message Protocol*.
IDE	*Integrated Development Environment*.
IDMEF	*Intrusion Detection Message Exchange Format*.
IDS	*Intrusion Detection System*.
IP	*Internet Protocol*.
IPS	*Intrusion Prevention System*.
JADE	*Java Agent Development Framework*.
JVM	*Java Virtual Machine*.
KDT	*Knowledge Data Table*.
KVM	*Kernel-based Virtual Machine*.
LAN	*Local Area Network*.
MA-IDPS	*Multi-Agent Intrusion Detection and Prevention System*.
MAC	*Media Access Control Address*.
NIDS	*Network-based Intrusion Detection*.
NIST	*National Institute of Standards and Technology*.
PaaS	*Platform as a Service*.
PDA	*Personal Digital Assistant*.
PING	*Packet Internet Grouper*.
SA	*Smart Agent*.
SaaS	*Software as a Service*.
SAGs	*Scenario Attack Graph*.
SMDA	*Smart Malware Detection Agent*.

SYN	*Synchronize*.
TCP	*Transmission Control Protocol*.
TI	Tecnologia da Informação.
TILAB	*Telecom Italia Lab*.
UA	*Update Agent*.
UDP	*User Datagram Protocol*.
UML	*Unified Modeling Language*.
VA	*Verifier Agent*.
VM	*Virtual Machine*.
VMI-based IDS	*Virtual Machine Introspection Based Intrusion Detection System*.
VMM	*Virtual Machine Monitor*.
WIS	*Worldwide Infrastructure Security*.
XML	*eXtensible Markup Language*.

Sumário

1 Introdução

Neste capítulo, apresenta-se o cenário atual da segurança nos ambientes de Infraestrutura de Nuvem com ênfase na relação de dependência entre a sociedade moderna e o uso de tecnologias de informação. Em seguida, serão abordados os principais problemas referentes ao tema e objetivos definidos no escopo da pesquisa.

1.1 Cenário atual

Durante a última década a sociedade moderna tornou-se cada vez mais dependente dos recursos e serviços fornecidos pela Tecnologia da Informação (TI), sendo assim, uma tendência que vêm evoluindo a cada dia. Entretanto, essa evolução tecnológica trouxe também uma série de novos desafios, tornando a segurança dos sistemas fundamentalmente mais complexa. Em outras palavras, os sistemas, tornaram-se mais expostos à interações maléficas, sujeitando-se a um conjunto de ameaças a sua segurança [13].

Nesse contexto, a Computação em Nuvem tem sido bastante utilizada como uma solução recente para as demandas crescentes de usuários de serviços de TI. Para [12,41], a Computação em Nuvem é um modelo computacional em evolução que permite o acesso conveniente sob demanda a uma gama de recursos computacionais que podem ser rapidamente provisionados e liberados de acordo com a demanda.

Concomitante ao desenvolvimento acelerado das tecnologias de rede e de computadores, novas formas de intrusão (e de intrusos) acabam emergindo. Desse modo, um ambiente marcado pela contínua evolução tecnológica faz com que novos ataques tenham como resposta a elaboração de novas estratégias de proteção, o que leva a reformulação de novas técnicas de ataque, representando um ciclo de ataque e defesa. Por isso, a área da segurança da informação independente da tecnologia empregada deve estar em constante atualização para fornecer o mínimo de qualidade de serviço aos usuários finais [13].

Atualmente, a maior preocupação de segurança nos ambientes de nuvem vêm sendo direcionada para detecção e prevenção de intrusos, principalmente na camada de *Infrastructure as a Service* (IaaS) [2]. Para [36], a Computação em Nuvem sofre de vários ataques tradicionais já conhecidos por outros modelos, como por exemplo, os ataques de *Denial of Service* (DoS) ou negação de serviço e *Distributed Denial of Service* (DDoS) ou ataques distribuídos de negação de serviço, que podem comprometer todos os serviços fornecidos pela nuvem desde o desempenho e disponibilidade a integridade das informações dos usuários.

Para amenizar essa situação, algumas técnicas de segurança devem ser adotadas para prevenir e/ou minimizar os danos causados por essas vulnerabilidades, entre elas estão: *firewalls*, *Intrusion Detection System* (IDS) e *Intrusion Prevention System* (IPS), criptografia e autenticação. Nesse contexto, sistemas IDS's específicos para Computação em Nuvem ganham destaque por sua crescente concepção, projeção e implementação de técnicas de segurança executadas por sistemas Multiagentes [13].

Os sistemas Multiagentes possuem um modelo que é adequado para a construção de componentes autônomos que agem e interagem de modo flexível para alcançar os objetivos de segurança em ambientes inseguros, incertos e dinâmicos. Outra característica importante, é a possibilidade de trocas de mensagens criptografadas entre os agentes, elevando ainda mais o nível de segurança desses sistemas [50].

1.2 Problemática

As atuais intrusões e ataques direcionados a ambientes computacionais distribuídos (como a Computação em Nuvem) tornam-se cada vez mais especializados, forçando a adequação de uma segurança constante e em evolução [36]. Técnicas de segurança que funcionavam contra tipos específicos de ataques, podem se mostrar falhas diante de tecnologias recentes.

Uma das maiores preocupações entre profissionais de TI em relação à implantação e utilização da Computação em Nuvem, é à segurança da informação, pois de acordo com as estatísticas apresentadas pelo *O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil* (CERT.br) em 2013, cerca de 352.925

(Trezentos e cinquenta e dois mil e novecentos e vinte e cinco) casos de incidentes contra a segurança da informação foram notificados [10]. Para o ano de 2014 o relatório anual do oitavo *Worldwide Infrastructure Security* (WIS) aponta que ameaças contra ambientes computacionais tendem a ser um problema recorrente durante todo ano, especialmente ataques de DDoS contra *Data Centers* e ambientes de nuvem computacionais [40].

Diversos Profissionais da área da segurança da informação afirmam que os sistemas baseados em nuvem são mais vulneráveis a ataques DDoS e DoS, uma vez que praticamente todo o *hardware* é compartilhado por todos os usuários, tornando esses tipos de ataques, mais prejudiciais, ou seja, usuários maliciosos ou intrusos podem implementar suas aplicações de modo a consumir mais recursos físicos do sistema do que outros usuários, o que pode ocasionar a indisponibilidade total do sistema. [1,3].

Um levantamento anual realizado por pesquisadores da *Arbor Networks* [39], mostrou que houve um aumento significativo de ataques DDoS em ambientes de nuvem nos últimos anos, segundo eles, no ano de 2013 foram reportados ataques cinco vezes maiores em volume de dados do que os ataques reportados no ano anterior. A Figura 1.1 apresenta os resultados obtidos nessa pesquisa.

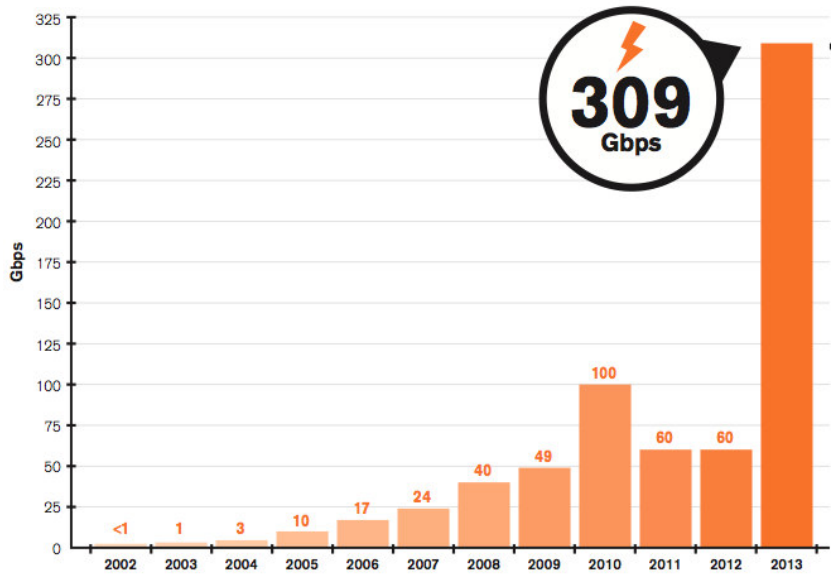


Figura 1.1: Notificação de Ataques DDoS, 2013. Adaptado de [39]

Os sistemas IDS's mostram-se grandes aliados dos administradores de segurança e, por isso, são considerados ferramentas essenciais no combate aos ataques DDoS e DDoS.

Pesquisas recentes voltadas a utilização de agentes no desenvolvimento de sistema IDS, vêm proporcionando ganhos significativos em eficiência, escalabilidade e cooperação. Além disso, técnicas de detecção são constantemente aprimoradas, principalmente em operações que envolvem análise de comportamento e ações de contenção à ataques [23].

No entanto, sistemas de detecção que utilizam agentes para garantir a segurança de redes distribuídas abertas, não necessariamente fazem deles sistemas verdadeiramente seguros [13]. A necessidade de garantir a segurança dos próprios componentes desse sistema, também pode ser considerada um ponto fundamental para a garantia de qualidade de serviço [49].

1.3 Objetivos Gerais e Específicos

O objetivo geral desta pesquisa é propor um modelo de Detecção de Intrusos na Camada IaaS da Computação em Nuvem e, desta forma, prover a disponibilidade da infraestrutura em situações de sobrecarga dos recursos de rede causadas por ataques de DDoS e DoS.

Para alcançar o objetivo geral, os seguintes objetivos específicos são pertinentes:

- Apresentar uma arquitetura de detecção baseada em sistemas Multiagentes;
- Prover autonomia e elasticidade ao sistema de detecção;
- Garantir a integridade e a confidencialidade das informações compartilhadas entre os componentes do sistema;
- Apresentar um modelo de identificação e neutralização de usuários maliciosos;
- Desenvolver um protótipo para avaliar o desempenho do modelo proposto;

1.4 Metodologia

A pesquisa foi dividida em etapas, que vão desde as definições, conceitos básicos, revisão literária e trabalhos relacionados, até a definição da problemática,

prototipação, implementação e contribuições da dissertação. A seguir a estrutura metodológica que levou a contribuição desta pesquisa.

1. **Busca do Tema**: A segurança da informação é um dos fatores mais importantes na atualidade e pode ser afetada por ações comportamentais e de uso de quem a utiliza, pelo ambiente ou infraestrutura que a cerca ou por pessoa mal-intencionada que tem o objetivo de furtar destruir ou modificar tal informação. Nesse contexto, muitas soluções precisam ser criadas ou evoluídas, o que permite explorar e desenvolver novos conceitos, aplicações e metodologias de defesa. O tema escolhido e abordado nesta pesquisa, envolve a criação de um modelo de detecção e mitigação de ataques de negação de serviço em Infraestruturas da Computação em Nuvem.

2. **Revisão Literária**: Iniciou-se com a revisão atualizada da literatura relacionada ao tema, buscando avaliar criticamente os estudos, as metodologias e os resultados de cada trabalho.

3. **Definição do Problema**: Com base na revisão literária pode-se observar que muitas metodologias de detecção de intrusão para Computação em Nuvem abrangem um cenário genérico de atuação. Diante disto, a problemática que foi explorada nesta pesquisa envolve a criação de um modelo de detecção e mitigação de ataques de negação de serviço na camada de virtualização de Infraestrutura de Nuvem, afim de neutralizar e minimizar os efeitos causados por estas ameaças.

4. **Modelo Proposto**: Implementar um modelo de detecção de ataques de negação de serviço com base na análise de tráfego proveniente do ambiente virtualizado da Nuvem.

5. **Implementação e Prototipagem**: A proposta foi implementada utilizando máquinas virtuais, agentes de software, ferramentas de modelagem de software e softwares de análise de *stress* de rede. As ferramentas selecionadas e utilizadas no decorrer da pesquisas são:

   (a) *Java Eclipse Integrated Development Environment* (IDE): Constitui em um ambiente para a integração de ferramentas de desenvolvimento [20];

(b) *Java Agent Development Framework* (JADE): Simplifica a implementação de sistemas Multiagentes através de um "*Middleware*" que cumpre as especificações *Foundation for Intelligent Physical Agents* (FIPA), e utiliza um conjunto de ferramentas gráficas que dão suporte a *Debuging* e as fases de desenvolvimento [47];

(c) *Astah*: Ferramenta para modelagem de dados orientado a objetos, usada para especificar, construir, visualizar e documentar um sistema de software baseada nos requisitos da *Unified Modeling Language* (UML) [45];

(d) *JMeter*: Ferramenta utilizada para testes de carga em serviços oferecidos por sistemas computacionais [46];

(e) *Kernel-based Virtual Machine* (KVM): Ferramenta para virtualização completa de *hardware* baseada em *Linux* [48];

(f) *Hping*: Ferramenta utilizada para gerar e analisar pacotes TCP/IP [53];

1.5 Estrutura da Dissertação

Esta dissertação encontra-se organizada em seis capítulos. No Primeiro Capítulo é apresentado o cenário atual do mercado da Computação em Nuvem e suas questões atuais de segurança, especificamente aos ataques de negação de serviço e a utilização de sistemas de detecção de intrusão. O crescimento progressivo do uso desse modelo de computação e o aumento de incidentes registrados nos últimos anos, bem como os objetivos gerais e específicos e a organização da dissertação.

No Capítulo 2, apresenta-se os conceitos e características básicas da Computação em Nuvem, dos problemas e soluções de segurança que ocorrem neste modelo, enfatizando-se as vantagens dos sistemas de detecção de intrusão baseados em agentes e o uso de técnicas de criptografia.

No Capítulo 3, é discutida algumas pesquisas recentes sobre sistemas de detecção de intrusão para Computação em Nuvem e suas contribuições para a realização desta pesquisa.

No Capítulo 4, é proposto um modelo de detecção de intrusão para camada IaaS da Nuvem baseado na tecnologia de Agentes e apresentando-se a sua arquitetura e funcionalidade através dos diagramas da Notação UML e *agent* UML.

No Capítulo 5 mostra-se a operacionalização do modelo proposto, destacando-se a implementação do protótipo e o resultados obtidos das simulações realizadas, através de pacotes capturados e analisados.

O Capítulo 6 apresenta as conclusões finais, enfatizando as contribuições desta dissertação e algumas indicações para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, serão apresentados os conceitos fundamentais das tecnologias que serviram de base para o desenvolvimento da proposta. Tais conceitos tiveram papel relevante no entendimento individual de cada tecnologia e na obtenção do conhecimento para melhor integrá-las.

2.1 Fundamentos da Computação em Nuvem

Com o rápido desenvolvimento das tecnologias de processamento e armazenamento de dados e o sucesso da *Internet*, os recursos de computação tornaram-se mais baratos, melhores e mais onipresentes. Desta forma, essa evolução tecnológica tem possibilitado a realização de um novo modelo de computação denominado *Cloud Computing* ou Computação em Nuvem [69].

Segundo [35], na computação tradicional, utilizando computadores *desktops*, as cópias de *softwares* são armazenadas em cada computador isoladamente e os documentos criados por usuários são geralmente armazenados nos computadores que foram criados. Entretanto esses documentos podem ser acessados a partir de outros computadores na mesma *Local Area Network* (LAN), porém é pouco provável que eles possam ser acessados por computadores de fora desta LAN. Por outro lado, na Computação em Nuvem, nem todos os *softwares* são executados diretamente nos computadores *desktop*, sendo, em sua maioria, armazenados e executados em computadores provedores de serviços de Nuvem, acessados diretamente via *Internet* pelos locatários dos serviços [69].

O princípio básico por trás deste novo conceito, é que ele atribui os recursos de computação a um número grande de computadores distribuídos em rede ao invés de computadores locais ou servidores remotos, ou seja, os recursos computacionais são compartilhados por todos os usuários do serviço [12].

Segundo [22], uma vantagem interessante neste modelo é que os usuários destes serviços tornam-se isentos de preocupações corriqueiras ocorridas em modelos

tradicionais de computação, como por exemplo, as manutenções e atualizações de *softwares*, *hardwares* e *backup* de arquivos. Por outro lado, tais responsabilidades passam a pertencer exclusivamente ao provedor de serviços em Nuvem.

Nesse contexto, a Computação em Nuvem pode ser resumida como um modelo computacional que permite acesso conveniente sob demanda a um grupo compartilhado de recursos computacionais configuráveis e que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor dos serviços [6, 12, 28, 68].

Para [61], a Computação em Nuvem é considerada uma camada conceitual que engloba a maioria dos serviços computacionais disponíveis, abstraindo para o usuário toda a sua Infraestrutura física e lógica.

Apesar de ser um assunto relativamente recente, a Computação em Nuvem por si só não é uma inovação tecnológica. Segundo [12], ela baseia-se em diversas tecnologias já existentes nos modelos tradicionais, por exemplo, a Virtualização de Computadores, *Grid Computing*, *Utility Computing* e o modelo *Pay-Per-Use* (modelo de pagamento baseado no uso, análogo aos serviços de telefonia e energia elétrica).

2.1.1 Modelos de Serviços

De acordo com [22], na Computação em Nuvem existem modelos de oferta de serviços que classificam a forma como os recursos são oferecidos para os usuários, estes modelos são descritos conceitualmente pelo *National Institute of Standards and Technology* (NIST) como:

***Software as a Service* (SaaS)**

Modelo que fornece ao consumidor a capacidade de utilizar aplicativos diretamente do provedor de Nuvem. As aplicações são acessíveis a partir de diversos dispositivos clientes por meio de uma *interface* cliente, por exemplo, um navegador *Web* [7].

Neste modelo o consumidor não gerencia ou controla a Infraestrutura de Nuvem, ou seja, não há controle rede, sistemas operacionais ou o armazenamento das instâncias que executam os serviços. Exceto a configuração de aplicativos específicos do próprio usuário mantedor do serviço [32].

Os prestadores de serviços disponibilizam o modelo SaaS como uma camada de execução de *softwares* na Nuvem. Do ponto de vista do usuário, esse modelo denota diminuição dos custos na aquisição de licenças de uso dos *softwares*, tendo em vista que todos os custos da utilização dos *softwares* estão incluídos na contratação dos serviços junto ao provedor de Nuvem [7,22].

***Platform as a Service* (PaaS)**

Modelo que fornece ao consumidor o serviço de implantar na Infraestrutura de Nuvem, a capacidade de criar ou adquirir aplicações desenvolvidas usando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor do serviço. Neste modelo assim como no modelo SaaS, o consumidor não gerencia ou controla a Infraestrutura de Nuvem, porém, tem controle sobre os aplicativos desenvolvidos na plataforma e liberdade de possíveis ajustes de configuração do ambiente de desenvolvimento e hospedagem dos aplicativos [7,32].

No PaaS, os desenvolvedores podem obter um pacote com todos os sistemas e ambientes necessários para o ciclo de vida de um *software*, ou seja, desenvolvimento, teste, implantação e a hospedagem de aplicações *Web*. Exemplos importantes de SaaS, são os serviços fornecidos pelo *Google App Engine* (GAE) e o *Microsoft Azure* [12].

IaaS

Modelo que fornece ao consumidor os serviços de processamento, armazenamento, redes e outros recursos de computação. De acordo com [26], nesse modelo o consumidor é capaz de implementar e executar *softwares*, incluindo sistemas operacionais, recursos de armazenamento e possivelmente obter controle limitado de componentes de rede, por exemplo, *firewalls* [26].

Para [7,22,26], no IaaS o usuário da Nuvem é responsável pela aplicação de *patches* e manutenção dos sistemas operacionais e dos aplicativos de *softwares*. Além disso, os provedores de Nuvem obtém receitas com base no controle de consumo da computação alocada, isto é, o custo reflete a quantidade de recursos alocados e consumidos em determinados picos de tempo [7].

Exemplos de provedores de IaaS incluem *Amazon CloudFormation*, *Amazon EC2*, *Windows Azure Virtual Machines*, *Google Compute Engine*, entre outros [32].

2.1.2 Modelos de Desenvolvimento de Nuvem

As nuvens computacionais também apresentam modelos de desenvolvimento que implicam na restrição de acesso aos usuários. Segundo [42, 64], os modelos de desenvolvimento são classificados em: Nuvens Privadas, Nuvens Públicas, Comunitárias e Híbridas. Esses modelos são descritos a seguir:

Nuvem Pública

Infraestrutura de Nuvem disponibilizada para o público em geral. Neste modelo, o provedor do serviço em Nuvem, disponibiliza ao cliente recursos através de uma *interface Web*, sempre de forma escalável, cobrando apenas o que se usa e sem necessidade de grandes investimentos em Infraestrutura por parte do consumidor [64].

Nuvem Privada

Neste modelo de serviço, a Infraestrutura, o gerenciamento e as operações efetuadas na Nuvem, são realizadas por uma organização particular. O acesso a informação pode ser restrito por políticas de segurança particulares. O uso de recursos também é disponibilizado através de *interface Web* assim como na Nuvem pública. Nuvem privada, geralmente são implantadas em *Data Center* existente na própria organização [42].

Nuvem Comunitária

Em nuvens comunitárias a infraestrutura é administrada especialmente por um conjunto de organizações cujo o gerenciamento pode está sujeito a regras estabelecidas pela comunidade proprietária [64].

Nuvem Híbrida

Trata-se de um grupo de nuvens que embora mantenham sua identidade diferenciada entre os demais modelos, podem ser construídas na combinação de dois ou mais modelos citados anteriormente [64]. Para [42], as nuvens pertencentes a esta categoria, podem está associadas entre si por protocolos ou padrões técnicos.

2.1.3 Características da Computação em Nuvem

Independente dos modelos de serviços e de desenvolvimento apresentados anteriormente, as nuvens computacionais devem seguir algumas características essenciais [63], por exemplo:

On-Demand Self-Service

Essa característica permite ao consumidor alocar ou usar os serviços da Nuvem a qualquer momento, conforme seja a demanda. Neste caso, sem a necessidade de interação humana com o provedor do serviço.

Ubiquitous Network Access

Característica que permite acesso ubíquo aos recursos de Nuvem. São utilizados mecanismos padrões que promovem o uso de plataformas heterogêneas. Por exemplo, telefones celulares, *laptops* e *Personal Digital Assistant* (PDA).

Resource Pooling

Característica que permite aos usuários da Nuvem, alocação dinâmica de recursos computacionais. Por exemplo, recursos de armazenamento, processamento, memória, largura de banda de rede e máquinas virtuais.

Rapid Elasticy

Permite provisionamento escalável ou a capacidade de fornecer serviços escaláveis. Do ponto de vista do usuário consumidor, pode-se adquirir a capacidade computacional que for necessária para suas aplicações a qualquer momento.

Measured Service

Essa característica permite ao provedor de Nuvem, controlar e otimizar automaticamente o uso dos recursos, oferecendo transparência tanto para o provedor quanto para o consumidor do serviços.

2.2 Infraestrutura de Plataforma de Nuvem

Em uma plataforma de Nuvem os serviços de IaaS, PaaS e SaaS, são disponibilizados por meio de um grande número de recursos computacionais, tais

recursos, são possíveis por meio de uma infraestrutrua computacional virtualizada que proporciona flexibilidade e eficiência aos serviços [44]. Segundo [64], com essa técnica é possível fornecer um conjunto de recursos virtualizados que compartilham os mesmos recursos físicos de *hardware*, por exemplo, os recursos de processamento, memória primária, armazenamento em disco e largura de banda.

A virtualização ressurgiu nos últimos anos como uma abordagem atraente para aumentar a utilização de recursos físicos de *hardware* e com isto evitar a subutilização. Para [44], A virtualização funciona como uma camada computacional lógica entre a Infraestrutura física real e os processos computacionais.

Em um nível básico, a tecnologia de virtualização permite a abstração do recurso físico subjacente [9], ou seja, o recurso físico pode ser dividido em recursos lógicos ou virtuais conforme necessário. Segundo [54], essa característica é conhecida como provisionamento de recursos.

Com o gerenciamento adequado do provisionamento, os recursos lógicos podem ser disponibilizados dinamicamente, sendo assim, os recursos podem ser entregues de acordo com a demanda de cada usuário [64]. Para [54] essa característica é conhecida como provisionamento dinâmico dos recursos virtuais.

Para se construir um computador com elevado poder de processamento compartilhado, os recursos devem ser provisionados e gerenciados dinamicamente em tempo real para que cada usuário possa desfrutar com qualidade dos serviços oferecidos pelos provedores [9, 54]. A técnica de executar vários sistemas operacionais heterogêneos no mesmo *hardware* físico só é possível, com inclusão de uma camada extra de gerenciamento de recursos de *hardware* e *software*. Tal camada, é denominada de *Hypervisor* ou *Virtual Machine Monitor* (VMM) [44].

O *Hypervisor* é responsável pela virtualização e controle dos recursos físicos compartilhados pelas máquinas virtuais ou *Virtual Machine* (VM), por exemplo, processadores, dispositivos de entrada e saída, memória e armazenamento [44]. Além do gerenciamento dos recursos de *hardware*, o *Hypervisor* é responsável pelo gerenciamento do sistema operacional *Guest Operating System* (GUEST) executado na VM [18].

Os trabalhos de [14, 59], descrevem dois tipos de Monitores de Máquinas Virtuais:

***Hypervisor* Tipo 1**

Neste modelo, o *Hypervisor* é executado diretamente no *hardware* subjacente igualmente ao sistema operacional tradicional. Desta forma, os sistemas operacionais *Guest* são executados em uma camada acima do *hardware*, logo acima do *Hypervisor*. A Figura 2.1 demonstra as características deste modelo.

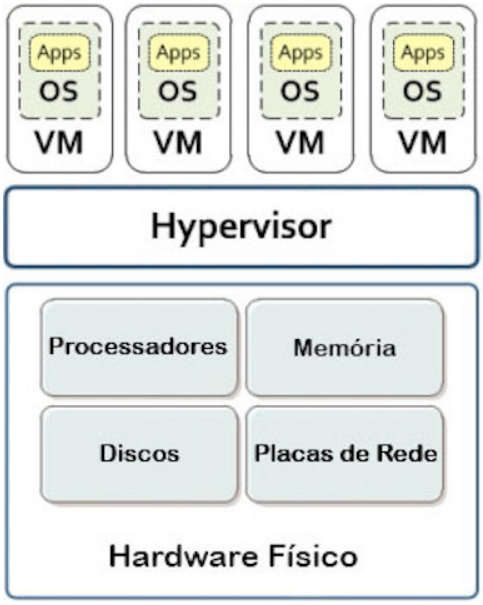


Figura 2.1: Arquitetura de *Hypervisor*. Adaptado de [14]

***Hypervisor* Tipo 2**

Neste modelo, toda a camada de gerenciamento de VM é aplicada sobre sistema operacional subjacente. Os sistemas operacionais *Guest* executam em um terceiro nível acima do *Hypervisor*. Como apresentado na Figura2.2.

Segundo [14, 18], existem duas técnicas para virtualização.

Virtualização Completa

Técnica que consiste em permitir que qualquer sistema *guest* possa ser executado sem qualquer alteração no seu *kernel* [1] [8]. Desta forma, uma simulação completa do *hardware* da máquina real é realizada de modo que qualquer sistema operacional possa ser executado.

[1] É o componente central do sistema operacional da maioria dos computadores; ele serve de ponte entre aplicativos e o processamento real de dados feito a nível de *hardware*

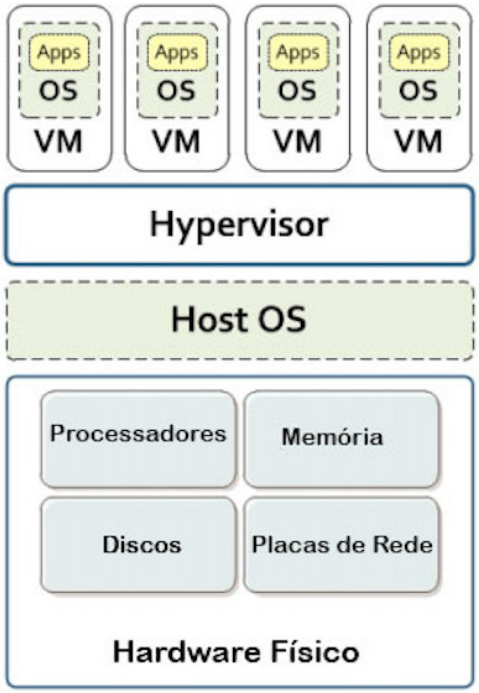


Figura 2.2: Arquitetura de *Hypervisor* Tipo 02. Adaptado de [14]

Segundo [18], para que essa técnica seja eficiente toda a Infraestrutura do *hardware* subjacente deve ser virtualizada. Entretanto podem ocorrer perdas de desempenho do ponto de vista da VM, uma vez que todo o *hardware* é virtualizado e as instruções de máquina devem ser interpretadas pelo *Hypervisor*.

Paravirtualização

Nessa técnica, diferentemente da virtualização completa, o sistema operacional *Guest* sofre alterações no seu *Kernel*. Desta forma, é possível uma interação mais eficiente com o *Hypervisor* [14]. Entretanto nessa abordagem o sistema virtualizado perde em portabilidade [18]. Por outro lado, será possível acessar diretamente os recursos físicos reais do *hardware*, isto possibilita aumento de desempenho em relação ao modelo de virtualização completa.

Atualmente estão disponíveis no mercado inúmeras ferramentas destinadas å construção de ambientes virtualizados. Muitas delas são de uso livre e de código aberto, baseadas em licença *General Public License* (GPL) [14]. Todavia, nem todas as ferramentas disponíveis no mercado são ferramentas recomendadas para a construção de Infraestrutura de Nuvem. Uma pesquisa pertinente ao desempenho de ferramentas de virtualização realizada por [59], mostrou que as ferramentas mais utilizadas na

construção de Infraestruturas de Nuvem são: VMware ESXi, Xen Server e KVM. Na Tabela 2.1 é possível notar as características específicas de cada uma.

Tabela 2.1: Características dos *Hypervisores* VMware ESXi, Xen Server e KVM

Características dos *Hypervisores* VMware ESXi, Xen Server e KVM. Adaptado de [18]

Nome do *Hypervisor*	**Tipo de *Hypervisor***	**Sistema Operacional Hospedeiro**	**Sistema Operacional *Guest***
VMware ESXi	1	Nenhum	Linux, Windows, Solaris e Outros
Xen Server	2	Nenhum ou FreeBSD, Linux e Solaris	Linux, Solaris e Windows
KVM	2	Linux	Windows e Linux

2.3 Segurança e Vulnerabilidades da Informação

2.3.1 Conceitos

A segurança da informação diz respeito à proteção dos dados com a intenção de preservar seus respectivos valores para uma organização ou um indivíduo [29]. No entanto, a definição universal clássica da segurança da informação é definida por [15] como sendo: Confidencialidade, integridade e disponibilidade da informação, também conhecidos como os três pilares da segurança da informação.

Para [66], a confidencialidade significa que as informações devem permanecer secretas e somente as pessoas autorizadas podem obter acesso a elas. Segundo [29], o acesso não autorizado a informações confidenciais pode ter consequências devastadoras, não só em aplicações de segurança nacional, mas também no comércio e na indústria.

Os Principais mecanismos de proteção da confidencialidade em sistemas de informação, são desenvolvidos com base nos conceitos de criptografia, políticas acesso e de uso. Exemplos de ameaças à confidencialidade são os *malware*, intrusos, a engenharia social, meio de comunicação inseguros e sistemas mal administrados [66].

A integridade da informação é a área preocupada com a confiabilidade, origem, integridade e exatidão das informações, bem como a prevenção de modificação indevida ou não autorizada. Integridade no contexto de segurança da informação não se refere apenas a integridade da informação em si, mas também para a integridade da origem da informação [60]. Mecanismos de proteção da integridade

podem ser agrupados em dois grandes tipos: Mecanismos de Prevenção, tais como controles de acesso que impedem a modificação não autorizada de informações e Mecanismos de Detecção, que se destinam a detectar modificações não autorizadas quando os mecanismos preventivos falham [66].

A disponibilidade, embora mencionada por último não é o pilar menos importante, segundo [60], ela pode ser definida como sendo a propriedade de um sistema ou de um recurso ser acessível e utilizável sob demanda por uma entidade autorizada. A disponibilidade é tão importante e necessária quanto as propriedades de confidencialidade e integridade [29]. Ataques de DDoS e DoS são corriqueiros contra disponibilidade dos sistemas [19].

2.3.2 Segurança em Redes de Computadores

Em redes de computadores, as questões de segurança da informação estão focadas principalmente na comunicação entre os equipamentos e os protocolos de comunicação.

Na camada física, os meios de comunicação quando não estão bem isolados fisicamente podem representar possibilidade de interceptação dos dados. Na camada de enlace, as informações transmitidas em texto claro podem ser lidas quando interceptadas. Ná na camada de rede, *Internet Protocol* (IP) falso e tráfegos não permitidos devem ser bloqueados para que não haja fluxo desnecessário ou malicioso. Na camada de transporte é possível criptografar conexões inteiras, de um extremo ao outro, afim de impedir a captura e a interpretação dos dados [19].

Com o uso crescente das redes de computadores pela indústria, aumentou o risco de roubo de informações particulares. Embora essas ameaças possam exigir diversas contramedidas, com certeza, a ferramenta automatizada mais importante para a segurança da rede e das comunicações é a criptografia [60]. Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário é claro, deve estar habilitado a recuperar os dados a partir dos dados disfarçados. Para tal, é necessário que tanto o emissor como o destinatário compartilhem uma chave de criptografia e um algoritmo de criptografia, somente desta forma é que será possível tornar um texto cifrado a partir de um texto claro ou vice-versa.

Segundo [29], duas formas de criptografia são amplamente utilizadas: Convencional ou Simétrica, e Criptografia por Chave Pública ou Assimétrica.

Criptografia Simétrica

Técnica mais antiga e mais conhecida. Uma chave secreta, que pode ser um número, uma palavra ou apenas uma sequencia de letras aleatórias, é aplicada ao texto de uma mensagem para alterar o conteúdo de uma determinada maneira. Desde que o remetente e o destinatário saibam a chave secreta, eles podem criptografar e descriptografar todas as mensagens que usam essa chave [17,34].

Criptografia Assimétrica

Um problema com chaves simétricas está na ação de troca da chave secreta em ambientes hostil, por exemplo, a *Internet*. Qualquer pessoa que conheça a chave secreta pode descriptografar a mensagem. Uma resposta a este problema é a criptografia assimétrica, em que há duas chaves relacionadas e distintas. Uma chave pública é disponibilizada gratuitamente a qualquer pessoa que queira enviar uma mensagem e uma segunda chave privada, mantida em sigilo, para que somente o usuário dono saiba [17,34].

A técnica consiste em criptografar a mensagem usando a chave pública e descriptografar aplicando o mesmo algoritmo, porém, usando a chave privada correspondente. Qualquer mensagem que é criptografada usando a chave privada só pode ser descriptografada usando a chave pública correspondente.

Ataques em Redes de Computadores

Ataques à segurança das redes de computadores é definido por [29], como sendo qualquer ação que comprometa a segurança da informação pertencente a uma organização. Segundo [60], os ataques à segurança da informação podem ser classificados de duas formas:

Ataques Passivos

Possuem a natureza de bisbilhotar ou monitorar transmissões. O objetivo é obter informações em enalces de comunicação. Para [29], um exemplo de ataque passivo (escuta de tráfego) muito utilizado nas redes de computadores é apresentado na Figura 2.3. Nela é possível perceber um intruso tentando obter informações

confidenciais entre um emissor (Paulo) e um receptor (Aline) em um meio de comunicação inseguro.

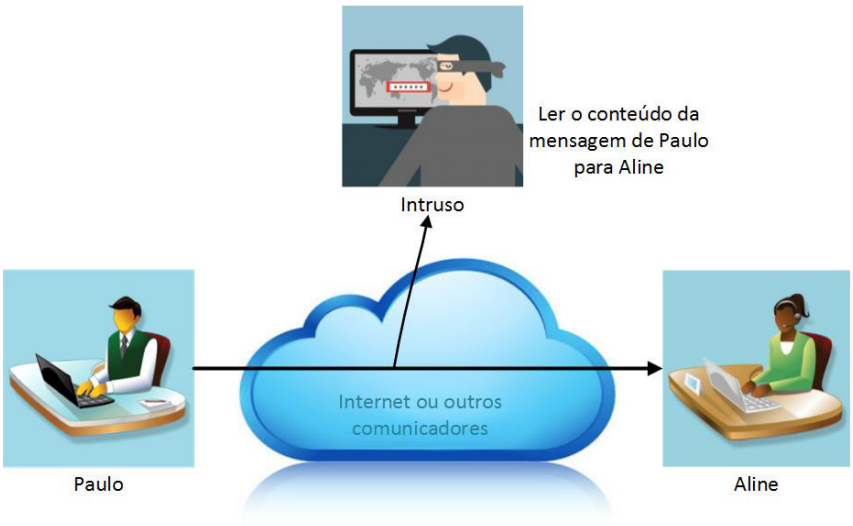


Figura 2.3: Ilustração Ataque Passivo - Escuta de Tráfego. Adaptado de [60]

Ataques passivos são muito difíceis de detectar, pois não envolvem alteração dos dados [29]. Normalmente, o tráfego de mensagens ocorre em um padrão aparentemente normal e, nem o emissor nem o receptor estão cientes de que um terceiro leu as mensagens ou observou o padrão de tráfego [60].

Ataques Ativos

Envolvem alguma modificação do fluxo de dados ou a criação de um fluxo falso. segundo [29], podem ser subdivididos em quatro categorias:

- **Disfarce**: Quando um entidade finge ser uma entidade diferente. Um ataque de disfarce normalmente inclui umas das outras formas de ataque ativo.
- **Repetição**: Envolve a captura passiva de uma unidade de dados e sua subsequente retransmissão para produzir um efeito não autorizado.
- **Modificação da Mensagem**: Significa que alguma parte da mensagem legítima foi alterada ou que as mensagens foram adiadas ou reordenadas para reproduzir um efeito não autorizado.
- **Negação de Serviço ou DoS**: Impede ou inibe o uso ou o gerenciamento normal das instalações de comunicações, Segundo [60], esse ataque pode ter um alvo específico, por exemplo, um servidor ou um recurso de rede específico. Uma

outra forma de ataque deste tipo é conhecida como ataque distribuído de negação de serviço. Ao contrário do ataque normal, no qual um computador e uma conexão com a *Internet* são usados para inundar um recurso ou servidor alvo, um ataque distribuído utiliza muitos computadores e muitas conexões com a *Internet*, muitas vezes distribuídos globalmente.

Os ataques ativos apresentam características opostas aos ataques passivos. Embora ataques passivos sejam difíceis de detectar, existem medidas para impedir seu sucesso. Por outro lado, é muito difícil impedir ataques ativos devido à grande variedade de vulnerabilidades físicas, de *software* e de rede. Em vez disso, o objetivo é detectar ataques ativos e recuperar-se de qualquer interrupção ou atrasos causados por eles [60].

Em [19], é descrita a classificação de ataques DDoS em Redes de Computadores. Os ataques estão divididos em três aspectos principais.

1. **Automação**: Significa o processo de preparação de um ataque DDoS e envolve a localização das *Botnets* [2] e o envio do comando contendo os dados do alvo e início de ataque;

2. **Vulnerabilidade**: Ataques DDoS podem explorar as falhas no sistema alvo para causar a negação de serviço;

3. **Impacto do ataque**: Os ataques DDoS podem ser também classificados quanto ao nível de ataque que causam as vítimas;

Ainda Segundo [19], os tipos de ataques amplamente utilizados em DDoS são:

User Datagram Protocol* (UDP) *Flooding

Este ataque DDoS utiliza-se das regras do protocolo UDP, um protocolo da camada de transporte não orientado a conexão [29]. Geralmente, o processo consiste em enviar em larga escala, grande quantidade de pacotes a portas aleatórias de um *host* alvo. Uma vez que o pacote chega ao seu destino, o *host* alvo verifica para qual porta e aplicativo aquele pacote está destinado. Após verificar que não existe serviço

[2]Coleção de *software* robôs que funcionam em computadores *host* de forma autônoma e automaticamente, controlados por um ou por vários invasores [37]

na porta indicada, o *host* alvo responde com um pacote *Internet Control Message Protocol* (ICMP) de destino inacessível, informando ao suposto usuário, que a solicitação não poderá ser atendida. Sendo assim, o *host* alvo acaba tendo que responder a uma grande quantidade de requisições [19]. Este processo limita os recursos do *host* e que em determinadas situações acaba tornando-se indisponível [25].

ICMP *Flooding*

Os ataques utilizando pacotes ICMP são, em princípio, similares aos ataques de inundação por UDP. Geralmente este tipo de ataque ocorre com o envio de pacotes do tipo *Packet Internet Grouper* (PING) ICMP *echo request* o mais rápidos possível ao *host* alvo sem que haja tempo de espera para as respostas [25]. Este tipo de ataque pode consumir toda a largura de banda de entrada e saída da rede, uma vez que o *host* alvo tentará responder as requisições com pacotes de resposta do tipo ICMP *echo Reply*.

Segundo [19], Um método conhecido de ataque ICMP *flooding* é o *Smurf attack*, neste tipo de ataque, o atacante envia um pacote do tipo ICMP *echo request* com endereço de origem da vítima para o endereço de *broadcast* de uma rede intermediária. Como resposta, todos os *hosts* da rede intermediária enviaram pacotes ICMP *echo reply* para a vítima.

Transmission Control Protocol* (TCP) *Synchronize* (SYN) *Flooding

O TCP SYN *flooding* atua com a intenção de sobrecarregar o alvo atacado através de solicitações sucessivas de conexão. O mecanismo consiste em explorar o processo de estabelecimento de conexão em três vias *Three-Way Handshake* do protocolo TCP [19]. A Figura 2.4 apresenta o formato para uma conexão normal utilizando o protocolo TCP. Nela é possível observar que para cada requisição TCP SYN utilizada para iniciar uma conexão TCP, é obrigatório o envio de uma resposta do tipo SYN *Acknowledgement* (ACK), em seguida uma confirmação ACK do solicitante.

Em um cenário de inundação SYN, os agentes iniciam múltiplas conexões que nunca são concluídas, esgotando os recursos do servidor [25]. A Figura 2.5 demonstra uma arquitetura para ataques de DDoS utilizando múltiplas conexões TCP. O atacante manipula uma máquina mestre para que a mesma coordene requisições em outras máquinas (Zumbis).

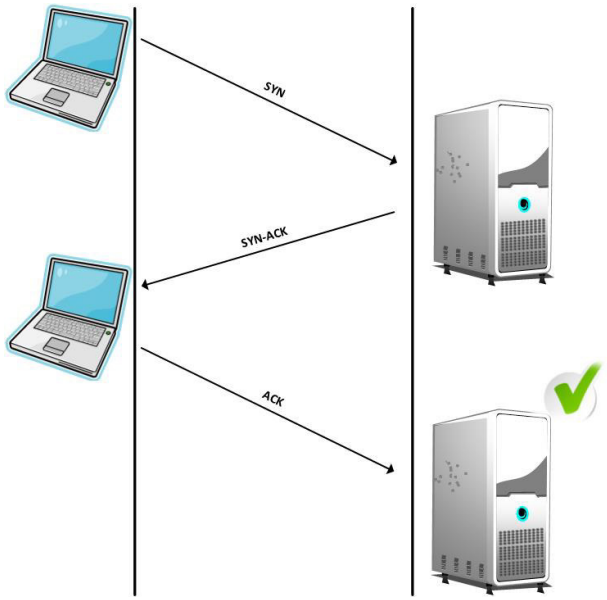


Figura 2.4: Etapas de Estabelecimento de Conexão TCP Normal. Adaptado de [19]

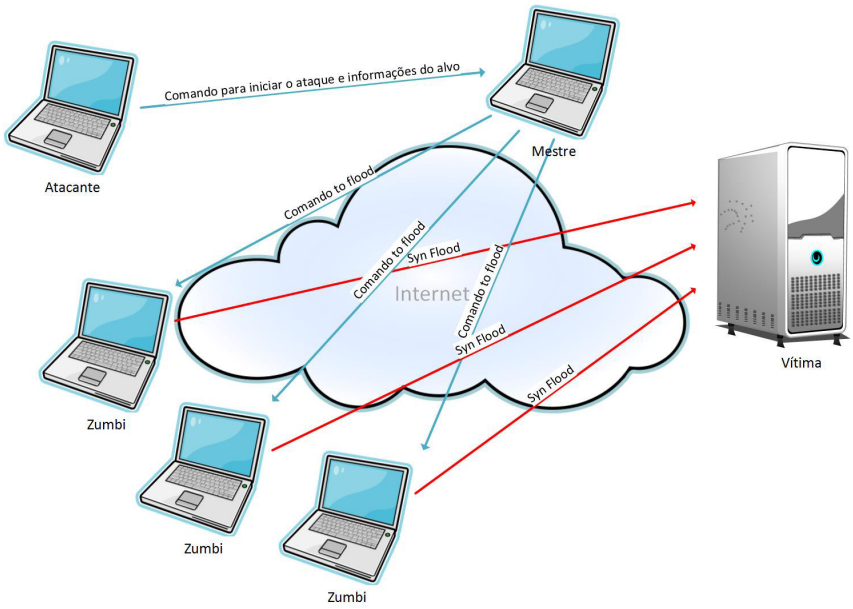


Figura 2.5: Ataque DDoS utilizando conexões TCP. Adaptado de [62]

2.3.3 Segurança em Computação em Nuvem

A Computação em Nuvem é uma das inovações da TI mais difundida na atualidade. A maioria das empresas nesse ramo de atividade, já utiliza ou pretende utilizar produtos que seguem os conceitos de Nuvem [57]. Porém, é notório que um dos principais problemas para aceitação consolidada desta inovação esteja relacionado às questões de segurança da informação. Segundo [43], há muitos problemas inerentes

a segurança da informação que devem ser considerados ao usar a Computação em Nuvem, são estes:

Confidencialidade

Neste caso, os provedores de Nuvem podem necessitar contratar empresas tercerizadas para armazenarem dados e informações de seus clientes, ou seja, em algum momento os dados do usuário pode se tornar exposto a terceiros.

Integridade

Quando os dados estão na Nuvem qualquer usuário que tenha acesso poderá acessá-lo [4]. Portanto, os provedores de Nuvem devem assegurar a integridade das informações, ou seja, o provedor deve ser capaz de impor restrições e privilégios de acesso aos usuários.

Roubo de Dados

A maioria dos provedores de Nuvem contratam servidores de outros prestadores de serviços, pois desta forma, além de tornar as operações mais fáceis de gerenciar, ainda reduz os custos operacionais. Contudo, há grande possibilidade de que os dados armazenados em outros servidores possam ser roubados por usuários mal-intencionados.

Exclusão dos Dados

Um usuário poderá excluir seus dados da Nuvem a qualquer momento, entretanto, não há garantias de que o dado será realmente excluído, neste caso, os clientes devem ser certificados de que os dados apagados no ambiente de Nuvem, não serão mais mantidos pelo provedor.

Segundo [36], existem várias invasões comuns que afetam a disponibilidade, confidencialidade e integridade dos recursos e serviços em Nuvem, são:

Ataque Interno ou (*Insider)*

Ocorre quando usuários autorizados fazem mau uso dos serviços fornecidos pelo provedor de Nuvem e tentam obter acessos não autorizados. Sendo assim, estes usuários podem cometer fraudes, divulgar informações a terceiros ou até mesmo modificar as informações de outros usuários intencionalmente [30]. De

acordo com [36], ataques deste tipo, denotam grave problema de confiança para o provedor, por exemplo, o ataque de DoS ocorrido contra os provedores da *Amazon Elastic Compute Cloud* [5].

Ataques de *Flood* Segundo [30], nestes ataques assim como nos ataques em redes de computadores tradicionais, o atacante tenta inundar a vítima enviando uma grande quantidade de requisições TCP, UDP e ICMP ou uma mistura deles. No caso da Nuvem, os pedidos de VM's são acessíveis por qualquer pessoa através da *Internet*, o que pode viabilizar ataques de DoS ou DDoS e a utilização da rede virtual como *Botnets*, neste caso, o atacante fará uso das instâncias virtuais para tentar inundar serviços em outros servidores externos e/ou internos.

Para [36], ataques de *Flood* afetam diretamente a disponibilidade dos serviços de Nuvem para os usuários autorizados. Ao atacar um servidor de Nuvem que forneça um determinado serviço, o atacante poderá causar a indisponibilidade desse serviço para outros usuários. Ataques desta magnitude são conhecidos como ataques de negação direta de serviço.

Ataques de negação direta fazem com que os recursos de *hardware* do servidor de virtualização atacado sejam completamente esgotados devido ao processamento das falsas solicitações, desta forma, outras instâncias de serviços disponíveis no mesmo *Hypervisor*, não serão capazes de realizar suas tarefas em tempo hábil [5]. Este tipo de ataque é conhecido como ataque indireto de negação de serviço [36].

Ataques a *Virtual Machine* e *Hypervisor*

Ataques deste tipo são executados com o objetivo de adquirir o controle total da camada virtual implementada na Nuvem. Ao obterem êxito, os atacantes podem ser capazes de comprometer todo o sistema de virtualização, desde o domínio de VM's e criação de *Botnets* ao controle total do próprio *Hypervisor* [5].

Ataques de *Backdoor*

Segundo [60], ataques deste tipo são considerados ataques passivos e permitem que o atacante obtenha acesso remoto ao *host* alvo, geralmente infectado com *malware*. O objetivo é controlar os recursos da vítima e, desta forma, torná-la um participante de uma *botnet*. No ambiente de Nuvem, o atacante poderá manipular

boa parte dos recursos da Nuvem, com isto, realizar ataques coordenados de DDoS a outros servidores ou aos próprios usuários da Nuvem [36].

Sistemas de *Firewall* protegem apenas pacotes de origem externa em uma rede, ou seja, é a primeira linha de defesa utilizada. Com base em regras e políticas de segurança, um *Firewall*, poderá negar ou permitir acesso a portas, protocolos e endereços IP em uma rede. Desta forma, ataques internos na rede geralmente não são detectados por esses sistemas de proteção [5].

Segundo [36], ataques DoS ou DDoS são complexos demais para serem detectados por sistemas de *firewall*. Por exemplo, se houver um ataque na porta 80 de um servidor *Web*, o *firewall* poderá não distinguir o bom tráfego do tráfego malicioso.

2.4 Sistemas de Detecção de Intrusão

Conforme visto anteriormente, existem vários tipos de ameaças direcionadas aos ambientes de nuvens computacionais. Segundo [5], Sistema de Detecção de Intrusão ou IDS é considerado a solução mais viável para resistir a essas ameaças.

De acordo com [58], o IDS tem como principal objetivo, identificar indivíduos que tentam utilizar um determinado sistema de forma não autorizada ou que desejam abusar de privilégios concedidos aos mesmos [31, 33, 51].

Para [16], um sistema IDS monitora o tráfego de uma rede em busca de sinais de intrusão, atividades de usuários não autorizados e a ocorrência de práticas mal-intencionadas, como usuários autorizados que tentam ultrapassar os limites de restrição de acesso aos recursos disponíveis. Desta forma, uma intrusão pode ser definida como um conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade desses recursos [31].

Para [58], um IDS é geralmente composto por três componentes principais:

- *Sensor*: Tem como objetivo coletar as informações do sistema;
- *Analisador*: Componente principal de um IDS. Sua funcão é analisar os dados anteriormente obtidos e, através da análise indicar se ocorreram ou não intrusões;

- *Interface com o usuário*: Deve apresentar os dados coletados e analisados de forma organizada para o administrador;

2.4.1 Tipos de Sistemas de Detecção de Intrusão para Computação em Nuvem

Embora muitos sistemas de deteção sejam vistos conceitualmente como compostos por um sensor, um analisador e uma *interface* de usuário, os tipos de dados examinados e gerados pelos IDS podem variar significantemente de acordo com suas características [31]. Algumas destas características são descritas a seguir por [13,31,58]:

Quanto à Arquitetura

- **Centralizada**: Arquitetura formada por componentes de monitoração de dados, analisadores e um gerenciador central. Ela possui a desvantagem de não ser escalável e possuir único ponto de falha;
- **Hierárquica**: Arquitetura formada por uma camada de coleta de dados, uma camada de analisadores, uma camada de tratamento de dados relevantes e por fim um coordenador central responsável pela ação a ser tomada. Porém, se uma camada falhar o sistema como um todo entrará em colapso, tornando o sistema vulnerável. Esta arquitetura possui ainda o problema de possuir as camadas como único ponto de falha;
- **Distribuída**: Arquitetura que não possui um gerenciador central. Ela possui componentes autônomos e independentes e escaláveis. Outra característica desta arquitetura é não possuir um único ponto de falha;
- **Híbrida**: Mecanismos centralizados interagem com módulos distribuídos, aproveitando as vantagens de cada um, tentando chegar a um ponto de equilíbrio entre desempenho, simplicidade, abrangência e robustez.

Quanto ao Posicionamento no Ambiente de Nuvem

Por ser um ambiente virtualizado, a Computação em Nuvem permite várias possibilidades de fonte de captura de tráfego. Para cada possibilidade há um

tipo específico de IDS. Nos trabalhos de [5, 36, 52], são descritas as características e particularidades de cada tipo:

***Network-based Intrusion Detection* (NIDS)**: Detectam acessos indesejáveis através de sensores situados em segmentos estratégicos da rede e são responsáveis pelo monitoramento de atividades suspeitas. Sua estrutura é baseada em um componente que recebe alarmes dos sensores e executa as contramedidas [13]. Ademais, o NIDS, se preocupa em analisar os pacotes de dados que trafegam somente pela rede. Estes pacotes são analisados e comparados com dados empíricos, para que desta forma, possa ser apontada sua natureza (maliciosa ou não).

Para detectar ações maliciosas na rede, o NIDS, compara o comportamento atual do usuário com comportamentos já observados anteriormente e considerados normais. O NIDS monitora principalmente cabeçalhos da camada IP e utiliza técnicas de detecção de intrusão baseado em assinaturas conhecidas de ataques e anomalias de tráfego [36].

Para [13], alguns pontos são considerados negativos em relação ao NIDS.

- Não examinam o tráfego total da rede, mas somente os segmentos no qual os sensores estão conectados;
- Capazes de gerar congestionamento na rede, devido ao grande volume de dados que trafega;
- Não reconhecem se um ataque foi bem sucedido, apenas apontam que um ataque foi iniciado;
- Dificilmente detecta ataques com dados encriptados.

Na Computação em Nuvem, o NIDS pode ser implantado no servidor de Nuvem, interagindo diretamente com a rede externa, porém, ataques quem venham ocorrer dentro da rede virtualizada, provavelmente, não serão detectados. No ambiente de Nuvem, a instalação de NIDS é de responsabilidade do provedor de Nuvem.

***Host-based Intrusion Detection* (HIDS)**: Foram os primeiros tipos de IDS's a serem desenvolvidos e implementados [52]. Seu objetivo é analisar o tráfego sobre um servidor ou uma estação qualquer sem se preocupar com o que está acontecendo

em outros *hosts* da rede. São capazes de detectar situações como intensas falhas de acesso ou violações críticas em arquivos.

HIDS's são capazes de manter o registro de todos os comandos utilizados e ficheiros abertos pelos usuários, além disso, possuem a menor taxa de detecção de falsos-positivos em relação aos NIDS. Entretanto, apresentam inconvenientes como [16] [13]:

- Requer instalação na máquina local que protege, criando uma carga adicional para o sistema;
- A confiança nas capacidades de auditoria e *logging* recaem somente na máquina o qual está instalado.

Em Infraestruturas de Nuvem, HIDS pode ser implementado tanto na VM como no *Hypervisor*. Caso seja instalado na VM, o processo de monitoração é de inteira responsabilidade do usuário do serviço. Por outro lado, caso seja no *Hypervisor*, a responsabilidade passa a ser do provedor de Nuvem [36, 52].

***Distributed Intrusion Detection System* (DIDS)**: Um IDS distribuído é composto por vários outros tipos de IDS, por exemplo, (HIDS, NIDS) ao longo de uma estrutura de rede, os quais se comunicam uns com os outro ou com um servidor central. Nessa abordagem, os componentes de detecção de intrusão coletam informações sobre sistema e utilizando um protocolo pré-definido de comunicação, trocam mensagens entre componentes de outros IDS's ou repassam a um analisador central. Geralmente, um analisador central é um componente que agrega e analisa informações de vários IDS's simultaneamente.

Para fins de detecção, neste IDS são utilizadas combinações de técnicas de análise (anomalias e assinaturas) de ataque. Portanto, DIDS's são eficazes tanto em ataques conhecidos como em ataques desconhecidos [13, 36].

Em ambientes de Computação em Nuvem, um DIDS pode ser posicionado na máquina *host* ou na camada de virtualização do ambiente, por exemplo, uma abordagem utilizando agentes cooperativos onde cada agente é posicionado em cada componente da Nuvem [36]. Se qualquer componente da Nuvem detecta tráfego malicioso, ele então, alertas outros componentes. Cada IDS troca mensagens de alerta

com outros IDS, se um novo ataque é detectado, um nova contramedida é adicionada e compartilhada com outros componentes.

***Virtual Machine Introspection Based Intrusion Detection System* (VMI-based IDS)**: Em VMI-based IDS é possível obter uma visão interna sobre todos os processos e *logs* do *host*, porém, esses *hosts* são altamente suscetíveis à ataques de rede. Por outro lado, NIDS são resistentes à ataques de rede, no entando, tem pouca visibilidade sobre o que acontece internamente no *host*. Em sistemas VMI-based IDS, é mantida toda a visibilidade de um HIDS e também é permitido ao usuário monitorar e analisar toda a comunicação entre VMs para *Hypervisor* e VM para VM. Isto é possível, devido a disponibilidade dos dados que trafegam na rede virtual do *Hypervisor* [36,52].

VMI-based IDS é diferente de HIDS tradicional, uma vez que observa diretamente o estado do *hardware* e oferece uma visão mais robusta do sistema. Nesta abordagem, a *interface* do sistema hospedeiro é usada na comunicação do sistema com o *Hypervisor*, o que permite monitorar, controlar e obter informações sobre o estado real da VM. Além disso, é possível isolar o IDS do *host* monitorado e ainda assim, manter excelente visibilidade sobre seu estado [36]. Esta arquitetura, permite ao usuário, quatro opções de localização dos sensores de monitoramento segundo [52]:

1. Na máquina virtual
2. No *Hypervisor* ou *host* do sistema;
3. Na rede virtual mantida pelo *Hypervisor*;
4. Na rede externa do ambiente virtualizado;

A Figura 2.6 ilustra o posicionamento de um VMI-based IDS na camada de virtualização de um ambiente de Nuvem. Nela é possível observar a camada em que este tipo de modelo poderá ser implementado.

Híbridos: Algumas abordagens usam a combinação de dois ou mais tipos de IDS, por exemplo (DIDS e HIDS), para aumentar a capacidade de detecção. De acordo com [5,58], a maioria dos sistemas de detecção incorporam características tanto de IDS's baseados em redes como de IDS's baseados em *host*.

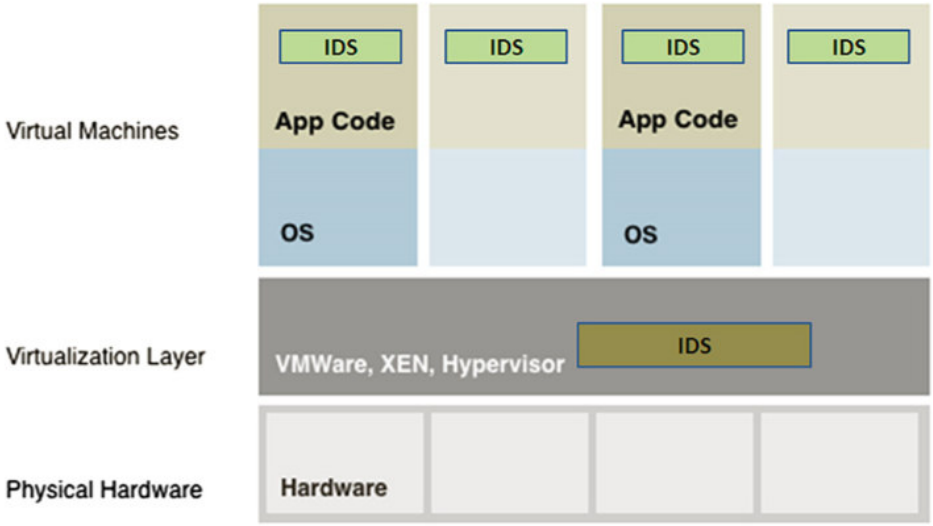


Figura 2.6: Modelo de VMI-based IDS. Adaptado de [36]

Quanto aos Métodos de Detecção

Detecão por Assinaturas: Este método de detecção trabalha procurando regras pré-estabelecidas no tráfego da rede. Quando é encontrado algum código na rede que esteja descrito em alguma regra, por exemplo (o endereço de origem e destino de uma mensagem), é gerado um alerta ou evento que permita uma ação defensiva [58]. Como resultado, os sistemas baseados em assinaturas são capazes de atingir altos níveis de precisão e números mínimos de falsos positivos na identificação de invasões.

Entretanto, ataques desconhecidos podem afetar o desempenho deste método caso a base de assinaturas usadas para as comparações não esteja atualizada com as novas regras. Segundo [52], uma das razões motivadoras para usar detecção baseada em assinatura é a facilidade na manutenção e atualização de novas assinaturas.

Em Nuvem, IDS's baseados em assinatura podem ser usados para detectar ataques conhecidos. Para [36, 52], ele pode ser usado tanto no *Front-End* de Nuvem, detectando invasões externas como no *Back-End*, detectando intrusões externas e internas.

Detecção por Anomalias: Este método pode ser implementado na intenção de observar as mudanças de uso em relação ao padrão normal do sistema (mudanças no padrão de utilização e comportamento dos usuários) [58]. Uma grande variedade

de técnicas, incluindo a mineração de dados, modelagem estatística e modelos de *Markov*, têm sido explorados como diferentes maneiras de abordar problemas de detecção por anomalia.

IDS baseado em anomalia envolve a captura dos dados relativos ao comportamento dos usuários legítimos do sistema durante um período de tempo e, em seguida, são aplicados testes estatísticos para o comportamento atual, o que determina se esse comportamento é legítimo ou não. A vantagem é detectar ataques que não tenham sido encontrados anteriormente.

Técnicas de detecção de anomalias podem ser usadas na Nuvem para detectar ataques desconhecidos em diferentes camadas. Porém, neste ambiente ocorrem muitos eventos simultâneos, o que poderá tornar difícil o monitoraramento [52].

A Tabela 2.2 resume as principais características de IDS para ambientes de Nuvem.

2.4.2 Sistemas de Detecção de Intrusão Baseado em Multiagentes

IDS's atuais sofrem de muitas deficiências que limitam sua adoção para um ambiente de Nuvem. À medida que a *Internet* constitui o requisito básico para estes ambientes, as tecnologias existentes podem ser inseridas nesse contexto com o propósito de encontrar soluções viáveis as ameaças de segurança [65]. Agentes e Sistemas Multiagentes (SMA), representam uma nova geração de sistemas de computação e constituem uma das mais recentes tecnologias de desenvolvimento de sistemas de detecção [58].

O conceito de agente foi originalmente concebido no campo da Inteligência Artificial (IA), evoluindo posteriormente como uma entidade computacional na área de engenharia de *software*. Do ponto de vista de *software*, ela é vista como uma evolução para superar as limitações inerentes às metodologias orientadas a objeto [23]. Possuem memória e comportamento, mas não são entidades passivas como os objetos. Um agente é um sistema computacional encapsulado que está situado em um ambiente, a fim de alcançar seus objetivos de projeto [38].

Tabela 2.2: Características de IDS's Baseados em Nuvem.

Características de IDS's Baseados em Nuvem. Adaptado de [36]

Tipo de IDS	Características	Limitações e Desvantagens	Posicionamento no Ambiente de Nuvem	Responsabilidade de Auditoria e Monitoramento
HIDS	Não requer *Hardware* extra; Identifica intrusão por meio do monitoramento do sistema de arquivo dos hots;	Necessário ser instalado em cada host (VMs, Hypervisor ou máquina hospedeira); Só consegue monitorar o próprio host;	Em cada VM, Hypervisor ou sistema hospedeiro;	Em VMs: usuários da Nuvem; Em Hypervisor: provedor de Nuvem;
NIDS	Identifica intrusões monitorando o tráfego de Rede; Pode monitorar vários sistemas simultaneamente;	Difícil de detectar intrusões de tráfego criptografado; É eficaz apenas para a detecção de intrusões externas ao host; Dificilmente detecta Intrusões na Rede Virtual;	Na rede externa ou na rede virtual;	Provedor de Nuvem;
DIDS	Utiliza características de ambos NIDS e HIDS, desta forma herda benefícios de ambos;	O analisador central se torna sobrecarregado; Exige muitas trocas de mensagens e elevado poder de processamento;	Na rede externa, no Hospedeiro, Hypervisor ou na VM;	Em VMs: usuários da Nuvem; Para outros casos: Provedor de Nuvem;
VMI-IDS	Permite ao usuário monitorar e analisar a comunicação entre VMs, entre Hypervisor e VM; Baseado na Rede Virtual do Hypervisor;	Novo e difícil Compreensão	No Hypervisor;	Provedor de Nuvem;

No contexto geral da AI, agente racional é qualquer coisa que percebe seu ambiente através de sensores e age sobre esse ambiente através de atuadores [13]. De um modo mais específico, um agente de *software* tem sido definido como um sistema com capacidade de adaptação e equipados com mecanismos que lhe permitam decidir o que fazer (de acordo com seus objetivos) [23]. Um sistema multiagente consiste de um número de agentes, que interagem uns com os outros para completar uma tarefa [24]. A Figura 2.7 apresenta a arquitetura básica de um agente de *software*. Nela é possível perceber os sensores e os atuadores de um agente.

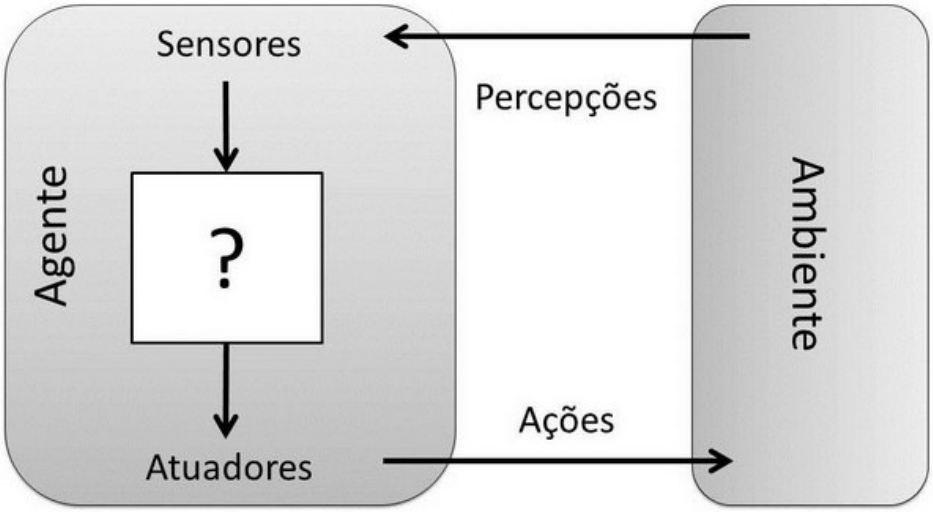


Figura 2.7: Arquitetura de um Agente. Adaptado de [38]

Características

Segundo [58], existem inúmeras vantagens em usar agentes de *software* para desenvolver sistemas de detecção, entre elas pode-se citar:

- Por serem entidades autônomas que trabalham independentemente um dos outros, os agentes podem ser adicionados ou removidos do sistema sem alterar os demais componentes, consequentemente sem reiniciar o sistema;
- Organizando o sistema de forma hierárquica (com múltiplas camadas de agentes que reduzem dados e repassam esses dados para a camada superior) é possível tornar o sistema escalável;
- A habilidade de parar ou iniciar um agente independentemente dos outros agentes no sistema que está sendo monitorado, adiciona a possibilidade de reconfiguração do IDS sem ter que reiniciá-lo. Se um agente com uma nova

funcionalidade precisar ser incluído no sistema ou uma nova funcionalidade em um agente precisar ser inserida, o sistema como um todo não necessita ser reinicializado;

- Devido ao fato de um agente ser programado arbitrariamente, ele pode obter dados de diferentes origens (logs, pacotes de rede, ou outras);
- Como agentes podem ser parados e reiniciados sem afetar o restante do sistema, então podem ser atualizados para novas versões, contanto que sua *interface* externa não seja alterada;
- Se um agente é implementado como um processo separado no *host*, cada agente pode ser implementado em uma linguagem de programação que melhor servir para a tarefa que ele deve executar;

2.4.3 Development Framework (JADE)

JADE é um *Framework* desenvolvido pela *Telecom Italia Lab* (TILAB) muito utilizado para o desenvolvimento de aplicações distribuídas baseadas em sistemas Multiagentes e arquitetura de comunicação *peer-to-peer* conforme as especificações FIPA [13, 55, 56]. Do ponto de vista funcional, JADE fornece todos os serviços básicos necessários para a comunicação entre as aplicações. Ele permite que cada agente descubra dinamicamente outros agentes e se comunique utilizando um mecanismo de passagem de mensagem assíncrona universalmente aceito para sistemas distribuídos. Ademais, o JADE abstrai ao programador muito das especificações da FIPA [13], tais como:

- Não há necessidade de implementar a plataforma de agentes;
- Não há necessidade de implementar um gerenciador de agentes;
- Não há necessidade de implementar transporte de mensagens e *parsing* (análise gramatical das mensagens);
- Protocolos de interação só podem ser herdados por meio de métodos específicos.

Uma característica importante da plataforma JADE é que ela pode ser distribuída por vários *hosts*, cada um deles executando apenas uma *Java Virtual*

Machine (JVM). Os agentes são implementados como *Threads* Java e inseridos dentro de repositórios de agentes chamados de *containers*. Estes *containers* provêm todo o suporte para a execução do agente e representam o ambiente de execução das aplicações de agentes [13].

A Figura 2.8 apresenta a estrutura de um agente simples em JADE. O agente é composto da classe principal *Agent* e de uma classe *Behaviour*. A primeira provê todas as características necessárias para realizar as interações básicas com a plataforma, tais como métodos para registro, configuração e gerenciamento remoto do agente. A classe *Behaviour* é uma classe usada para modelar uma tarefa genérica do agente, ou seja, seu comportamento. O método *action* descreve as ações que são executadas pelos comportamentos. O método *action* é uma das funcionalidades mais importantes no desenvolvimento de sistemas Multiagentes, pois é nele que vão ser definidos os comportamentos que esses agentes terão no ambiente [58].

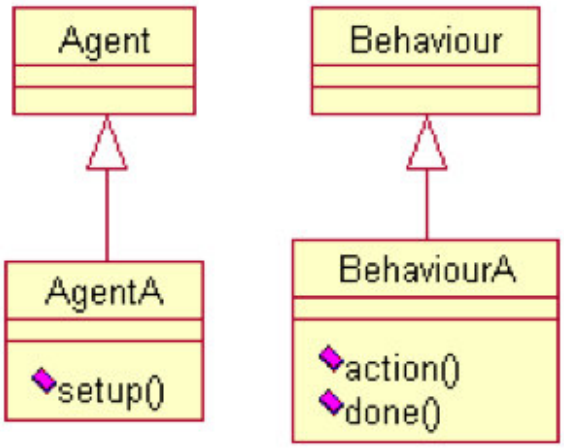


Figura 2.8: Estrutura Básica de um Agente JADE. Adaptado de [58]

2.5 Conclusões

Neste capítulo foram apresentados os conceitos e as características da Computação em Nuvem, bem como as questões de segurança relacionadas a esse modelo de computação. Além disso, foram vistos conceitos relevantes sobre virtualização e a infraestrutura básica para o oferecimento dos serviços de nuvem na camada IaaS.

Foram discutidas soluções de segurança para ataques de negação de serviço. Para isto, abordou-se conceitos e técnicas relacionadas ao uso de sistemas de detecção de intrusão e criptografia. Discutem-se razões para o emprego da tecnologia de agentes aplicada na detecção de intrusos, mostrando-se as suas vantagens sobre os sistemas monolíticos, bem como, a ferramenta de desenvolvimento de sistemas Multiagentes JADE.

De fato, a abordagem de agentes é desejável, pois fornece robustez, confiabilidade e flexibilidade a um IDS, promovendo disponibilidade, execução contínua, tolerância a falhas, sobre carga mínima e elevado nível de escalabilidade e configuração, permitindo a inserção e remoção de agentes de acordo com o cenário de atuação e descoberta de novas tecnologias e conhecimentos de prevenção e proteção de sistemas computacionais.

3 Estado da Arte

Neste Capítulo serão apresentadas algumas das pesquisas mais relevantes sobre sistemas de detecção de intrusão para Computação em Nuvem.

3.1 EICIDS

O projeto *Elastic and Internal Cloud-based Intrusion Detection System* (EICIDS) proposto por [27], consiste em um sistema de detecção de intrusão para ambientes de Nuvem e emprega o conceito de módulos de *software*. O objetivo do EICIDS é fornecer segurança as VM's que compõem a Infraestrutura de Nuvem. Para isto, o IDS utiliza um componente que captura o tráfego diretamente no *Switch* Virtual do *Hypervisor* da Nuvem.

Segundo [27], o EICIDS utiliza técnicas de detecção de tráfego malicioso com base em assinaturas de ataques conhecidos e comportamento anômalo de usuários internos. Outra característica importante a ser destacada, é a arquitetura centralizada empregada pelo projeto. As Figuras 3.1 e 3.2 apresentam as características de cada componente do EICIDS, em seguida serão descritas as responsabilidades de cada um.

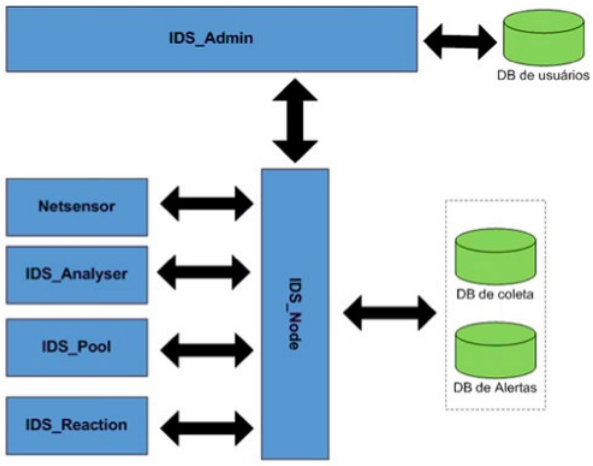


Figura 3.1: Arquitetura Padrão EICIDS. Adaptado de [27]

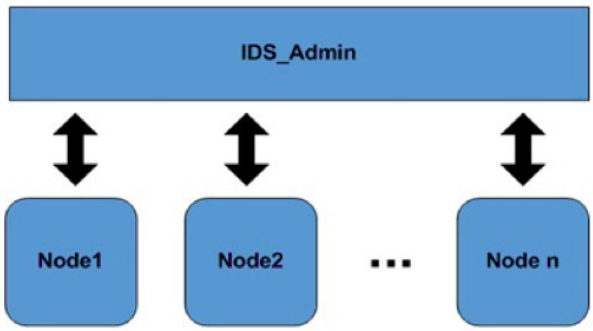


Figura 3.2: Arquitetura EICIDS com Múltiplos Nós. Adaptado de [27]

- *IDS_Admin*: Componente responsável por gerenciar o processo de detecção de intrusão por meio da integração de todos os componentes do IDS, pelo controle dos componentes e também pela organização, sumarização e apresentação dos alertas de segurança emitidos pelos vários componentes *Node* do sistema de Nuvem;

- *IDS_Node*: Tem a função de instanciar e controlar os sensores e analisadores localizados nas máquinas virtuais. É responsável por receber e repassar os alertas de segurança provenientes da análise de assinatura;

- *IDS_Pool*: Responsável por monitorar o ambiente virtual da Nuvem buscando por máquinas virtuais ativas. É o componente responsável por gerenciar a elasticidade da ferramenta;

- *Netsensor*: Tem a função de capturar os pacotes que trafegam na rede virtual do *Hypervisor*. O sensor interage com a rede de forma passiva, atuando como um *sniffer* de rede;

- *IDS_Reaction*: Componente responsável por executar contramedidas aos alertas emitidos pelo IDS_Admin;

- *IDS_Analyzer*: Responsável por analisar os dados capturados pelo *Netsensor*;

3.2 GCCIDS

Em [67], é proposto o *Grid and Cloud Computing Intrusion Detection System* (GCCIDS), um sistema de detecção de intrusão distribuído para ambientes de computação em Nuvem e *grid*, baseado em análise de comportamento (utilizando técnicas de redes neurais artificiais do tipo *feedforward*) e assinaturas de ataques conhecidos.

O objetivo da proposta é prover cooperação entre os serviços fornecidos por cada componente do sistema de detecção e, desta forma, identificar eventos locais (ações de tráfego de rede, configurações e arquivos de *logs* dos *hosts*), que possam representar violações de segurança a si e a outros nós da Nuvem. Cada componente é distribuído uniformemente no ambiente e compartilha informações com componentes vizinhos utilizando uma camada de *Middleware*, ou seja, a comunicação entre os componentes é independente da plataforma empregada.

A Figura 3.3 apresenta o compartilhamento de informações entre os serviços fornecidos pelo sistema de detecção e outros elementos que participam da arquitetura.

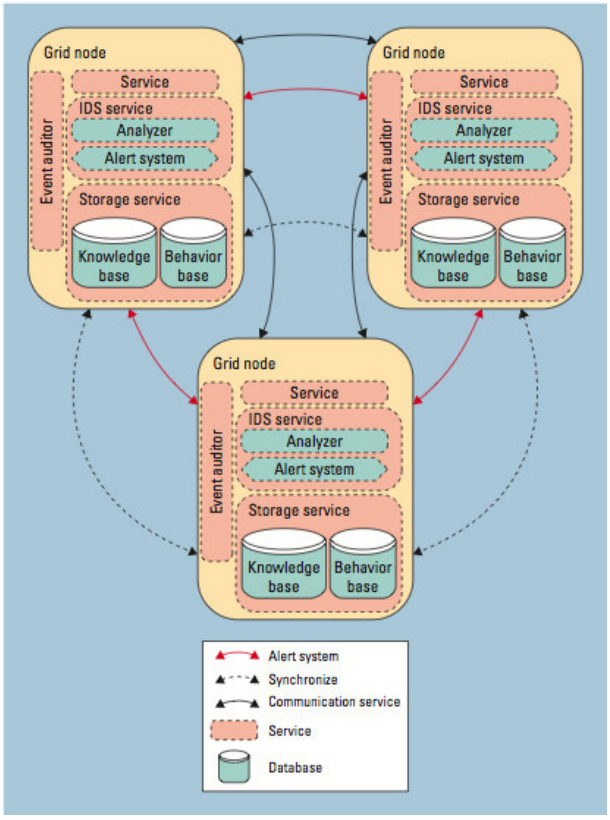


Figura 3.3: Arquitetura do GCCIDS. [67]

- *Node*: Contém os recursos que são acessados homogeneamente através do *Middleware*;
- *Service*: Responsável integrar as funcionalidades dos componentes e facilitar a comunicação através do *Middleware*;
- *Event Auditor*: Responsável pela captura de dados de várias fontes, tais como, *logs*, serviços e mensagens dos nós;
- *Storage Service*: Armazena os dados que o sistema deve analisar. Desta forma, garante que todos os nós tenham acesso aos mesmos dados;
- *IDS Service*: Responsável por analisar os dados provenientes do *Storage Service*, aplicando técnicas de detecção com base no comportamento do usuário e no conhecimento de ataques anteriores;

3.3 MA-IDPS

A pesquisa desenvolvida por [65], propõe um Sistema de Detecção e Prevenção de Intrusão Multiagentes para ambientes de Nuvem utilizando metodologias de análise baseadas em anomalias de comportamento e assinaturas de ataques conhecidos *Multi-Agent Intrusion Detection and Prevention System* (MA-IDPS). O objetivo é lidar com ataques do tipo:

- Ataques Mascarados: Onde as ameaças são representadas por usuários legítimos da Nuvem;
- Ataques Baseados em *Host*: Estes podem ser consequência do ataque anterior e geralmente resultam em um comportamento anômalo por parte do *host* comprometido;
- Ataques Baseados em Rede: Caracterizado por invasores que tentam penetrar na rede;

O MA-IDPs é constituído por agentes de *software* que compõem um ambiente de execução do tipo *Agente Runtime Environment* (ARE) (Componente do IDPs que irá criar e destruir os agentes). A comunicação entre os agentes é feita por

mensagens *Intrusion Detection Message Exchange Format* (IDMEF) baseada no esquema *eXtensible Markup Language* (XML). A comunicação com o cliente é efetuada com o componente *Client Agent Runtime Environment* (CARE), que atua como um sandbox. A proposta MA-IDPs é apresentada na Figura 3.4, logo em seguida serão descritas suas com suas funcionalidades.

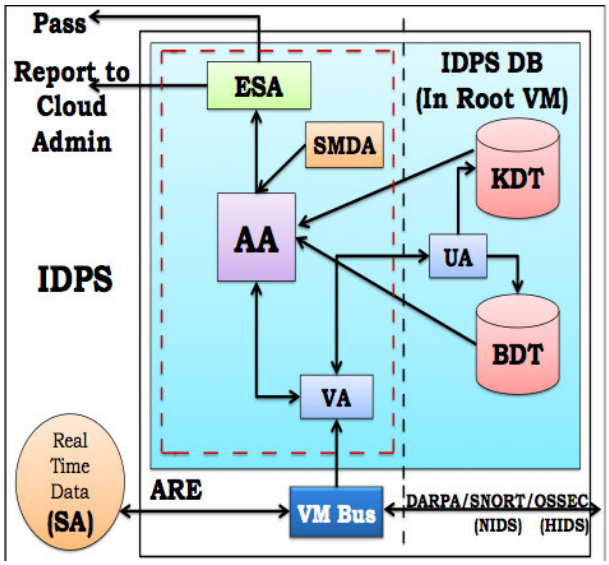


Figura 3.4: Arquitetura do MA-IDPS. [65]

- *Knowledge Data Table* (KDT): Contém dados sobre os padrões de assinatura de *malware*. É atualizado pela UA após detectar comportamento anormal ou nova assinatura por um agente analisador AA usando algoritmo MBM;
- *Behavioral Data Table* (BDT): Contém dados sobre normas e padrões de comportamento aprendidos e atualizados pela UA. É utilizado para detecção de comportamento anormal usando um algoritmo de lógica difusa;
- *Smart Agent* (SA): É um agente inteligente presente no nó do cliente ou em um nó que faz parte da Nuvem. Tem a função de investigar o comportamento do usuário e atividades anormais em cada Nó;
- *Verifier Agent* (VA): Tem a função de verificar a identidade do SA antes dos dados serem enviados para o IDPs. Desta forma, evitar a intrusão de um agente não autorizado;
- *Update Agent* (UA): Responsável por atualizar novas assinaturas de *malware* ou padrões de comportamento no KDT e BDT;

- *Analyzer Agent* (AA): Responsável por analisar os pacotes a procura de ataque ou padrão de comportamento anômalo. Utiliza padrões de correspondência e técnicas de lógica *Fuzzy*;

- *Event Signaling Agent* (ESA): Tem a função de Receber o alerta emitido pelo AA e reportar ao administrador da Nuvem sobre a gravidade do ataque e a contramedida a ser tomada;

- *Smart Malware Detection Agent* (SMDA): É um agente *daemon* que roda em intervalos regulares no sistema que hospeda o IDPs. Tem a função de descobrir qualquer processo ou ação de *malware* atacando o próprio IDPs;

3.4 Secure Cloud Computing Based on Mutual Intrusion Detection System

O modelo de detecção de intrusos proposto por [12], utiliza cooperação mútua entre os módulos que compõem o sistema. O objetivo é diminuir o impacto causado por ataques de DoS em ambientes de Nuvem. Este modelo é baseado na análise de tráfego recebida pelo IDS Snort [1] e três outros componentes: Block, *Mutual* e *Communication*. A detecção da intrusão ocorre com base na análise do comportamento de tráfego da rede e de assinaturas de ataques conhecidos.

Nesta abordagem, um agente localizado em cada região da nuvem é usado para receber mensagens de alerta a partir de outros sistemas de detecção de intrusão. Ao coletar esses informações, o agente julga a confiabilidade do alerta, Após a avaliação, a nova regra de bloqueio é adicionada a tabela *Block* do seu ambiente. A ideia é reduzir o tempo necessário para a comparação da assinatura de um tráfego malicioso já detectado em outros ambientes e, dessa forma, melhorar o desempenho do sistema. A arquitetura do sistema proposto por [12] é apresentada na Figura 3.5, abaixo são descritas as funcionalidades dos componentes.

- *Intrusion Detection System*: Componente usado para coletar pacotes de rede e analisar esses pacotes. É uma adaptação da ferramenta *Snort*;

[1] Snort é um software livre de detecção de intrusão para rede (NIDS), capaz de desenvolver análise de tráfego em tempo real. [12]

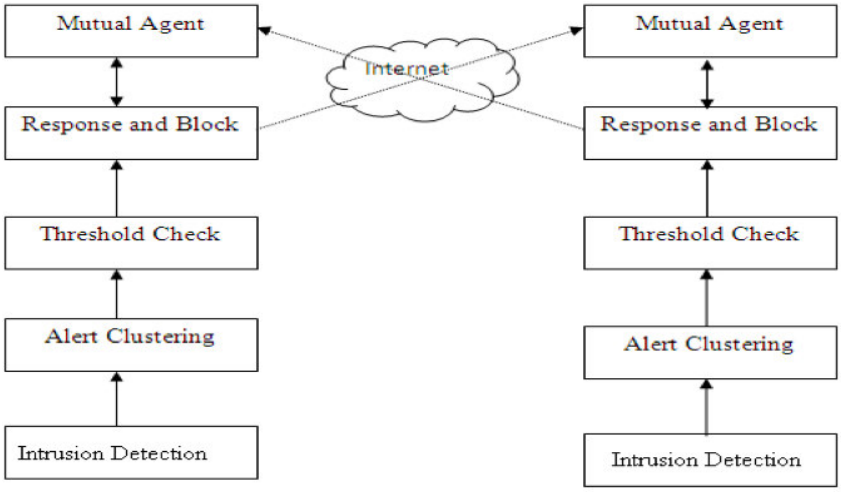


Figura 3.5: Arquitetura do MA-IDPS. [65]

- *Alert Clustering and Threshold Check*: Componente utilizado para identificar o grau de gravidade que o pacote malicioso pode oferecer ao sistema;
- *Response and Block*: A função deste componente é bloquear pacotes defeituosos e enviar uma mensagem de alerta aos outro IDS's.
- *Mutual Operations*: Componente usado para receber mensagens de alerta de outros IDS's. Sua função é julgar a procedência e gravidade do alerta.

3.5 NICE

O modelo de detecção de intrusos em nuvem IaaS proposto por [11], é constituído em modelos analíticos de grafos de ataque e contramedidas baseadas em reconfiguração de redes virtuais. O objetivo é combater tentativas de comprometimento de VM's e, desta forma, evitar que as mesmas sejam utilizadas para realizar ataques distribuídos de negação de serviço. Em geral, o NICE inclui duas fases principais:

- Implementar um agente de detecção de intrusão (NICE-A) em cada servidor da nuvem para capturar e analisar o tráfego da rede. O objetivo é verifica periodicamente as vulnerabilidades que ocorrem dentro do ambiente virtual da Nuvem e, desta forma, estabelecer um *Scenario Attack Graph* (SAGs) para

identificar a gravidade do ataques. Com base nisso, o sistema decidirá se deve ou não colocar a VM comprometida em estado de inspeção de rede.

- Uma vez que a VM entra no estado de inspeção, o *Deep Packet Inspection* (DPI) é aplicado e a reconfiguração da rede virtual é implantada. O objetivo é analisar o comportamento dos possíveis ataques diretamente no *host* comprometido.

A Figura 3.6 apresenta a arquitetura de detecão de intrusos proposta por [11]. O modelo é ilustrado em um único servidor de Nuvem, em seguida são relacionadas as características de cada componente.

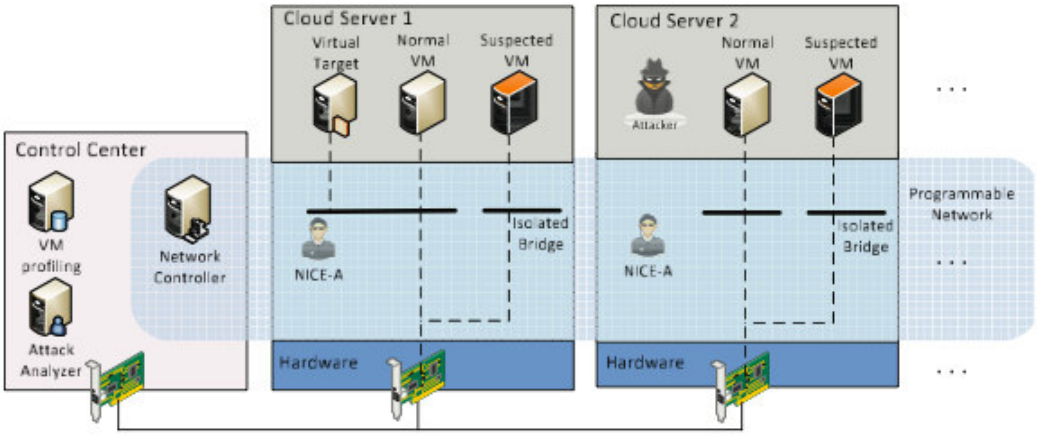


Figura 3.6: Arquitetura do NICE. [11]

- NICE-A: É um NIDS baseado no *Snort* implementado no *Hypervisor* de cada servidor da Nuvem. Sua Função é analisar o tráfego no *Switch* de rede virtual entre as VMs:

- *VM Profiling*: Responsável por obter o estado atual da VM, isto inclui, serviços em execução e conexões de rede;

- *Attack Analyzer*: Componente central do sistema. Sua função é analisar o tráfego malicioso, construir e atualizar o gráfico de ataque, comparar assinaturas e emitir contramedidas.

- *Network Controller*: Componente responsável por realizar a reconfiguração da rede virtual dentro do *Hypervisor* de Nuvem:

3.6 Análise Comparativa dos Trabalhos Relacionados

Na Tabela 3.1, mostrada a seguir, tem-se um quadro-resumo com as principais características e fatores apontados nos projetos vistos nesta abordagem.

Tabela 3.1: Características dos Trabalhos Relacionados relativos à Detecção de Intrusos na Computação em Nuvem.

Características dos Trabalhos Relacionados relativos à Detecção de Intrusos na Computação em Nuvem. Elaboração Própria

Projeto	Baseado em Host	Baseado em Rede	Baseado em Hypervisor e/ou VM	Técnicas de Assinatura	Técnicas de Comportamento	Elasticidade No Ambiente de Nuvem	Validação
3.1	Não	Não	Sim	Sim	Sim	Sim	Não
3.2	Sim	Não	Não	Sim	Sim	Não	Sim
3.3	Sim	Sim	Não	Sim	Não	Não	Não
3.4	Não	Sim	Não	Sim	Não	Não	Não
3.5	Não	Não	Sim	Sim	Sim	Não	Sim

Na Tabela 3.1 observa-se que a maioria das soluções propostas para detecção de intrusos em ambientes de nuvem não oferece contramedidas ativas capazes de minimizar os danos causados por ataques de negação de serviço. Outro fato é que, embora consigam detectar atacantes e invasores em seus diversos níveis de interação, na sua maioria, não oferecem, dentre outras características e habilidades:

- Autônomia;
- Flexibilidade para adicionar novos comportamentos;
- Garantia de Integridade na troca de Mensagens entre os componentes;
- Elasticidade ao ambiente de Nuvem;

Enfim, há uma escassez de soluções que tenham habilidades de prover respostas automáticas e que não fiquem restritas apenas à respostas notificativas. Existe a necessidade de recursos que possibilitem, por exemplo, uma solução de reconfiguração automática do ambiente atacado de modo que o invasor possa ser contido e identificado sem causar danos aos recursos.

3.7 Conclusão

Neste Capítulo foram mostradas algumas pesquisas recentes sobre IDS's para ambientes de Nuvem. Analisando essas pesquisas, constatou-se que a grande maioria não contempla certas características e habilidades exigidas para o ambiente dinâmico e heterogêneo de Infraestrutura de Nuvem, tampouco garantem a integridade dos seus próprios componentes.

Portanto, a continuidade dos projetos e a busca de métodos alternativos de segurança nestes ambientes têm sido uma preocupação constante dos pesquisadores. Nesse contexto, a presente proposta representa mais uma possibilidade de expansão do conhecimento teórico e prático de novas tecnologias aplicadas a essa área. O correspondente detalhamento será apresentado no Capítulo 4.

4 Modelo Proposto

Neste capítulo é proposto um modelo de detecção de intrusão em Infraestrutura de Computação em Nuvem, utilizando-se a tecnologia de sistemas Multiagentes, com o objetivo de explorar as características desejáveis apresentadas por vários IDS's existentes na literatura. Apresentar-se-á a arquitetura e a funcionalidade dos agentes que compõem o modelo, fornecendo uma visão de modo a delimitar e justificar a contribuição deste trabalho.

4.1 Considerações Iniciais

A proposta do modelo de sistema de detecção de intrusão apoia-se em diversas arquiteturas disponíveis como: [11, 12, 27, 65, 67], a fim de buscar melhores soluções para os problemas enfrentados na implementação de um sistema desse porte. Com isto, buscou-se modelar um sistema que abordasse as principais características desejáveis, principalmente quanto à aplicabilidade e respostas aos ataques, visto que muitos sistemas não possuem respostas ativas às tentativas de intrusão.

O modelo proposto prevê a metodologia de detecção por assinaturas de ataques conhecidos de negação de serviço. A escolha se deu em virtude de que a grande maioria dos ataques corriqueiros, são modificações de ataques já conhecidos (assinaturas iguais). Sendo possível, desta forma, detectar variantes de ataques que exploraram as mesmas vulnerabilidades dos sistemas alvo. Para tal, diversos agentes são responsáveis pelas funções de monitoramento, análise de tráfego, detecção e reposta às atividades suspeitas.

No monitoramento, o sistema adota uma combinação de agentes sensores em cada máquina virtual instânciada no ambiente. O objetivo é capturar uma cópia dos pacotes suspeitos tanto na entrada quanto na saída de cada instância virtual disponibilizada pela Nuvem. Após a coleta, os pacotes são repassados aos agentes analisadores que serão responsáveis por verificar o conteúdo de cabeçalho de cada mensagem. Para cada pacote considerado uma ameaça, um alerta é enviado aos seus

superiores para que as contramedidas correspondentes sejam adotadas. Em outros casos os pacotes serão descartados.

Os tipos de reações a serem tomadas podem variar de acordo com a segurança de cada provedor da Nuvem, tendo-se obrigatoriamente, respostas *default* para casos padrões de negação de serviço, podendo-se simplesmente enviar notificações ao administrador do sistema ou reagir de forma mais ativa, alterando-se as configurações de rede dos prováveis atacantes e/ou alterando as regras de *Firewall* do ambiente.

Neste modelo foram adotadas algumas características de cooperação entre agentes. Ataques detectados por agentes locais podem ser compartilhados com agentes externos. Essa abordagem facilita a identificação e neutralização do atacante mesmo que ele não faça parte da mesma infraestrutura virtual. Além disso, é possível detectar ataques em outras infraestruturas antes mesmo que eles aconteçam. A Figura 4.1 apresenta a visão externa do modelo adotado para a detecção de ataques. Nela é possível perceber a interação entre os IDS's na mesma Infraestrutura de Nuvem, porém, em servidores distintos.

As características de autônomia e comunicação entre os agentes são implementadas utilizando a camada de *Middleware* criada pelo JADE. A integridade e autenticidade das mensagens trocadas entre os agentes tanto de borda (Agente de Controle) como internos (Agentes de Análise e Monitoramento) são garantidas utilizando técnicas de criptografia assimétrica e padrões de assinatura digital.

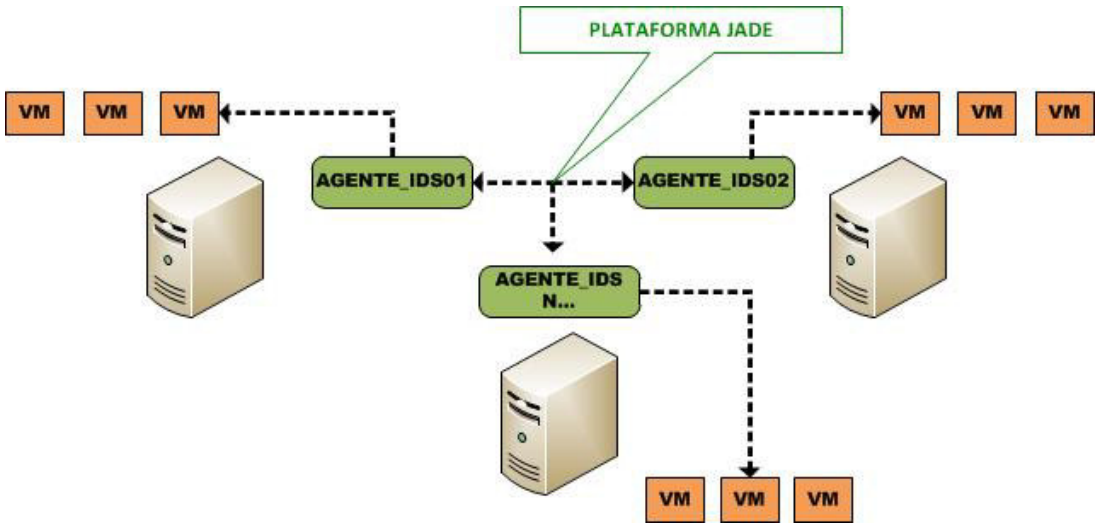


Figura 4.1: Visão Externa do Modelo Proposto. Elaboração Própria

4.2 Arquitetura Geral do Sistema

A arquitetura do sistema proposto é composta pelo ambiente de execução e as camadas de (Monitoramento, Análise e Controle), Armazenamento e Administração do Sistema, ilustrados na Figura 4.2. Essa arquitetura é proposta para cada servidor de virtualização implementado na Nuvem.

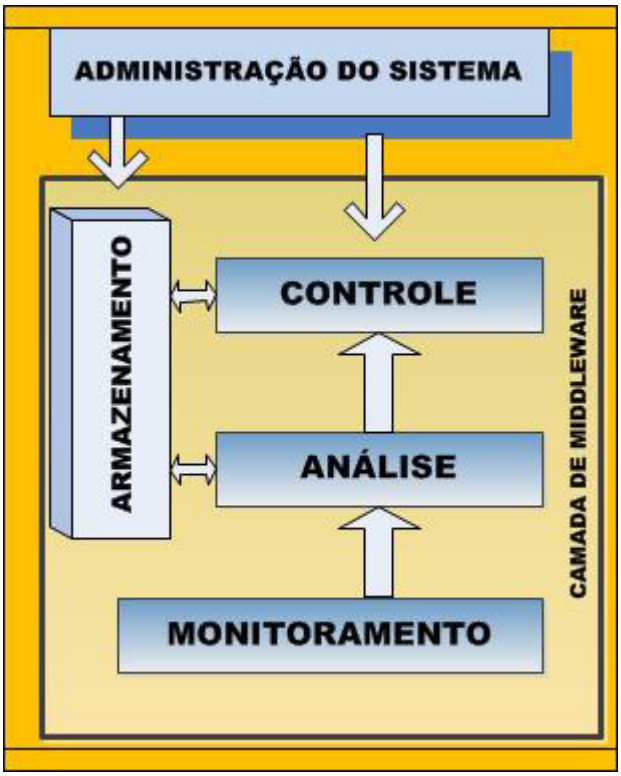


Figura 4.2: Arquitetura Geral do Modelo Proposto. Adaptado de [27, 58]

A seguir, descrevem-se as responsabilidades de cada componente da arquitetura proposta.

Ambiente de Execução O ambiente de execução foi proposto tendo como base as característica físicas e lógicas da camada IaaS de uma nuvem computacional. Nesse contexto, o ambiente é composto por *Hypervisores*, Máquinas Virtuais e uma Rede Virtual. Para simplificar as características de cada ambiente eles serão classificados como **NODE**;

Camada de Monitoramento: Composta por grupo de agentes responsáveis por capturar os pacotes que trafegam dentro do ambiente virtualizado pelo *Hypervisor*. Sua função é coletar o máximo de pacotes de E/S (Entrada e Saída) e filtrá-los por

protocolo (TCP, UDP ou ICMP). Para cada VM em execução haverá um agente de monitoramento.

Camada de Análise: Camada responsável por receber, organizar, formatar e analisar os pacotes coletados pelo camada de monitoramento, de forma que possam ser identificados padrões de ataques. Para cada VM em execução haverá um agente de análise.

Camada de Controle: Principal camada do modelo proposto, é responsável por coordenar todas as atividades que ocorrem no ambiente, incluindo ações de contenção e atualização do sistema. É através desta camada que o administrador gerencia o *status* e a configuração dos agentes.

Camada de Armazenamento: Constitui-se de repositórios de base de dados que são classificados em:

- **Base de Comportamentos**: Repositório responsável por guardar ações de atividades maliciosas para fins de inspeção e alimentação da base de assinaturas;
- **Base de Assinaturas**: Repositório responsável por armazenar assinaturas de ataques conhecidos. Pode ser atualizado dinamicamente baseado nos comportamentos adquiridos no decorrer da execução do sistema ou de acordo com as políticas de cada provedor de Nuvem. Desta forma garantindo que novas técnicas de ataques sejam detectadas;
- **Base de Contramedidas**: Repositório responsável por armazenar as ações que devem ser tomadas de acordo com a severidade do ataque detectado. Varia de acordo com as políticas de cada provedor de Nuvem e deve ser constantemente atualizada para garantir que novas técnicas de ataques sejam neutralizadas;

Administrador do Sistema: Agente humano que interage com o IDS através de uma *Interface* de comunicação com o Agente Controle e a Camada de Armazenamento. O objetivo é realizar diversas tarefas, tais como: Ativar ou desativar agentes, atualizar base de assinaturas, analisar base de comportamentos e atualizar base de contramedidas.

4.3 Funcionamento Genérico do IDS

Para melhor entendimento da arquitetura proposta, é explicado o funcionamento genérico do sistema de detecção de instrusão. Isto é feito com auxílio de diagramas em notação UML e *agent* UML (Indicado para representar requisitos de agentes de Software) [21]. Inicialmente é descrito os principais diagramas que compõem o modelagem do sistema.

4.3.1 Administrador do Sistema

O papel do administrador de segurança é de vital importância para um ambiente de nuvem, visto que o mesmo é responsável por diversas tarefas que exigem um conhecimento técnico elevado para administração de segurança de um provedor de serviços. É ele, o responsável por interagir com o ambiente de gerenciamento e controle do IDS.

As responsabilidades do administrador incluem por exemplo:

- Alimentação da base de dados de assinaturas em caso de descoberta de novos padrões de ataques;
- Alimentação da base de dados das contramedidas. A Flexibilidade da ferramenta proporciona a inserção de novas estratégias de defesa em tempo real;
- Ativar ou desativar o agente controle. É o processo que inicia a busca por VM's ativas dentro do servidor de virtualização;
- Visualizar *Log's* em tempo real;
- Capacidade para inspecionar de forma mais específica uma provável VM suspeita em estado de inspeção.

A Figura 4.3 apresenta o diagrama de casos de uso do administrador de segurança dentro do modelo de detecção de intrusão proposto.

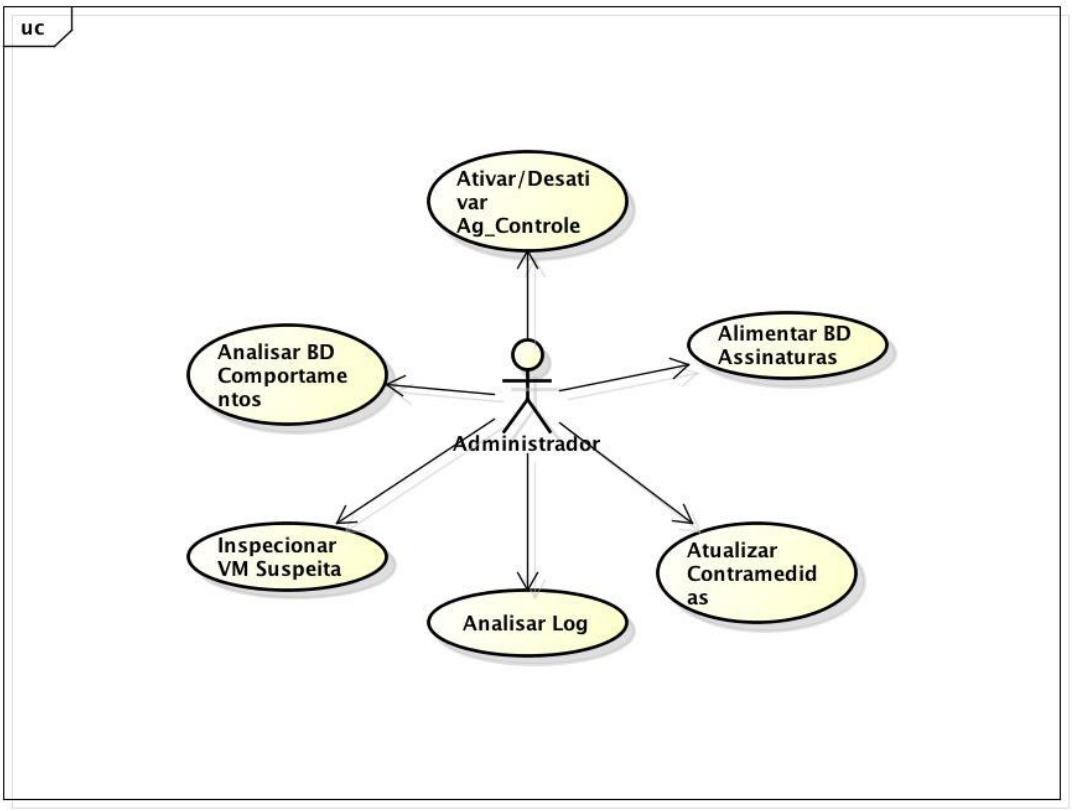


Figura 4.3: Principais Casos de Uso do Administrador do Sistema Proposto. Notação UML. Elaboração Própria

4.3.2 Agente de Controle

Conforme descrito anteriormente, o administrador do sistema é responsável por ativar ou desativar o agente de controle do sistema. Após estar ativado, é responsabilidade do agente controle procurar por VM's que estão em processo de execução no ambiente.

Para cada VM encontrada o agente de controle deverá criar agentes de análise e monitoramento. Para isto, será emitido um scanner de VM por toda a infraestrutura do servidor de virtualização de nuvem em busca de endereços IP's ativos. Após essa busca, é gerada uma lista de endereços de VM's em execução, o qual será utilizada para fins de identificação da origem dos ataques.

O agente controle é responsável também por receber alertas de intrusão emitidos pelo agente analisador e por outros agentes de controle. Para cada recebimento de um alerta é possível definir uma ação a ser tomada, por exemplo:

1. **Ataques Internos ao Servidor**: É possível isolar o atacante em uma rede de inspeção e/ou aplicar regras de *Firewall*;

2. **Ataques Externos ao Servidor**: É possível aplicar regras de Firewall e emitir alertas para outros agentes controles localizados em outros servidores, neste caso, o agente de controle proprietário da VM atacante é que será responsável por manter a VM comprometida em modo de inspeção;

3. **Ataques Externos a Infraestrutura de Nuvem**: Neste caso é possível emitir regras de *Firewall* para conter as ações do atacante;

Na Figura 4.4 é apresentado o diagrama de caso de uso que representa a explosão do agente controle e os demais componentes que interagem com o mesmo. Essa abordagem tem como parâmetro os requisitos adotados pela *agent* UML.

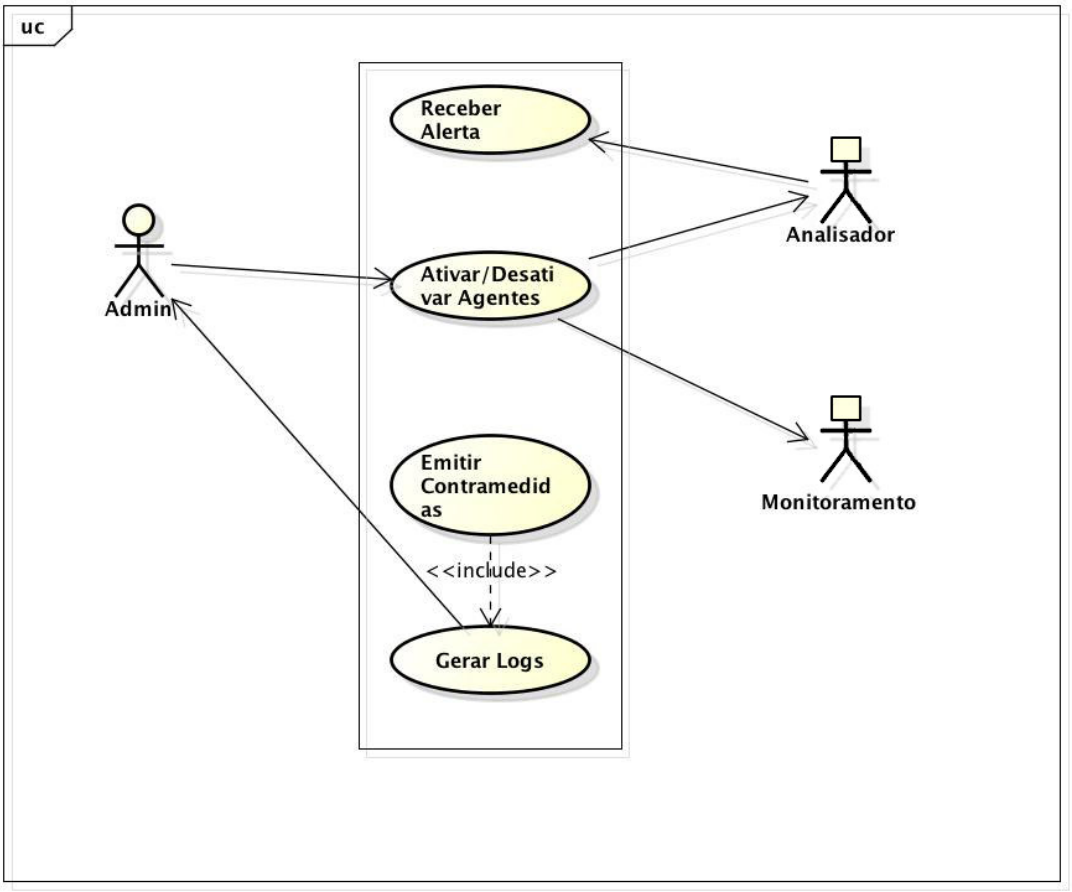


Figura 4.4: Diagrama de Casos de Uso (Explosão dos Agentes Controle) Notação *Agent* UML. Elaboração Própria

A Figura 4.5 ilustra o diagrama de classes do agente controle. As atribuições de cada classe são descritas a seguir:

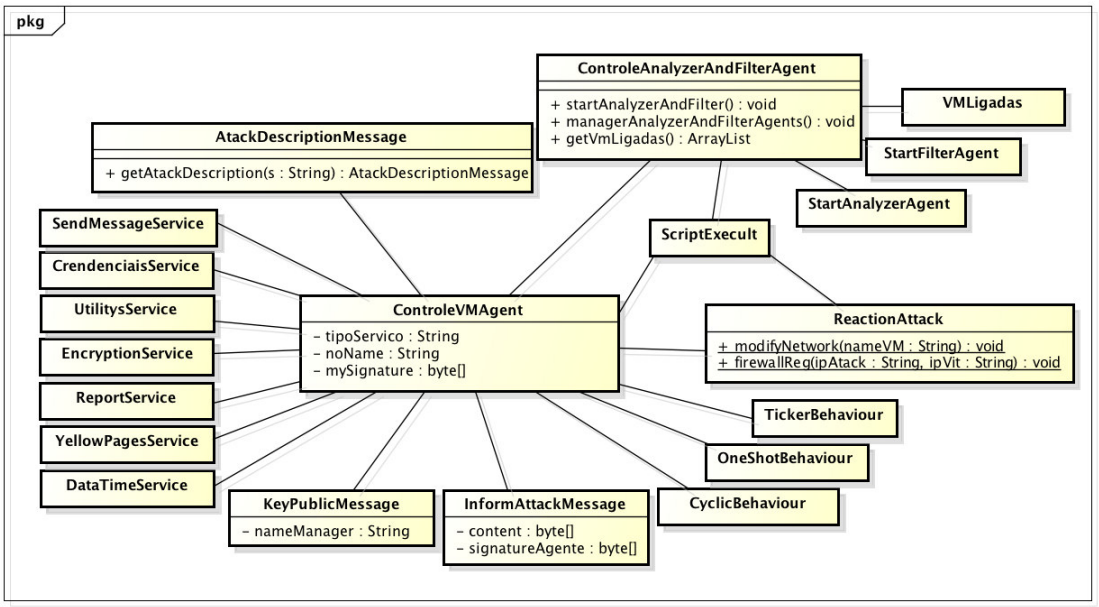


Figura 4.5: Diagrama de Classes do Agente Controle. Elaboração Própria

- ***SendMessageService***: Classe responsável pelo envio de mensagem entre agentes;
- ***EncryptionService***: Responsável por criptografar e desencriptar o conteúdo das mensagens;
- ***CrendenciaisService***: Classe responsável por gerar os pares de chaves e assinatura digital;
- ***UtilitysService***: Responsável por alguns utilitários como resolver *Media Access Control Address* (MAC), Nome de uma VM por IP etc..;
- ***YellowPagesService***: Função de registrar o serviço dos agentes nas páginas amarelas e notificar os demais controladores;
- ***KeyPublicMessage***: Mensagem que contém uma chave pública;
- ***InformAttackMessage***: Mensagem com conteúdo criptografado;
- ***CyclicBehaviour***: Comportamento cíclico de agentes;
- ***OneShotBehaviour***: Comportamentos que ocorre apenas uma única vez;
- ***TickerBehaviour***: Comportamento que ocorre em intervalo de tempos;
- ***AtackDescriptionMessage***: Informa a descrição de um ataque;

- ***ReactionAttack***: Responsável por métodos de contramedidas dos ataques;
- ***ScriptExecult***: Permite executar comandos *shell* e capturar o seu resultado;
- ***ManageAnalyzerAndFilterAgent***: Classe utilizada para ativação e destruição de agentes Analisadores e Monitores;
- ***StartFilterAgent e StartAnalyzerAgent***: *Threads* que devem ser instanciadas para criação de agentes do gênero;
- ***DataTimeService***: Fornece serviço para capturar a hora e data atual;
- ***ReportService***: Realiza a escrita no arquivo de *log*;

4.3.3 Agente Analisador

Ao ser ativado pelo agente controle o agente analisador espera por pacotes enviados pelo agente monitor, após o recebimento, o agente faz a leitura dos pacotes em busca de assinaturas de ataques conhecidos, para isto, a base de assinaturas é consultada a cada comparação.

No momento em que um pacote é identificado como malicioso ou como um comportamento suspeito, o agente analisador monta uma mensagem de alerta que será enviada ao agente de controle, essa mensagem é criptografada e contém as informações de (tipo de ataque, atacante, vítima e comportamento que originou o pacote) necessárias para que o agente controle tome as providências cabíveis. O diagrama de sequência ilustrado na Figura 4.6 apresenta as sequências de ações tomadas pelo agente analisador.

4.3.4 Agente Monitor

O agente monitor funciona com uma espécie de farejador de pacotes localizado no *switch* virtual controlado pelo *Hypervisor*. Ele é vinculado à VM através de suas configurações de rede IP e endereço MAC. A principal função deste agente é capturar os pacotes de E/S específicos para cada VM. Com essa abordagem é possível identificar pacotes maliciosos que tenham como destino a VM ou que se tenham

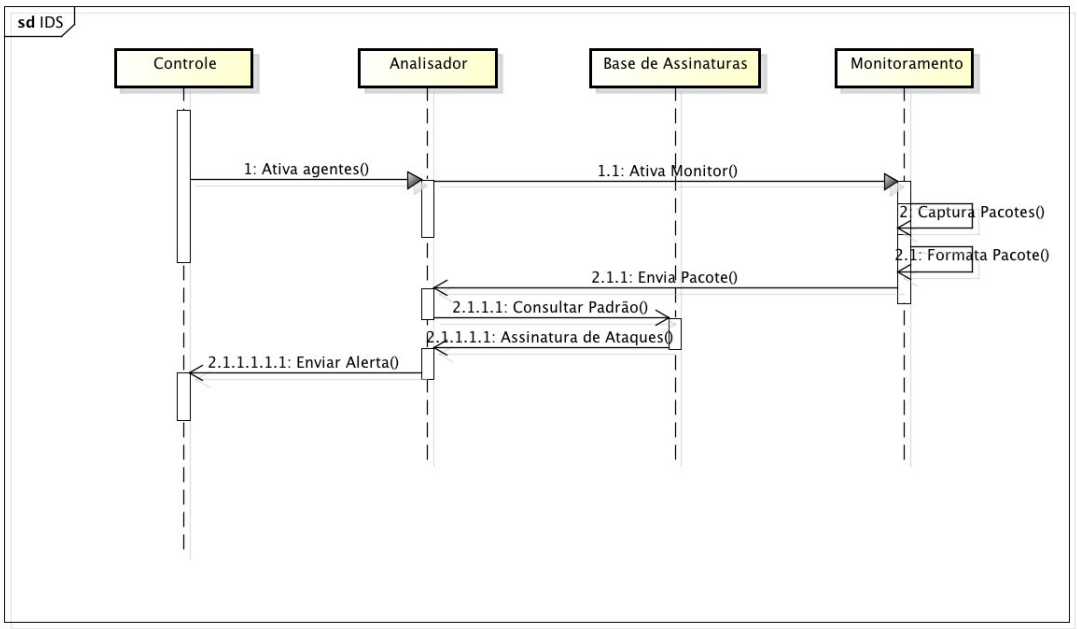


Figura 4.6: Diagrama de Sequência do Agente Controle. Elaboração Própria

originado dela. Além disso, o agente monitor é responsável por formatar os pacotes capturados antes de serem enviados ao analisador. Essa formatação inclui:

- Tipo de Protocolo;
- Endereço de Origem;
- Endereço de Destino;
- Endereço MAC de Origem;
- Endereço MAC de Destino;

O diagrama de sequência do agente de monitoramento é mostrado na Figura 4.7. O cenário do agente inclui também o *switch* virtual implementado pelo *Hypervisor*.

4.3.5 Cooperação entre os Agentes Controle

O diagrama de atividade apresentado na Figura 4.8 mostra o fluxo de operações necessárias para as ações de cooperação entre os IDS's localizados na mesma Infraestrutura de Nuvem, porém, em outros servidores de virtualização. Com

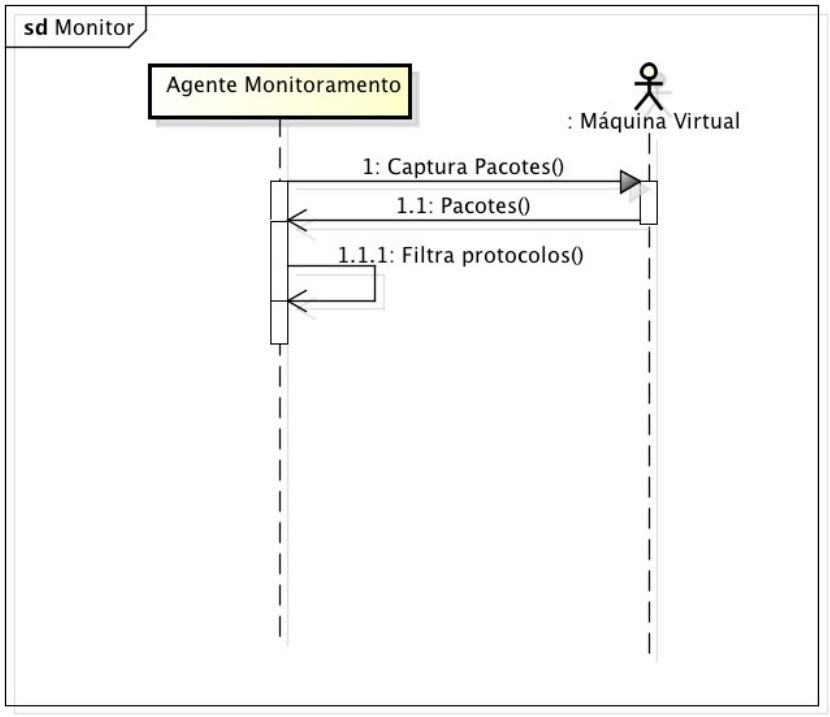


Figura 4.7: Diagrama de Sequência do Agente Monitor. Elaboração Própria

esta abordagem é possível detectar ataques distribuídos de negação de serviço que tenham como alvo tanto serviços internos quanto serviços externos a infraestrutura, por exemplo, é possível evitar que VM's sejam controladas para executar ataques coordenados de negação de serviço contra *sites Web* na *Internet*.

A técnica consiste em detectar o ataque em um determinado servidor e emitir mensagens criptografadas do tipo *broadcast* para todos os outros agentes controladores da Infraestrutra de Nuvem. Esta mensagem inclui:

- Tipo de Ataque;
- Endereço de Origem;
- Endereço de Destino;
- Endereço MAC de Origem;
- Endereço MAC de Destino;

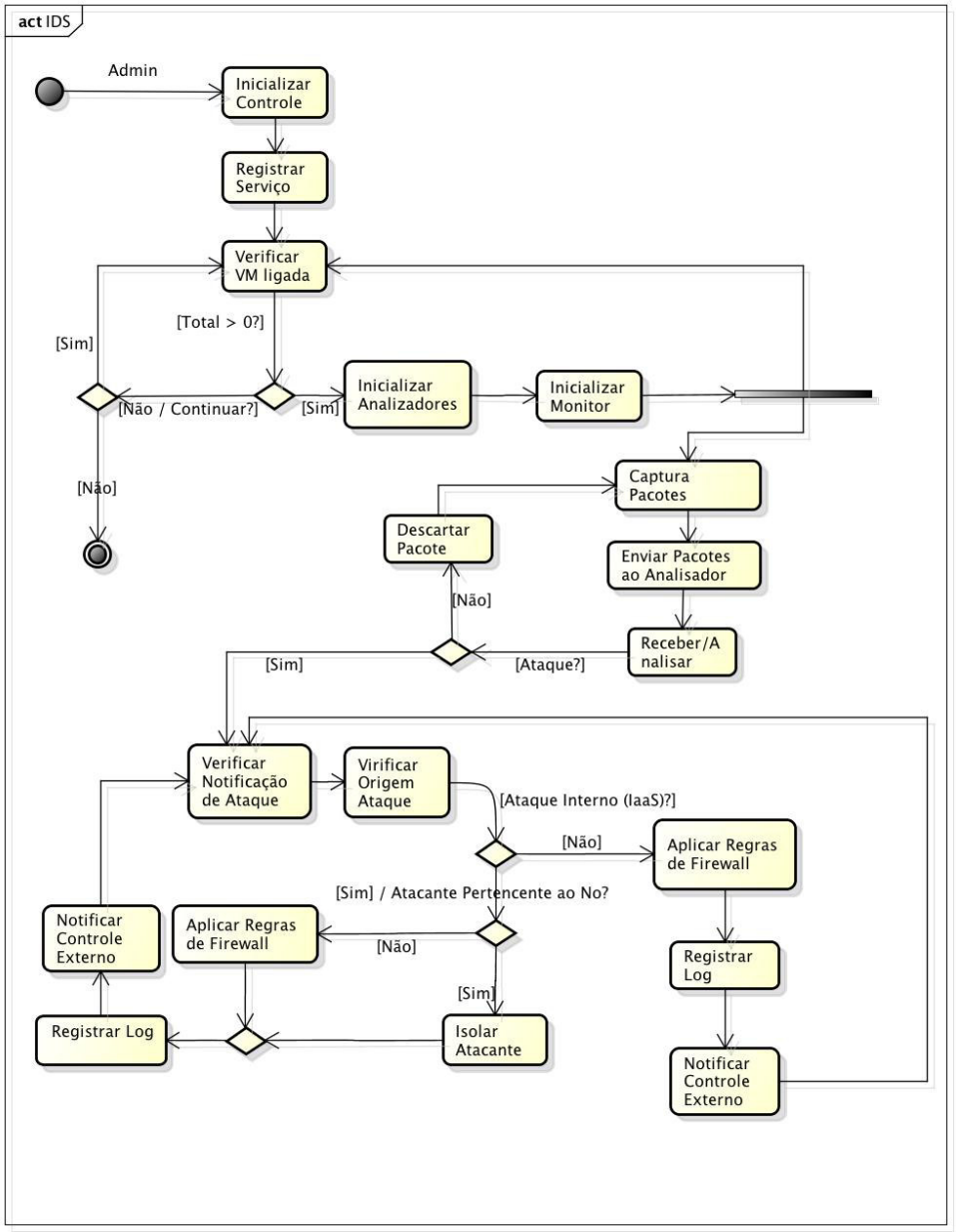


Figura 4.8: Diagrama de Atividade Cooperação entre Agentes Controle. Elaboração Própria

4.3.6 Troca de Mensagens entre os Agentes

Outra característica peculiar do modelo proposto é garantir a integridade das mensagens trocadas entre os agentes que compõem o IDS. Isto é possível, utilizando-se técnicas de criptografia assimétrica e padrões de assinaturas digitais específicos para cada agente do sistema. As sequências para o envio de mensagens criptografadas entre os agentes envolvem as seguintes etapas:

1. Gerar um par de Chaves Assimétrica;

2. Enviar Chave Pública ao destinatário;

3. Assinar Mensagem com Parâmetros de identificação única do emissor (Nome, Dono, ID);

4. Criptografar Mensagem com Chave Pública do Destinatário;

O diagrama de sequência apresentado na 4.9 mostra o processo de troca de chaves públicas entre os agentes controle e o agente analisador. O mesmo processo pode ser considerado para outros agentes.

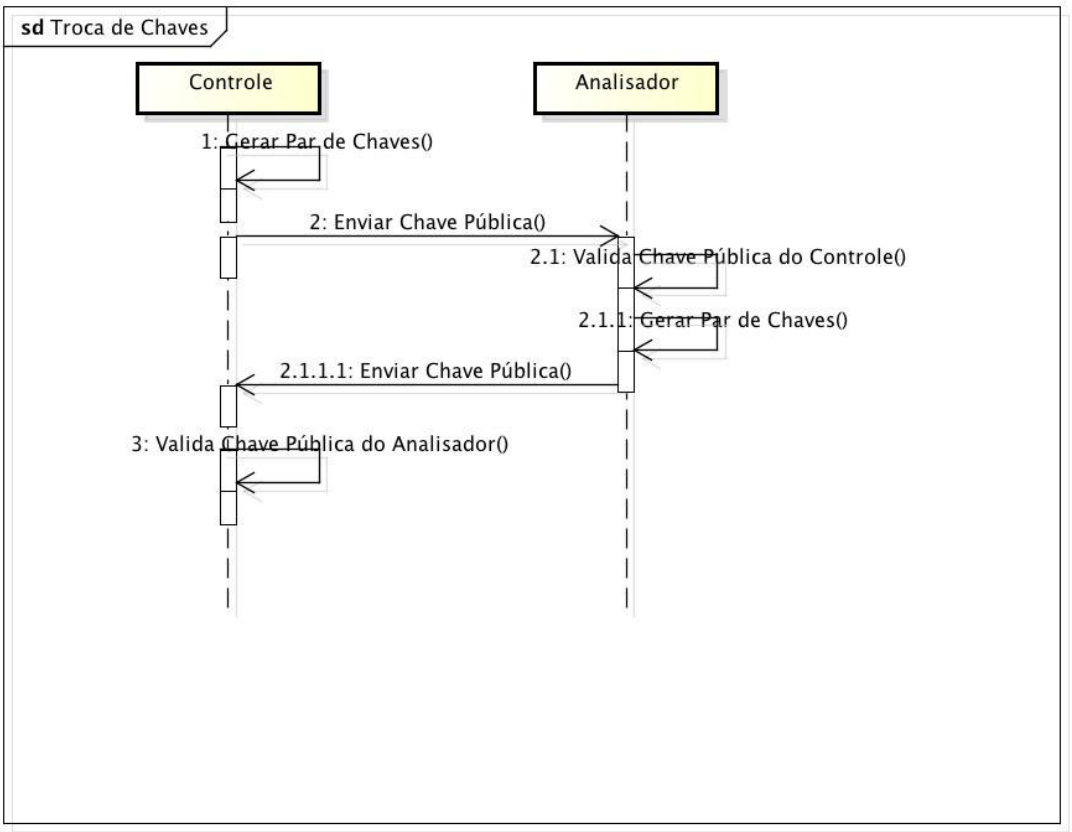


Figura 4.9: Diagrama de Sequência para Compartilhamento de Chaves. Elaboração Própria

4.4 Conclusões

Neste Capítulo foi apresentada uma proposta de um modelo de IDS para Infraestrutura de Nuvem implementada em camadas de agentes de software, destacando-se a função definida para cada agente e demonstrando-se sua capacidade de cooperação e o compartilhamento de mensagens criptografadas.

Para a implementação do protótipo, foi escolhida a plataforma JADE, uma ferramenta para o desenvolvimento de agentes inteiramente desenvolvida em Java.

5 Prototipagem e Resultados Parciais

Neste Capítulo são apresentados as etapas de implementação e prototipagem dos agentes que compõem o modelo proposto, bem como a utilização da plataforma de desenvolvimento JADE, por fim, os resultados parciais obtidos com a utilização da ferramenta desenvolvida.

5.1 Implementação do Protótipo

A implementação do agente controle, responsável por iniciar os demais agentes da ferramentas IDS, é o ponto de partida para a construção do sistemas proposto. Além disso este agente é responsável por disponibilizar uma *Interface* administrativa para o administrador do sistema. Logo em seguida serão apresentadas as implementações dos agentes de análise e monitoramento.

5.1.1 Agente Controle

Ao ser iniciado pelo administrador, o agente controle executa um pedido de endereços ativos direcionado ao endereço de *Broadcast* da rede, como mostra a Figira 5.1. A resposta desta solicitação é escrita em um arquivo de texto contendo as informações de (IP, MAC e Nome) de cada VM ativa. Esse processo de busca por VM é um comportamento temporal durante todo cilco de vida do agente controle.

De acordo com a totalidade de VM's ativa, é iniciado o processo de distribuição de agentes de análise e monitoramento para cada VM ativa. Cada agente é identificado de forma única no ambiente (ID), essa identificação é pertinente durante todo o processo de validação de assinaturas entre os agentes.

Ao iniciar os agentes, o agente controle fica em processo de espera, aguardando mensagens de alerta dos agentes analisadores e/ou agentes controles hospedados em outros servidores. Após o recebimento de um alerta é iniciado o processo de validação que será descrito na seção 5.2.4. Com a resposta do processo

```
public void startAnalyzerAndFilter() throws IOException{

    shell.executarComando(arpScan);
    String resultCommand = shell.getResultado();

    this.vmLigadas = UtilitysService.getIpVms(resultCommand);
    ArrayList<String> vmLigadasMac = UtilitysService.getMACVms(resultCommand);

    if (vmLigadas.size() == 0) {

        System.err.println("Nenhuma Vm está Ligada!!!");


    } else {

        BufferedWriter writer = new BufferedWriter(new FileWriter(file));
        BufferedWriter writerMacIp = new BufferedWriter(new FileWriter(fileMacIp));

        ex = Executors.newCachedThreadPool();

        for (int i = 0; i < vmLigadas.size(); i++) {

            String ipCurrent = vmLigadas.get(i);
            System.err.println("Iniciado Agentes para ---> "+ ipCurrent);

            ex.execute(new StartAnalyzerAgent(ag, vmLigadas.get(i)));
            ex.execute(new StartFilterAgent(vmLigadas.get(i)));

            writer.write(ipCurrent + ";");
            writer.flush();

            writerMacIp.write(vmLigadas.get(i)+";"+vmLigadasMac.get(i)+"\n");
            writerMacIp.flush();

        }

        writer.close();
        writerMacIp.close();

    }

}
```

Figura 5.1: Código de Verificação de VM's Ativas. Elaboração Própria

de validação é possível identificar a origem da mensagem (Analisador ou Controle Externo). O conteúdo extraído da mensagem será enviado como parâmetro para o método *GetNameVmByMAc* da classe UtilityService, apresentado na Figura 5.2, para verificar se o MAC contido na mensagem de alerta pode ser resolvido pela lista de endereços controlados pelo controle local.

Com resultado do método *GetNameVmByMAc* é possível determinar o tipo de contramedida a ser tomada. Sendo assim, para VM's localizadas no mesmo servidor é possível executar o método *Modifynetwork* da base de dados de contramedidas, em outros casos, é emitido uma mensagem de alerta criptografada para outros controles.

A classe *ReportService* executa o processo de escrita no arquivo de *log*.

```
adm.setMacBlocked(ipMac[1]);
adm.setNameVm(UtilitysService.getNameVmByMac(ipMac[1]));
adm.setIpMasked(ipMac[0]);
adm.setIpOrign(UtilitysService.getIpVmByName(UtilitysService.getNameVmByMac(ipMac[1])));

if(!(adm.getNameVm().equals("false"))){

    ReactionAttack.modifyNetwork(adm.getNameVm());
    adm.setReaction("Modificacao da Rede");
    meAllManager.setId(Identifier.TCP_SYN_FLOOD_EXTERNAL);

}else{

    String ipVit = adm.getAnalyzerName().split("_")[1];
    ReactionAttack.firewallReg(adm.getIpMasked(), ipVit);
    adm.setReaction("Regra de Firewall");
    meAllManager.setId(Identifier.TCP_SYN_FLOOD_EXTERNAL_SUSPECT_NOT_FOUND);

}

ReportService.writeDescricao(adm);


meAllManager.setPuKey(myPublicKey);
meAllManager.setSignatureAgente(mySignature);
meAllManager.setContent(EncryptionService.encryptMesage(adm.toString().getBytes(), myPrivateKey));
```

Figura 5.2: Código de tratamento para mensagens recebidas. Elaboração Própria

5.1.2 Agente de Análise

Como descrito anteriormente, o agente de análise é responsável por avaliar os pacotes filtrados pelo agente de monitoramento. Sendo assim, logo após ser iniciado e autenticado pelo agente de controle, o agente de análise inicia o processo de recebimento de pacotes enviados pelo agente de monitoramento, logo em seguida, os pacotes são agrupados e organizados por sequência de protocolo.

Para cada agrupamento de pacotes recebidos é executado um processo de verificação de assinatura. O processo consiste em extrair o conteúdo existente no cabeçalho de cada pacote e compará-los com uma uma assinatura específica de ataque. Na Figura 5.3 é mostrado o processo de comparação de assinaturas para ataques do tipo *UDP Flood*. O método *VerifyFlood* da classe *UDPAttackSignature* é responsável por receber é e executar o processo de comparação de ataque. O resultado do processo de comparação pode ser desde um descarte dos pacotes como o envio de uma mensagem de alerta criptografada ao agente de controle.

```
public class UDPAtackSignature {

    public ArrayList<String> verifyFlood(ArrayList<UDPPacket> pacotesUdp){

        ArrayList<String> ipMacs = new ArrayList<>();
        Set<byte[]> macsEncontrados = new HashSet<>();
        ArrayList<IpAndMACAddress> ipsAndMac = new ArrayList<>();
        Set<String> macAtacantes = new HashSet<>();
        Set<String> macs = new HashSet<>();


        UDPPacket p;
        InetAddress ip;
        byte[] mac;

        IpAndMACAddress ipMac;

        for(int i = 0; i < pacotesUdp.size(); i++){

            ipMac = new IpAndMACAddress();

            p = pacotesUdp.get(i);

            ip = p.src_ip;
            EthernetPacket e = ((EthernetPacket) p.datalink);
            mac = e.src_mac;

            ipMac.setIp(ip.toString());
            ipMac.setMac(UtilitysService.hex2String(mac));
            ipsAndMac.add(ipMac);

        }


        for(int i = 0; i < ipsAndMac.size() - 1; i++){

            IpAndMACAddress ipmac2 = ipsAndMac.get(i);

            for(int j = (i+1); j < ipsAndMac.size(); j++){

                if(ipmac2.getMac().equals(ipsAndMac.get(j).getMac())){
                    macAtacantes.add(ipmac2.getMac());
                    break;
                }

            }

        }

        Iterator<String> it = macAtacantes.iterator();
```

Figura 5.3: Trecho do Código do Agente de Análise. Elaboração Própria

5.1.3 Agente de Monitoramento

Iniciado juntamente com o agente de análise, o agente de monitoramento é responsável por capturar o tráfego de E/S para uma VM específica dentro da Infraestrutura de Nuvem. A Figura 5.4 apresenta um trecho do código de captura dos pacotes. Inicialmente, o agente de monitoramento recebe do agente controle o endereço de IP da VM que irá ser monitorada, esse paramêtro é repassado ao método

Action e o resultado da captura armazenado na variável *PacketCaptured*. O processo de formatação dos pacotes é executado durante a captura pelo método *Captor.SetFilter*.

Após a captura e formatação, os pacotes são enviados ao agente de análise específico da VM monitorada. Este processo é executado pelo método *Send.MessageService*.

```
public class FilterAgent extends Agent{

    private String idInterface;
    private String ipVM;
    private PacketMessage pm;
    private final NetworkInterface[] interfaces = JpcapCaptor.getDeviceList();

    @Override
    protected void setup() {

        Object[] args = getArguments();

        ipVM = (String) args[0];
        idInterface = (String) args[1];

        addBehaviour(new CyclicBehaviour() {

            @Override
            public void action() {

                try{

                    JpcapCaptor captor = JpcapCaptor.openDevice
                            (interfaces[Integer.parseInt(idInterface)], 68524, true, 1000);

                    captor.setFilter("tcp or udp and dst host "+ipVM, true);

                    Packet packetCaptured = null;

                    for(int i = 0; i < 1000; i++){

                        packetCaptured = captor.getPacket();

                        pm = new PacketMessage();
                        pm.setId(Identifier.PACKET_CAPTURED);
                        pm.setPacket(packetCaptured);

                        SendMessageService.sendMensagem(myAgent, "analyzer_"+ipVM, pm);

                    }

                }catch(Exception ex){
                    ex.printStackTrace();
                }

            }

        });
```

Figura 5.4: Trecho do Código do Agente de Monitoramento. Elaboração Própria

5.1.4 Processo de Validação das Mensagens

Para garantir a integridade das informações geradas pelo tráfego, foi implementado um processo de autenticação e validação nas mensagens trocadas entre

os agentes. O processo consiste em criptografar e assinar a mensagem antes que ela seja enviada para outro agente. Desta forma, agentes infiltrados no sistema não poderão se comunicar com outros agentes até que o processo de validação seja executado.

O trecho de código apresentado na Figura 5.5 mostra o processo de criação do par de chaves assimétricas do agente de análise. As etapas para a criação das chaves e da assinatura digital são descritos a seguir:

1. Executar o método *GenerateMypairKey* da classe *CrendenciaisService* para gerar o par chaves *MyprivateKey* e *MypublicKey*;
2. Utilizar a chave *MyprivateKey* e o identificador agente (ID) para gerar a assinatura digital;
3. Receber a chave *MypublicKey* do agente Controle;
4. Enviar ao agente controle uma mensagem de validação contendo (a chave *MypublicKey* e a assinatura digital do agente analisador);

```
try {
    mafa.startAnalyzerAndFilter();

    KeyPair kp = CrendenciaisService.generateMyPairKey();

    myPrivateKey = kp.getPrivate();
    myPublicKey = kp.getPublic();

    mySignature = CrendenciaisService.generatyMySignature(myAgent.getLocalName(), myPrivateKey);
```

Figura 5.5: Trecho do Código de Geração de Chaves Assimétricas. Elaboração Própria

5.2 Testes e Resultados Parciais

Nesta seção serão apresentados os resultados das simulações realizadas com o protótipo de agentes, dando ênfase as ações de contramedidas, neutralização e disponibilidade do ambiente vitimado. As simulações de ataques foram realizadas com base nas características dos seguintes ataques de negação de serviço:

- TCP SYN *Flood*

- UDP *Flood*
- ICMP *Flood*
- ICMP *Smurf*

5.2.1 Cenário de Testes

Para a realização dos testes foi construído um ambiente típico de Infraestrutura de Nuvem. O ambiente proposto é composto por dois computadores que representam os ambientes de virtualização e são denominados CLOUDNODE01 e CLOUDNODE02 respectivamente. Para efeito de administração da Infraestrutura de Nuvem e consequentemente da ferramenta proposta, foi implementado um ambiente de administração denominado CLOUDFRONTEND. Esses computadores foram interligados por uma rede local do tipo TCP/IP baseada nas características físicas e lógicas da topologia estrela, utilizada em ambientes virtualizados. As características do cenário proposto podem ser observadas na Figura 5.6.

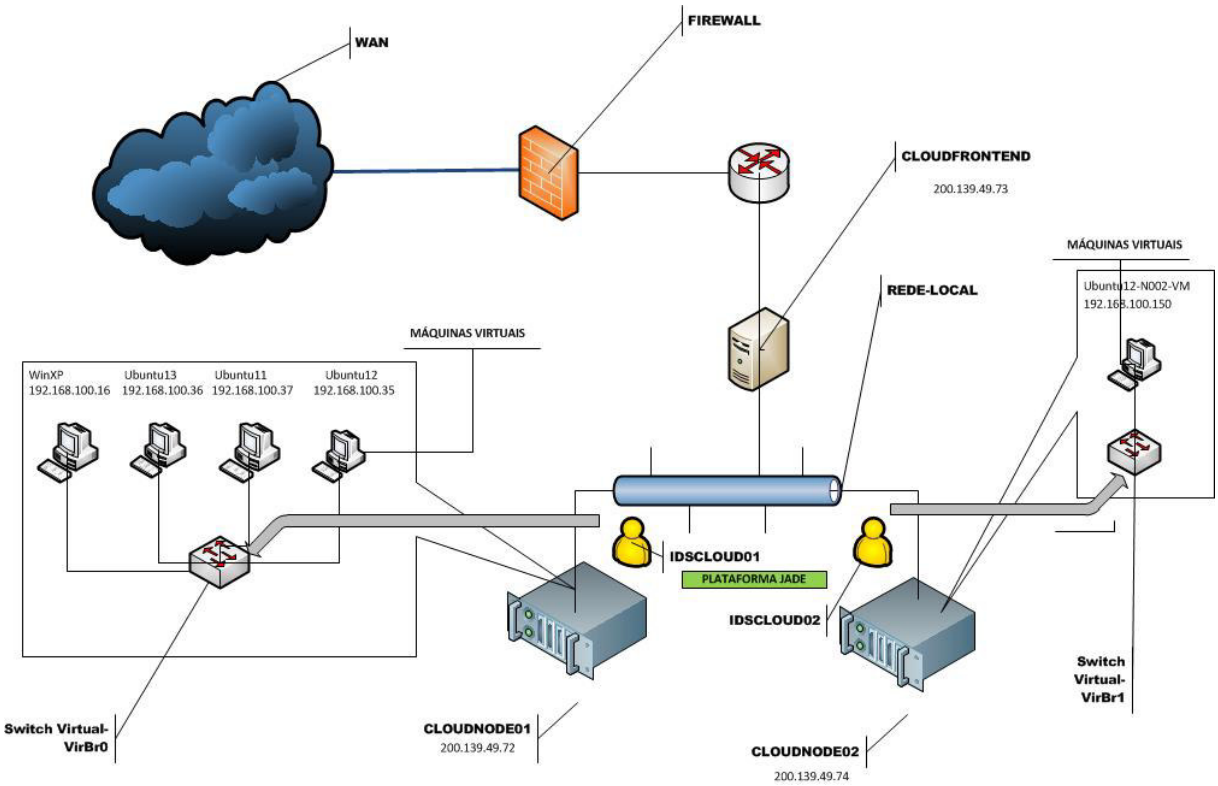


Figura 5.6: Cenário proposto para os Testes. Elaboração Própria

Na Figura 5.6 pode ser observado também as localizações estratégicas onde foram disponibilizadas as ferramentas de IDS propostas nesse trabalho. Cada IDS foi denominado com base nas características dos servidores o qual estão instalados, sendo IDSCLOUD01 para CLOUDNODE01 e IDSCLOUD02 para CLOUDNODE02.

As máquinas virtuais utilizadas para os testes foram instanciadas manualmente em cada servidor CLOUDNODE. As configurações de *hardware* e *software* de cada VM, bem como as configurações dos demais servidores são apresentadas na Tabela 5.1.

Tabela 5.1: Configuração de *hardware* e *software* do Cenário Proposto.

Configuração de *hardware* e *software* do Cenário Proposto. Elaboração Própria

NOME	Endereço de Rede	Hardware	Software
CLOUDNODE01	200.139.49.72	Processador: Core 2quad 2.6ghz, Memória: 6GB.	Opensuse 12.13; Virt-manager 10.1; JAVA 6; Hypervisor KVM.
CLOUDNODE02	200.139.49.74	Processador: Core 2 quad 2.6 ghz, Memória: 6Gb.	Opensuse 12.13; Virt-manager 10.1; JAVA 6; Hypervisor KVM.
CLOUDFRONTEND	200.139.49.73	Processador: Core 2 duo, Memória: 4GB.	Opensuse 12.13; Browser Firefox
UBUNTU11	192.168.100.37	Processador: 1 Nucleo, Memória: 512MB.	Ubuntu 12.10; Hping3.0.0 Alpha;
UBUNTU12	192.168.100.35	Processador: 1 Nucleo, Memória: 512MB.	Ubuntu 12.10; Hping3.0.0 Alpha;
UBUNTU13	192.168.100.36	Processador: 1 Nucleo, Memória: 512MB.	Ubuntu 12.10; Hping3.0.0 Alpha;
WINXP	192.168.100.16	Processador: 1 Nucleo, Memória: 512MB.	Windows XP Sp1; Apache HTTP Server
UBUNTU12-N02-VM	192.168.100.150	Processador: 1 Nucleo, Memória: 1GB.	Ubuntu 12.10; Hping3.0.0 Alpha;

5.2.2 Teste 01 - Amostragem (TCP SYN Flood)

No primeiro teste realizado, foi proposto uma simulação de tráfego anormal em solicitações de conexões TCP para um determinado *host*. O objetivo deste teste é verificar o desempenho da ferramenta de IDS em situações críticas de ataques DDoS

utilizando SYN *Flood*, ou seja, quando a vítima sofre intensas solicitações simultâneas de conexões ao ponto de não mais conseguir resolvê-las.

Neste cenário de teste, a vítima em questão é a VM com endereço de rede 192.168.100.16 e as VM's denominadas atacantes são as de endereço de rede 192.168.100.36, 192.168.100.37, 192.168.100.35. Ao detectar o provável ataque, o IDS proposto deverá reconfigurar as VM's comprometidas em modo de inspeção de rede.

Na Figura 5.7 é mostrado o ambiente virtualizado proposto para o primeiro teste. Neste ambiente é possível observar os níveis de processamento de *Central Processing Unit* (CPU) e Rede em tempo real. O servidor utilizado para este cenário foi o CLOUDNODE01.

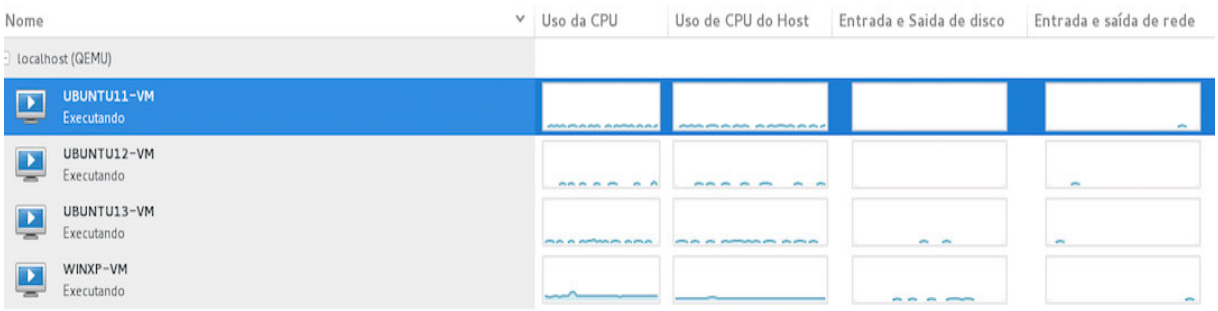


Figura 5.7: Cenário para o Primeiro Teste. Elaboração Própria

Logo após a inicialização das VM's, é iniciado as simulações de tráfego malicioso contra a vítima. Para a realização do tráfego foi utilizada a ferramenta *hping* juntamente com os parâmetros descritos a seguir:

- **-V**: Utilizado para detalhar outros parâmetros;
- **-c**: Define a quantidade de pacotes que será enviado ao destino;
- **-d**: Define o tamanho do Pacote a ser enviado;
- **-S**: Obriga a ativação da *flag* SYN do TCP em todos os pacotes;
- **-w**: Indica o tamanho da Janela TCP para a conexão;
- **-p**: Define a porta de origem para conexão;
- **-s**: Define a porta de destino para a conexão;

- **–flood**: Indica que os pacotes devem ser enviados o mais rápidos possível;
- **-a**: Utilizado para mascarar o IP de origem do atacante. O último parâmetro identifica o endereço de destino do pacote;

Na Figura 5.8 são demonstradas as execuções dos comandos citados a cima para cada VM representando um provável atacante. Como dito anteriormente, o objetivo é simular um ataque distribuído ao um único *host* de destino.

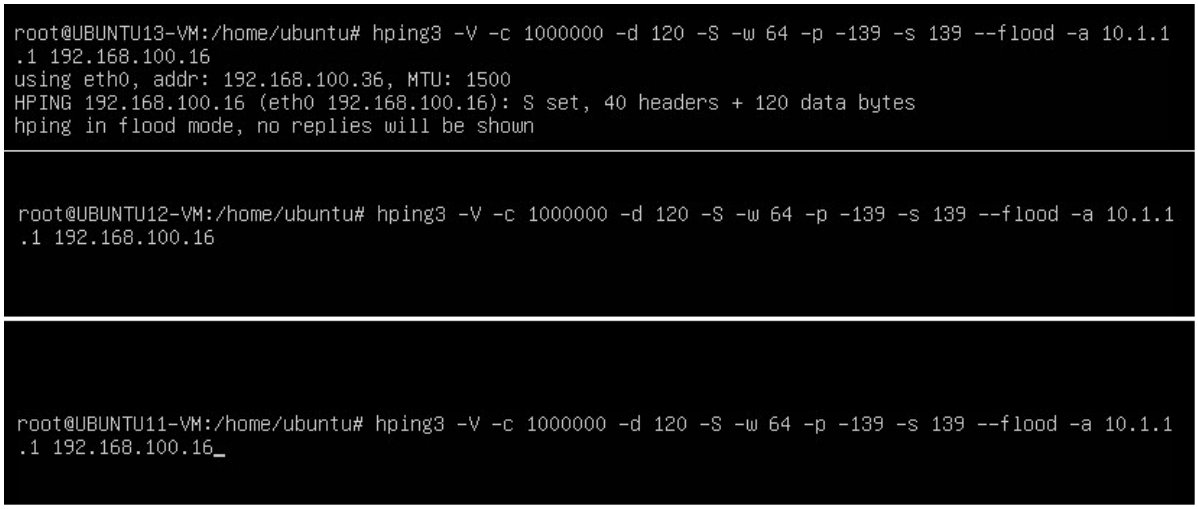


Figura 5.8: Comandos de Solicitações TCP SYN Flood. Elaboração Própria

O resultado da emissão de múltiplas conexões ao *host* vítima pode ser observado na captura de tráfego gerada pela ferramenta *Wireshark* apresentada na Figura 5.9. Nesta imagem é possível perceber o início de múltiplas solicitações TCP de origem 10.1.1.1 e destino 192.168.100.16.

Na Figura 5.10 é mostrado o nível de processamento de CPU e Rede exigidos durante as tentativas frustradas de respostas aos solicitantes. Nesse contexto, todo o *hardware* exigido para tentar resolver as solicitações é consumido de outras VM's gerenciadas pelo mesmo *Hypervisor*.

Para conter as tentativas de conexões simultâneas ao *host*, o processo de inicialização do IDS é ativado. O comando pode ser executado manualmente no terminal de execução do próprio *Hypervisor*. Ao ser executado, a plataforma de gerenciamento de agentes JADE é iniciada e os processos de *scanning* por VM's ativas também.

2	1.024592000	fe80::5054:ff:fef2:e09f	ff02::fb	MDNS	101 Standard query 0x0000 PTR _sane-po
3	2.028400000	192.168.100.40	192.168.100.1	DNS	76 Standard query 0x68e6 A daisy.ubun
4	2.028566000	192.168.100.1	192.168.100.40	DNS	108 Standard query response 0x68e6 A 9
5	7.030659000	RealtekU_c8:ec:5f	RealtekU_d7:83:97	ARP	42 Who has 192.168.100.1? Tell 192.16
7	24.674999000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
8	24.675264000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
9	24.675281000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
11	24.675293000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
12	24.675304000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
13	24.675332000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
14	24.675344000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
15	24.675368000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
16	24.675386000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
17	24.675396000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
18	24.675410000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
19	24.675434000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]
20	24.675451000	10.1.1.1	192.168.100.16	TCP	174 [TCP segment of a reassembled PDU]

Figura 5.9: Tráfego gerado por Múltiplas Solicitações TCP SYN Flood. Elaboração Própria

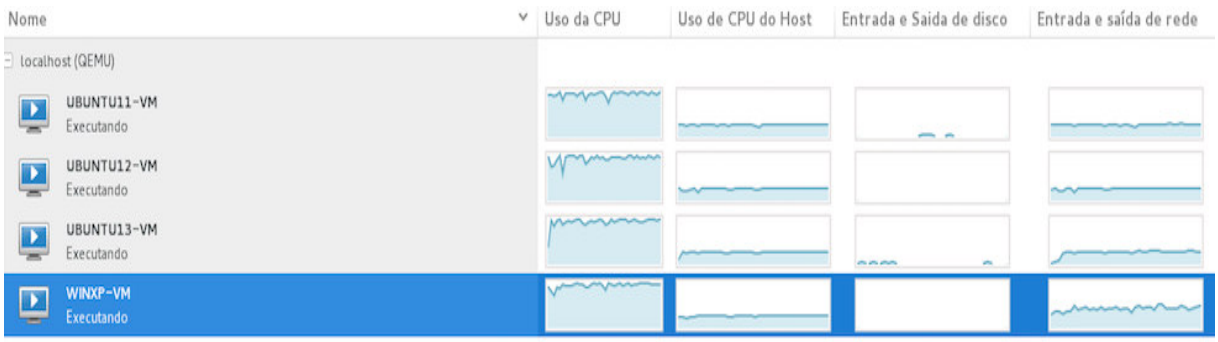


Figura 5.10: Nível de processamento CPU e Rede durante as Solicitações TCP SYN Flood. Elaboração Própria

A Figura 5.11 mostra o momento em que a plataforma é ativada para o CLOUDNODE01, bem como a identificação e inicialização de agentes para todas as VM's ativas. Neste mesmo momento também são iniciados os processos de autenticação e troca de chaves entre os agentes. Os comandos utilizados para iniciar o IDS são:

- **java**: Comando para executar o ambiente java;
- **-classpath**: Indica o caminho da classe;
- **$JadeClasses**: Define o nome da classe que contém os códigos dos agentes;
- **jade.Boot**: Indica que deve ser invocada a plataforma de agentes JADE;

Na Figura 5.11 também pode ser observado a execução de cada agente e seu vínculo com o *Hypervisor* e a VM. Desta forma, para cada VM ativa existe um agente

filter_IP representando o monitoramento e um agente *Analyzer_IP* representando o analisador. O agente que representa o controle é único por servidor e é identificado por *managerNode01*.

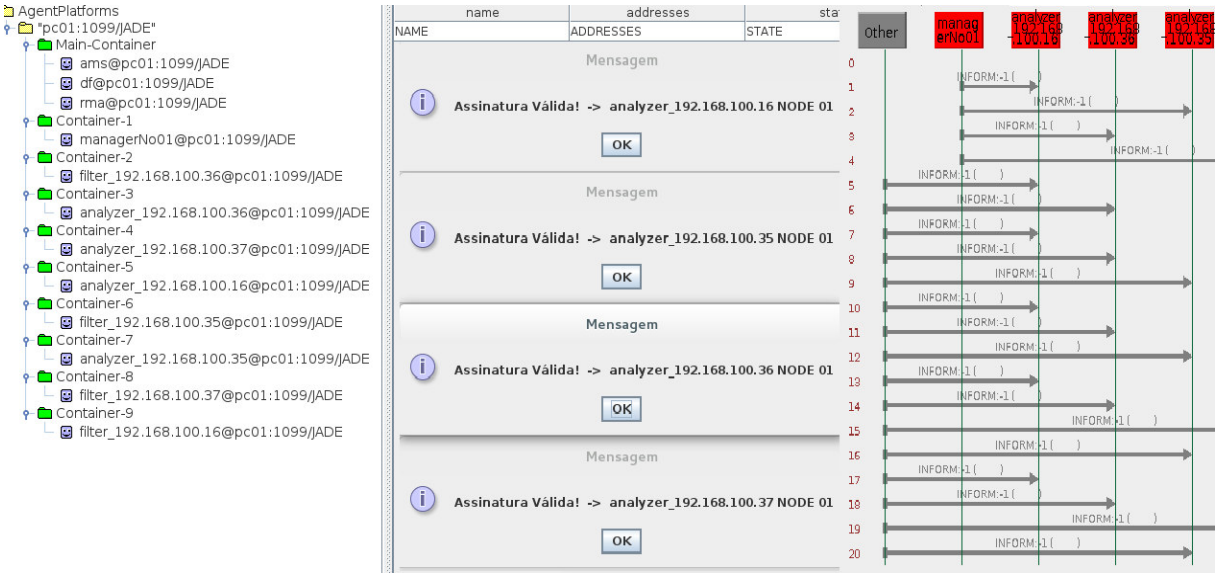


Figura 5.11: Processo de Inicialização do IDS. Elaboração Própria

Os resultados obtidos com a inicialização da Ferramenta de IDS, podem ser observados nas Figuras 5.12 e 5.13. Nelas é possível perceber a diminuição no processamento de CPU da VM vítima (WinXP) e o Isolamento em uma rede subjacente (192.168.200.0) do provável atacante (Ubuntu13). As etapas tomadas para identificação de cada VM comprometida seguem os passos a seguir:

1. Identificar a VM vítima;
2. Identificar o provável atacante;
3. Isolar o atacante em uma rede subjacente para fins de Inspeção;
4. Notificar o administrador;
5. Notificar outros agentes Controle (Se houver)

5.2.3 Teste 02 - Amostragem (UDP Flood)

O segundo teste proposto para a ferramenta, tem como finalidade simular tráfego malicioso utilizando as características do protocolo UDP. Usar UDP para

```
eth0      Link encap:Ethernet  Endereço de HW 52:54:00:57:88:74
          inet end.: 192.168.200.36  Bcast:192.168.200.255  Masc:255.255.255.0
          endereço inet6: fe80::5054:ff:fe57:8874/64 Escopo:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Métrica:1
          pacotes RX:62 erros:0 descartados:0 excesso:0 quadro:0
          Pacotes TX:63 erros:0 descartados:0 excesso:0 portadora:0
```

Figura 5.12: VM comprometida em modo de Inspeção. Elaboração Própria

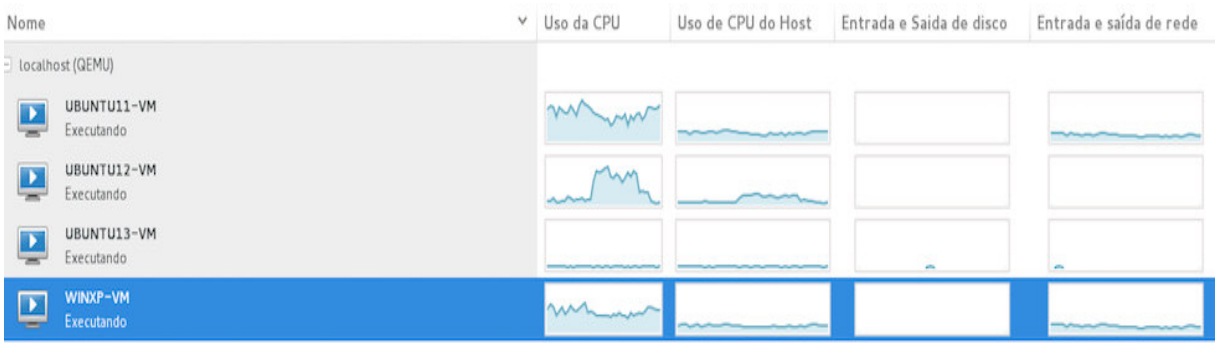


Figura 5.13: Diminuição no processamento de CPU da Vítima. Elaboração Própria

ataques de negação de serviço, não é tão simples como o TCP. No entanto, um ataque de inundação UDP pode ser iniciado através do envio de um número grande de pacotes UDP para portas aleatórias de um *host*. Como resposta para as solicitações, o *host* irá executar os passos a seguir:

1. Verificar se há uma aplicação aguardando uma conexão para a porta solicitada;
2. Identificar que nenhuma aplicação responde por essa porta;
3. Responder a solicitação com uma resposta ICMP de destino não acessível;

Assim, para um grande número de pacotes UDP, o sistema vitimado será forçado a enviar muitos pacotes ICMP, o que acarretará em negação de serviço para outras solicitações.

Portanto, para a segunda simulação será instanciada uma VM em outro servidor de virtualização localizado na mesma Infraestrutura de Nuvem. Desta forma, é possível analisar as mensagens de cooperação trocadas entre os agentes controle localizados em cada servidor. O novo servidor será denominado com CLOUDNODE02 e a VM denominada UBUNTU12NOO2-VM ambos apresentados nas Figuras 5.8 e 5.14.

Figura 5.14: Identificação da VM UBUNTU12NOO2-VM. Elaboração Própria

Os comandos para a execução do tráfego malicioso UDP também foram gerados utilizando a ferramenta *hping*. Os comandos são demonstrados na Figura 5.15 e descritos a seguir:

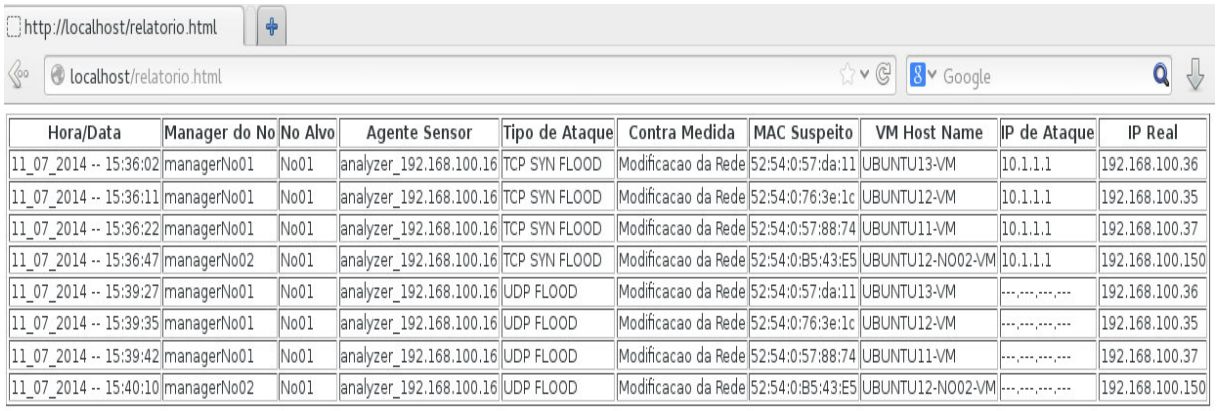
- **–flood**: Indica que os pacotes devem ser enviados o mais rápidos possível;
- **–rand-source**: Indica que os pacotes enviados devem ser destinados a portas aleatórias;
- **–udp**: Define o protocolo de estabelecimento de conexão;
- **-p**: Define a porta UDP de origem do emissor;

```
root@UBUNTU12-VM:/home/ubuntu# hping3 --flood --rand-source --udp -p 455 192.168.100.16
```

Figura 5.15: Comandos para UDP Flood. Elaboração Própria

O resultado obtido no segundo teste foi satisfatório, tendo em vista que a VM foi identificada e neutralizada no CLOUDNODE02. Todas as ações realizadas pelo IDS são gravadas em *log's* para posteriormente serem armazenadas na base de comportamento proposta no modelo. No servidor CLOUDFRONTEND é disponibilizado um relatório dessas atividades para o administrador do sistemas. O relatório final dos testes 01 e 02 são visualizados na Figura 5.16.

- **Hora/Data**: Identifica o dia e a hora que ocorreu o ataque;
- **ManagerNo**: Identifica o servidor de onde partiu o provável ataque;
- **NoAlvo**: Servidor de destino do ataque;



http://localhost/relatorio.html

localhost/relatorio.html

Google

Hora/Data	Manager do No	No Alvo	Agente Sensor	Tipo de Ataque	Contra Medida	MAC Suspeito	VM Host Name	IP de Ataque	IP Real
11_07_2014 -- 15:36:02	managerNo01	No01	analyzer_192.168.100.16	TCP SYN FLOOD	Modificacao da Rede	52:54:0:57:da:11	UBUNTU13-VM	10.1.1.1	192.168.100.36
11_07_2014 -- 15:36:11	managerNo01	No01	analyzer_192.168.100.16	TCP SYN FLOOD	Modificacao da Rede	52:54:0:76:3e:1c	UBUNTU12-VM	10.1.1.1	192.168.100.35
11_07_2014 -- 15:36:22	managerNo01	No01	analyzer_192.168.100.16	TCP SYN FLOOD	Modificacao da Rede	52:54:0:57:88:74	UBUNTU11-VM	10.1.1.1	192.168.100.37
11_07_2014 -- 15:36:47	managerNo02	No01	analyzer_192.168.100.16	TCP SYN FLOOD	Modificacao da Rede	52:54:0:B5:43:E5	UBUNTU12-NO02-VM	10.1.1.1	192.168.100.150
11_07_2014 -- 15:39:27	managerNo01	No01	analyzer_192.168.100.16	UDP FLOOD	Modificacao da Rede	52:54:0:57:da:11	UBUNTU13-VM	---,---,---,---	192.168.100.36
11_07_2014 -- 15:39:35	managerNo01	No01	analyzer_192.168.100.16	UDP FLOOD	Modificacao da Rede	52:54:0:76:3e:1c	UBUNTU12-VM	---,---,---,---	192.168.100.35
11_07_2014 -- 15:39:42	managerNo01	No01	analyzer_192.168.100.16	UDP FLOOD	Modificacao da Rede	52:54:0:57:88:74	UBUNTU11-VM	---,---,---,---	192.168.100.37
11_07_2014 -- 15:40:10	managerNo02	No01	analyzer_192.168.100.16	UDP FLOOD	Modificacao da Rede	52:54:0:B5:43:E5	UBUNTU12-NO02-VM	---,---,---,---	192.168.100.150

Figura 5.16: Relatório Administrativo do IDS. Elaboração Própria

- **AgenteSensor**: Indica o agente que identificou o ataque;
- **Ttpo de Ataque**: Mostra a assinatura que foi identificada para o ataque;
- **ContraMedida**: Mostra a ação que foi tomada diante do ataque;
- **MAC Suspeito**: Identifica o endereço físico do *host* suspeito;
- **VM Host Name**: Identifica o nome da VM Suspeita;
- **IP de Ataque**: Mostra o IP utilizado para o ataque em casos de mascaramento;
- **IP Real**: Indica o IP Real da VM suspeita;

5.2.4 Teste 03 - Teste de Viabilidade da Ferramenta

Para o terceiro teste, foi proposto um teste de viabilidade da ferramenta em um cenário de utilização dos serviços de nuvem. Neste teste foram utilizadas as ferramentas *Jmeter*: Utilizada para medir a latência [1] em requisições *Hypertext Transfer Protocol* (HTTP) e o *apache*: Servidor web utilizado para responder requisições HTTP.

O cenário proposto é idêntico ao cenário utilizado no primeiro teste, porém, a vítima (192.168.100.16) implementará um servidor *web* na porta 80. A VM 192.168.100.37 testará através da ferramenta *Jmeter* a latência gerada no servidor *web*. A VM 192.168.100.36 simulará um tráfego de TCP SYN *Flood* com destino ao servidor *web*.

[1] Diferença de tempo entre o início de um evento e o momento em que seus efeitos tornam-se perceptíveis

Os Parâmetros de teste inseridos na ferramenta *Jmeter* foram:

- **Tipo de Requisição**: Requisições HTTP;
- **Número de Usuários Virtuais**: 10 Usuários;
- **Tempo de Inicialização de cada Requisição**: 0,2 Segundos
- **Contador de Iteração**: 1 Contador;

Os testes foram divididos em três etapas:

1. **1º Etapa**: Envio de 10 (Dez) requisições HTTP para o servidor *web* em condições normais de tráfego;
2. **2º Etapa**: Envio de 10 (Dez) requisições HTTP para o servidor *web* em situação de ataque TCP SYN *Flood*;
3. **3º Etapa**: Envio de 10 (Dez) requisições HTTP para o servidor *web* em sequência à 2º Etapa;

Os resultados obtidos na 1º Etapa são apresentados no gráfico ilustrado na Figura 5.17. Nele é possível observar o tempo de resposta das requisições HTTP feitas ao servidor em situações normais de tráfego de rede. No gráfico os tempos de latência são apresentados em Millissegundos (ms).

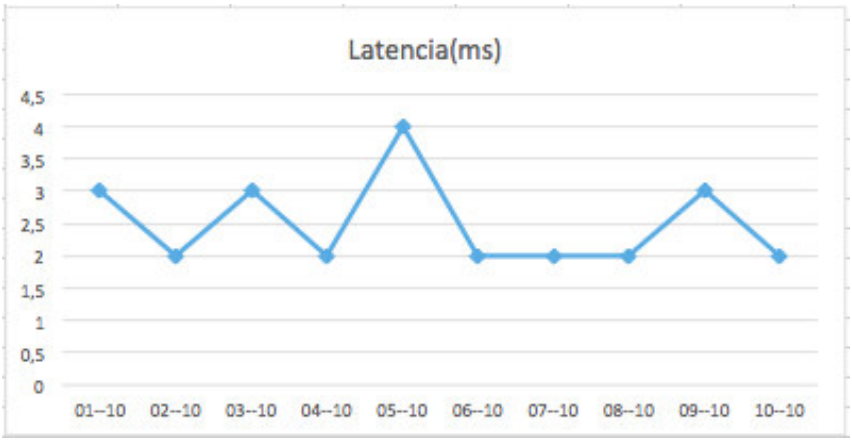


Figura 5.17: Gráfico de Resultado do 1º Teste. Elaboração Própria

Os resultados obtidos na 2º e 3º etapa são ilustrados na Figura 5.18. Com o resultado, foi possível observar um aumento no tempo de resposta das requisições

feitas ao servidor *web* em situação de ataque. No ponto (08-10), é possível identificar o pico máximo de tempo atingido de uma resposta (8000ms). Nos pontos (09-10) e (10-10) as requisições não são mais resolvidas indicando negação de serviço por parte do servidor web.

Dando continuidade a 2º etapa, a 3º etapa é iniciada no ponto (01-10), neste mesmo instante o IDS proposto é inicializado. Sendo assim, é possível identificar que a partir do ponto (04-10) o servidor *web* volta a operar em condições normais de tráfego bem próximas aos resultados obtidos na 1º Etapa.

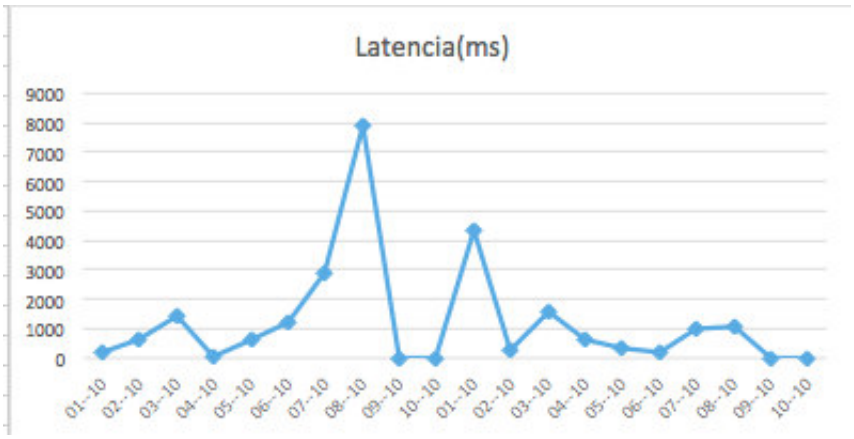


Figura 5.18: Gráfico de Resultado do 2º e 3º Teste. Elaboração Própria

5.3 Conclusões

Neste capítulo, foram apresentadas as etapas de implementação e prototipagem de cada componente do IDS proposto. Também foram realizadas simulações de tráfego malicioso para testar a eficácia da ferramenta implementada. Os resultados das simulações demonstram que com a utilização da ferramenta é possível minimizar os efeitos causados por ataques de negação de serviço em uma Infraestrutura de Nuvem.

6 Conclusões e Trabalhos Futuros

Neste capítulo, apresentam-se as conclusões, através dos principais pontos alcançados com esta dissertação e suas contribuições. Também, delineiam-se sugestões de futuras pesquisas em segurança e detecção de intrusão em Infraestruturas de Computação em Nuvem, utilizando-se sistemas multiagentes.

6.1 Contribuição do trabalho

A segurança da informação em Infraestruturas de Computação em Nuvem constitui uma problemática relevante na atualidade, motivando inúmeras tentativas de implementação de mecanismos de proteção mais eficientes, tendo em vista que os utilizados atualmente ainda não tenham conseguido alcançar o grau de segurança almejado. Ademais, boa parte dos problemas de segurança nestes ambientes, são causados por usuários da própria Infraestrutura, a maioria causada por ataques de DoS que comprometem a disponibilidade dos serviços e acabam causando negação indireta a outros usuários legítimos.

Um dos meios mais utilizados para conter ataques em Infraestrutura de Computação em Nuvem é o *Firewall* que, apesar de possuir inúmeras vantagens, apresenta algumas limitações que podem possibilitar ataques bem sucedidos e consequentemente a indisponibilidade do sistema. Nesse contexto, Sistemas de Detecção de Intrusão ou IDS, são considerados atualmente uma das medidas de segurança mais utilizadas pelos administradores de sistemas, com um bom nível de aceitação por parte dos usuários. Entretanto, ferramentas IDS específicas para Infraestruturas de Nuvem são necessárias, visto o ambiente dinâmico e heterogêneo que as define.

Baseado nas pesquisas existentes na área de IDS para Computação em Nuvem, foi proposta, nesta dissertação, como contribuição, uma arquitetura de sistema de detecção de intrusão, baseada em sistemas multiagentes, associadas ao uso de técnicas de criptografia e assinatura digital, para detectar e tratar as tentativas de

ataques de negação de serviço em uma Infraestrutura de Nuvem IaaS, englobando diversas características desejáveis para esses ambientes, como autônomia, flexibilidade para adicionar novos comportamentos para os agentes e garantias de integridade na troca de mensagens entre os componentes do IDS.

O modelo proposto foi implementado em níveis abstratos de camadas de agentes, possibilitando maior interação entre a ferramenta e o administrador de sistemas. Além disso, é possível atualizar um componente agente, por exemplo (a base de assinaturas), sem que todo o sistema seja desligado.

Como contribuição, também foi implementado a possibilidade de cooperação entre agentes de IDS localizados em *Hypervisores* distintos na mesma Infraestrutura de Nuvem, essa característica possibilita a identificação e o tratamento de usuários maliciosos antes mesmo que o ataque aconteça.

Por fim, obteve-se um protótipo funcional do modelo proposto, com a capacidade de coletar dados em tempo real e gerar informações necessárias para uma análise consistente, com o propósito de detectar e tratar tentativas de intrusão em uma Infraestrutura de Nuvem. Os resultados obtidos nos três testes implementados no Capítulo 5 são considerados satisfatórios, tendo em vista o desempenho apresentado pela ferramenta.

6.2 Trabalhos Futuros

Como proposta para trabalhos futuros e sugestões para o enriquecimento deste, pode-se listar:

- Avaliar o desempenho da ferramenta em relação a notificação de ataques (Falsos Positivos e Falsos Negativos);
- Avaliar o grau de integridade nas mensagens trocadas entre os agentes de cada camada do modelo proposto;
- Adicionar aos agentes a capacidade de detecção de ataques desconhecidos baseados no tráfego gerado e a combinação de ontologias e Inteligência Artificial;

- Adicionar camadas de agentes móveis que possibilitem a captura de tráfego com base nos *host's* dos usuários e das VM's;

- Adicionar Técnicas de Introspeção de VM para detecção de *Malwares* sem que haja sobrecarga para os usuários da Infraestrutura;

Referências Bibliográficas

[1] Mitigando ataques de egoísmo e negação de serviço em nuvens via agrupamento de aplicações. *Simposio Brasileiro de Segurança e de Sistemas Computacionais*, 1(1), 2012.

[2] An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*, 36(1):25 – 41, 2013.

[3] M. S. Ajey Singh. Overview of attacks on cloud computing. *International Journal of Engineering and Innovative Technology*, pages 321–323, 2012.

[4] R. E. Anthony T. Velte, Toby J. Velte. *Cloud Computing: A Practical Approach*. 2010.

[5] M. C. Ashish. K. Ids: Survey on intrusion detection system in cloud computing. *International Journal of Computer Science and Mobile Computing*, 3(1):497 – 502, 2014.

[6] H. Banafar and S. Sharma. Article: Intrusion detection and prevention system for cloud simulation environment using hidden markov model and md5. *International Journal of Computer Applications*, 90(19):6–11, March 2014. Published by Foundation of Computer Science, New York, USA.

[7] I. Bojanova and A. Samba. Analysis of cloud computing delivery architecture models. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 453–458, March 2011.

[8] G. Booth, A. Soknacki, and A. Somayaji. Cloud security: Attacks and current defenses. *8th ANNUAL SYMPOSIUM ON INFORMATION ASSURANCE (ASIA?13), JUNE 4-5, 2013, ALBANY, NY*, 8(1):56 – 62, 2013.

[9] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[10] Cert.br. *Estatísticas dos Incidentes Reportados ao CERT.br*. O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, 2013. Disponível em: http://www.cert.br/stats/incidentes/, Acessado em: 11-01-2014.

[11] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. Nice: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):198–211, July 2013.

[12] P. Dalapati and G. Sahoo. A survey on cloud computing. *International Journal on Computer Science and Engineering*, 5(1):435, 2013.

[13] F. de Arêa Leão Moraes. Segurança e confiabilidade em ids baseados em agente. Master's thesis, Universidade Federal do Maranhão, 2009.

[14] J. de Jesus. *Navegando na Nuvem IBM, Parte 1: Um Manual sobre Tecnologias em Nuvem*. Copyright, 2013. Disponível em: http://www.ibm.com/. Acessado em 20-06-2014.

[15] A. B. de Normas técnicas. *Técnicas de Segurança - ABNT NBR ISO/IEC 27001*. ABNT, 2006. Disponível em: http://www.abnt.org.br/. Acessado em: 02-05-2014.

[16] H. Debar and J. Viinikka. Intrusion detection: Introduction to intrusion detection and security information management. In A. Aldini, R. Gorrieri, and F. Martinelli, editors, *Foundations of Security Analysis and Design III*, volume 3655 of *Lecture Notes in Computer Science*, pages 207–236. Springer Berlin Heidelberg, 2005.

[17] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976.

[18] C. F. dos Santos. *Ambiente de Virtualização: Uma análise de desempenho*, 2011.

[19] W. H. dos Santos. *Métodos de Deteção de Ataques DDoS Compostos baseados em Filtragem Sequencial.* Instituto Militar de Engenharia, 2012.

[20] eclipse Project. *Java Eclipse IDE. Acessado em 20-03-2014*. https://www.eclipse.org/, 2014.

[21] S. Flake, C. Geiger, and J. M. Küster. Towards uml-based analysis and design of multi-agent systems, 2001.

[22] B. Gupta. Cloud computing. *COMPUSOFT, An international journal of advanced computer technology*, 1(1):31 – 34, 2012.

[23] A. Herrero and E. Corchado. Multiagent systems for network intrusion detection: A review. volume 63 of *Advances in Intelligent and Soft Computing*, pages 143–154. Springer Berlin Heidelberg, 2009.

[24] C. W. Huaibin Wang, Haiyun Zhou. Virtual machine-based intrusion detection system framework in cloud computing environment. *Journal of Computers*, 7(20):2397–2403, Oct 2012. Published by Foundation of Computer Science, New York, USA.

[25] Incapsula. *Distributed Denial of Service Attack (DDoS) Definition*, 2013. Disponível em: http://www.incapsula.com/ddos/ddos-attacks/. Acessado em: 03/03/2014.

[26] A. Jain, M. Kumar, and A. Lambha. Article: An overview and trends in cloud computing. *IJCA Proceedings on International Conference on Advances in Computer Engineering and Applications*, ICACEA(1):37–42, March 2014. Published by Foundation of Computer Science, New York, USA.

[27] Z. A. Josenilson Dias Araújo. Eicids- elastic and internal cloud-based intrusion detection system. Master's thesis, Universidade Federal do Maranhão, 2013.

[28] P. Kalagiakos and P. Karampelas. Cloud computing learning. In *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pages 1–4, Oct 2011.

[29] J. F. Kurose. *Redes de computadores e a Internet: uma abordagem top-dow.* 2006.

[30] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013.

[31] C. F. L. Lima. Agentes inteligentes para detecção de intrusos em redes de computadores. Master's thesis, Universidade Federal do Maranhão, 2002.

[32] S. Manavi, S. Mohammadalian, N. I. Udzir, and A. Abdullah. Secure model for virtualization layer in cloud infrastructure. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 1(1):32–40, 2012.

[33] C. Mazzariello, R. Bifulco, and R. Canonico. Integrating a network ids into an open source cloud computing environment. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 265–270, Aug 2010.

[34] Microsoft. *Técnicas de Criptografia*. Microsoft, 2014. Disponível em: http://support.microsoft.com/kb/246071/pt-br. Acessado em: 23-07-2014.

[35] M. Miller. *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. 2008.

[36] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42 – 57, 2013.

[37] F. A. G. Muzzi. *Análise de botnet utilizando plataforma de simulação com máquinas virtuais visando detecção e contenção.* PhD thesis, Universidade de São Paulo.

[38] R. J. O. D. Nascimento. *Engenharia de Software Orientada a Agentes. Acessado em 30-03-2014*. http://www.devmedia.com.br/engenharia-de-software-orientada-a-agentes/29238, 2006.

[39] A. Networks. *9th Annual Worldwide Infrastructure Security Report and ATLAS data*. Copyright, 2014. Disponível em: http://www.arbornetworks.com/. Acessado em: 02-07-2014.

[40] A. Networks. *Worldwide Infrastructure Security Report Volume IX*. Copyright, 2014. Disponível em: http://pages.arbornetworks.com/rs/arbor/images/WISR2014.pdf, Acessado em: 10-04-2014.

[41] N. I. of Standards and Technology. *The NIST Definition of Cloud Computing*. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2010.

[42] O. of the privacy comissionar of Canada. *Introduction to Cloud Computing*. http://www.priv.gc.ca, 2010.

[43] C. Onwubiko. Security issues to cloud computing. In *Cloud Computing*, pages 271–288. Springer, 2010.

[44] B. P. Prabahar and B. E. Edwin. Survey on virtual machine security. *International Journal of Advanced Research in Computer Engineering Technology (IJARCET)*, 1(8):115–121, 2012.

[45] A. Project. *Astah UML. Acessado em 12-02-2014.* http://astah.net.

[46] A. Project. *Jmeter. Acessado em 30-05-2014*. http://jakarta.apache.org/, 2011.

[47] J. Project. *JAVA Agent DEvelopment Framework. Acessado em 30-03-2014*. http://jade.tilab.com/, 2014.

[48] K. Project. *Kernel-based Virtual Machine*. http://www.linux-kvm.org, 2014.

[49] R. Proudfoot, K. Kent, E. Aubanel, and N. Chen. High performance software-hardware network intrusion detection system. In *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*, pages 309–312, Dec 2007.

[50] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *Knowl. Eng. Rev.*, 19(1):1–25, mar 2004.

[51] S. Roschke, F. Cheng, and C. Meinel. An advanced ids management architecture, 2010.

[52] E. Saad, K. Mahdi, and M. Zbakh. Cloud computing architectures based ids. In *Complex Systems (ICCS), 2012 International Conference on*, pages 1–6, Nov 2012.

[53] S. Sanfilippo. *Hping*. http://www.hping.org/, 2014.

[54] V. Sarathy, P. Narayan, and R. Mikkilineni. Next generation cloud computing architecture: Enabling real-time dynamism for shared distributed physical infrastructure. In *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, pages 48–53, June 2010.

[55] J. Sen. An agent-based intrusion detection system for local area networks. *IJCNIS*, 2(2), 2010.

[56] U. Siddiqui, G. Tahir, A. Rehman, Z. Ali, R. Rasool, and P. Bloodsworth. Elastic jade: Dynamically scalable multi agents using cloud resources. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 167–172, Nov 2012.

[57] A. Singh and D. M. Shrivastava. Overview of attacks on cloud computing. *International Journal of Engineering and Innovative Technology (IJEIT)*, 1(4), 2012.

[58] L. G. Siqueira. Tolerância a falhas para o nidia: um sistema de detecção de intrusão baseado em agentes inteligentes. Master's thesis, Universidade Federal do Maranhão, 2006.

[59] S. Sridharan. A performance comparison of hypervisors for cloud computing. Master's thesis, University of North Florida, 2012.

[60] W. Stallings. *Criptografia e Segurança de Redes - Princípios e Práticas*. 2008.

[61] C. TAURION. *Cloud Computing - Computação em Nuvem*. BRASPORT.

[62] Telelink. *Ataques de DDoS.*, 2013. Disponível em: http://itsecurity.telelink.com/distributed-denial-of-service-attack/synflood. Acessado em: 10/12/2013.

[63] G. K. Thiruvathukal and M. Parashar. Cloud computing [guest editorial]. *Computing in Science Engineering*, 15(4):8–9, July 2013.

[64] H. Tianfield. Cloud computing architectures. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1394–1399, Oct 2011.

[65] K. Venkataramana and M. Padmavathamma. Multi-agent intrusion detection and prevention system for cloud environment. *International Journal of Computer Applications*, 49(20):24–29, July 2012. Published by Foundation of Computer Science, New York, USA.

[66] P. Veríssimo and L. Rodrigues. Fundamental security concepts. In *Distributed Systems for System Architects*, volume 1 of *Advances in Distributed Computing and Middleware*, pages 377–393. Springer US, 2001.

[67] K. Vieira, A. Schulter, C. Westphall, and C. Westphall. Intrusion detection for grid and cloud computing. *IT Professional*, 12(4):38–43, July 2010.

[68] J. S. Ward and A. Barker. A cloud computing survey: Developments and future trends in infrastructure as a service computing. *CoRR*, abs/1306.1394, 2013.

[69] Q. Zhang, L. Cheng, and A. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7 – 18, 2010.