UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

JOSENILSON DIAS ARAÚJO

# UM MODELO DE DETECÇÃO DE INTRUSÃO PARA AMBIENTES DE COMPUTAÇÃO EM NUVEM

São Luís
2013

JOSENILSON DIAS ARAÚJO

# UM MODELO DE DETECÇÃO DE INTRUSÃO PARA AMBIENTES DE COMPUTAÇÃO EM NUVEM

Dissertação de Mestrado submetida à Coordenação do Curso de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão para obtenção do título de Mestre em Engenharia da Eletricidade – Área de Concentração: Ciência da Computação.

Orientador: Prof. Ph.D. Zair Abdelouahab

São Luís
2013

Araújo, Josenilson Dias.

Um modelo de detecção de intrusão para ambientes de computação em nuvem/ Josenilson Dias Araújo – São Luís, 2013.

120 f.

Impresso por computador (fotocópia).

Orientador: Zair Abdelouahab.

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2013.

1. Computação em nuvem. 2. Sistema de detecção de intrusão. 3. Máquinas virtuais. I. Título.

CDU 004.056.52

**UM MODELO DE DETECÇÃO DE INTRUSÃO PARA AMBIENTES DE COMPUTAÇÃO EM NUVEM**

**Josenilson Dias Araújo**

Dissertação aprovada em 28 de junho de 2013.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Prof. Djamel Fawzi Hadj Sadok, Ph.D.
(Membro da Banca Examinadora)

Prof. Denivaldo Cicero Pavão Lopes, Dr.
(Membro da Banca Examinadora)

A Deus.

A meus pais.

A minha esposa e a minha

filha, razões de minha vida.

# AGRADECIMENTOS

A Deus, pela oportunidade que me foi concedida de realizar o mestrado.

Ao Prof. Zair Abdelouhab, pela orientação, confiança e incentivo que foram de fundamental importância para a conclusão desta dissertação.

A minha querida esposa Carla Marina, pela compreensão, carinho e apoio incondicionais.

Aos meus pais, que sempre me apoiaram e me incentivaram em todos os momentos de minha vida.

A minha filhinha Lair, pelo carinho, brincadeiras e incentivo.

Aos meus amigos Mauro Melo e Eduardo Henrique, companheiros de jornada no mestrado.

Aos meus colegas de mestrado, Renato, Gleidson, Dhuleane, Luís, Berto, Jesseildo, Rafael, Liana e Wladmir, pelo companheirismo e auxílio técnico indispensável.

A minha família que sempre se manteve firmes ao meu lado.

A Leonardo Melo, que muito auxiliou na finalização de minhas pesquisas.

A todos que de alguma forma contribuíram com o meu trabalho e que certamente não serão esquecidos.

*“Não sabendo que era impossível, foi lá e fez.”*

*Paulo Leminsk*

# RESUMO

A elasticidade e abundante disponibilidade de recursos computacionais são atrativos para intrusos explorarem vulnerabilidades da nuvem, podendo assim lançar ataques contra usuários legítimos para terem acesso a dados privados e privilegiados. Para proteger efetivamente os usuários da nuvem, um Sistema de Detecção de Intrusão ou IDS deve ter a capacidade de expandir-se, aumentado ou diminuindo rapidamente a quantidade de sensores, de acordo com o provisionamento de recursos, além de isolar o acesso aos níveis de sistema e infraestrutura. A proteção contra ameaças internas na nuvem deve ser planejada, pois a maioria dos sistemas de proteção não identifica adequadamente ameaças internas ao sistema. Para isso, a solução proposta utiliza as características de máquinas virtuais como rápida inicialização, rápida recuperação, parada, migração entre diferentes hosts e execução em múltiplas plataformas na construção de um IDS que visa monitorar o ambiente interno de máquinas virtuais da nuvem, inserindo sensores de captura de dados na rede local das VMs dos usuários, podendo assim detectar comportamentos suspeitos dos usuários.



Palavras Chaves: Computação em Nuvem, sistema de detecção de intrusão, máquinas virtuais, elasticidade, ameaças internas.

ABSTRACT

The elasticity and large consumption of computational resources are becoming attractive for intruders to exploit the cloud vulnerabilities to launch attacks or have access of private and privileged data of cloud users. In order to effectively protect the cloud and its users, the IDS must have the capability to quickly scale up and down the quantity of sensors according to the resources provisioned, besides of isolating the access to the system levels and infrastructure. The protection against internal cloud threats must be planned because of the non-adequate threatening identification system in most protection systems. For this, the proposed solution uses virtual machines features as fast recovery, start, stop, migration to other hosts and cross-platform execution in IDS based VM, to monitor the internal environment of the cloud virtual machines by inserting data capture sensors at the local network of the VM users, this way, it can detect suspicious user behaviors.

Keywords: cloud computing, intrusion detection system, virtual machines, elasticity, internal threatening’s.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

| | |
|---|---|
| AMI | *Amazon Machine Images* |
| API | *Application Programing Interface* |
| BD | Banco de Dados |
| CIDF | *Common Intrusion Detection Framework* |
| CIDS | *Cloud Intrusion Detection System* |
| CSA | Cloud Security Alliance |
| DOS | *Denial of Service* |
| EC2 | *Elastic Cloud Computing* |
| ENISA | *European Network and Information Security Agency* |
| Eucalyptus | *Elastic Utility Computing Architecture Linking Your Programs To Useful System* |
| HIDS | *Host-based Intrusion Detection System* |
| IaaS | *Infrastructure as a Service* |
| ICMP | *Internet Control Message Protocol* |
| IDC | *International Data Corporation* |
| IDMEF | *Intrusion Detection Message Exchange Format* |
| IDS | *Intrusion Detection System* |
| IDSaaS | *Intrusion Detection System as a Service* |
| IIDB | Padrão de Intrusos e Intrusões |
| IP | *Internet Protocol* |
| KVM | *Kernel-based Virtual Machine* |
| LABSAC | Laboratório de Sistemas em Arquiteturas Computacionais |
| NIDIA | *Network Intrusion Detection System based on Intelligent Agents* |

| | |
|---|---|
| NIDS | *Network-based Intrusion Detection System* |
| NIST | *National Institute of Standards and Technology* |
| P2P | *Peer to Peer* |
| PaaS | *Platform as a Service* |
| QoS | *Quality of Service* |
| SaaS | *Software as a Service* |
| SLA | *Service Level Agreement* |
| SQL | *Structured Query Language* |
| SSL | *Security Socket Layer* |
| TCP | *Transmission Control Protocol* |
| TI | Tecnologia da Informação |
| UDP | *User Datagram Protocol* |
| VLAN | *Virtual Local Area Network* |
| VM | Virtual Machine |
| VMM | Virtual Machine Monitor |
| VN | *Virtual Network* |
| VPC | Virtual Private Cloud |
| VPN | Virtual Private Network |

SUMÁRIO

# INTRODUÇÃO

A computação em nuvem vem surgindo como um novo paradigma computacional, propondo uma nova forma de utilização dos recursos computacionais, como software e hardware, através do pagamento por uso, do consumo sob demanda e com acesso a esses serviços pela internet. Tal paradigma permite que o consumidor ou cliente, que pode ser uma empresa ou um simples usuário, contrate os serviços de um provedor e pague de acordo com o que consome, terceirizando o desenvolvimento de software e administração de servidores.

De uma forma geral, podemos dizer que computação em nuvem é um tipo de computação onde escalabilidade, adaptabilidade e elasticidade de recursos de TI são providos como serviço para múltiplos usuários [1].

O número de empresas que utilizam ou pretendem utilizar serviços de computação em nuvem é cada vez maior. Segundo uma pesquisa realizada pela CISCO em 2012 [2], mais de 90% das organizações consultadas consideraram que a computação em nuvem exerce um papel relevante na sua estrutura de TI, enquanto que apenas 52% consideraram essa importância em pesquisa realizada anteriormente. Essa mudança em direção à nuvem é atraída pelas inúmeras vantagens oferecidas por esse modelo, como elasticidade, abundante disponibilidade de recursos e medição de consumo.

Contudo, a mudança em direção a esse novo paradigma traz vários desafios, entre eles a preocupação com a segurança dos dados armazenados e também com a própria integridade do ambiente de nuvem. Tal preocupação é justificada pelo

aumento do número de incidentes de segurança reportados ao CERT[1] [3], como pode ser visto na Figura 1.1, onde se pode observar que de 1999 a 2012 houve um crescimento de 15.000%, com 466.029 incidentes relatados somente em 2012. Portanto, é importante desenvolver mecanismos de monitoramento adequados para nuvem que possam detectar e responder a ações maliciosas que configurem uma intrusão.

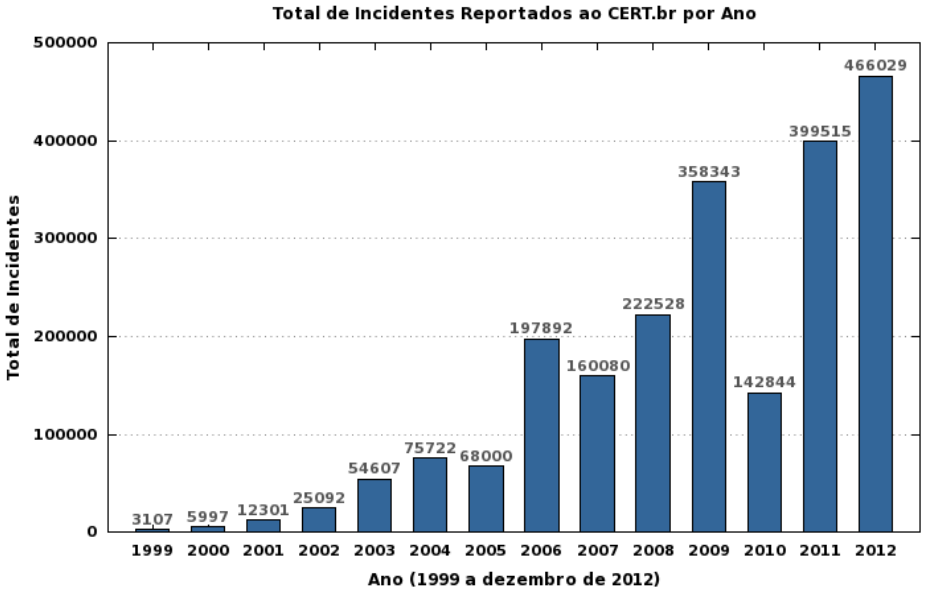



**Figura 1.1:** Estatísticas segundo o site CERT.br [3]

Mecanismos de proteção como *firewall* [4], Sistemas de Detecção de Intrusão (IDS), VPN´s (*Virtual Private Networks*) e antivírus são de grande importância para compor um ambiente seguro, dentre os quais podemos destacar os Sistemas de Detecção de Intrusão (IDS) que focam na detecção de atividades intrusivas na rede [5]. No entanto, por conta da elasticidade e da arquitetura distribuída da nuvem, o IDS precisa ser adaptado para proteger eficazmente esse novo ambiente. Tal IDS deve ser dinamicamente expansível, pois, para monitorar com eficácia um ambiente de nuvem, que aumenta sob demanda a quantidade de recursos, deve acompanhar

[1] O CERT-BR é um Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil é mantido pelo NIC.br, do Comitê Gestor da Internet no Brasil, e atende a qualquer rede brasileira conectada à Internet. O seu site é http://www.cert.br/.

essa expansão à medida que a nuvem disponibiliza, remove ou redireciona recursos computacionais aos usuários de acordo com a demanda de consumo.

## 1.1 Definição do Problema

A preocupação com a segurança é apontada como uma das principais barreiras à utilização da computação em nuvem pelas instituições, conforme pesquisa realizada pelo International Data Corporation (IDC)[2] em 2009 [6], ilustrado na Figura 1.2. A possibilidade de intrusos acessarem informações sigilosas ou utilizarem os recursos da nuvem de forma indevida tem preocupado tanto grandes empresas quanto usuários comuns devido aos prejuízos que uma invasão poderia ocasionar.

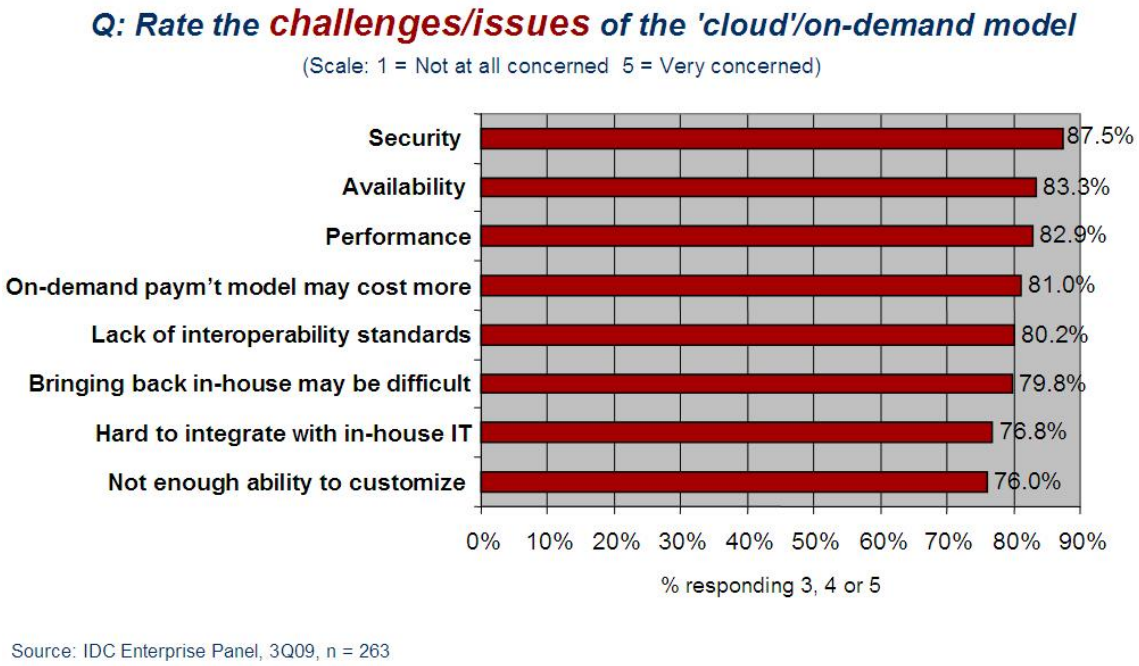



**Figura 1.2:** ranking IDC de desafios de segurança em Computação em nuvem [6]

[2] International Data Corporation (IDC) é uma empresa de pesquisa, análise e consultoria especializada em tecnologia, telecomunicações e tecnologia da informação do consumidor. A IDC é subsidiária da IDG, estando sediada em Framingham, Massachusetts e tem mais de 1.000 analistas de todo o mundo, cobrindo a tecnologia e a indústria de oportunidades, tendências e previsões em mais de 110 países.

Atualmente, os administradores de rede podem utilizar os sistemas de detecção de intrusão (IDS) para identificar ações intrusivas, visando à prevenção de ataques e à tomada de decisões quando um ataque é iniciado ou detectado.

Porém, devido à natureza distribuída dos ambientes de computação em nuvem, o IDS precisa monitorar vários nós físicos e virtuais, além de ser adaptado aos níveis de serviço para impedir que intrusos possam explorar possíveis vulnerabilidades. Além disso, ameaças internas podem não ser detectadas devido a usuários legítimos utilizarem seus privilégios para realizarem ações maliciosas contra outros usuários ou mesmo contra o sistema de computação em nuvem.

Por isso, para proteger efetivamente os usuários da nuvem, um IDS deve ter a capacidade de expandir-se, aumentado ou diminuindo rapidamente a quantidade de sensores, de acordo com o provisionamento de recursos, além de isolar o acesso aos níveis de sistema e infraestrutura. Para a implantação de um IDS em ambientes de computação em nuvem, a virtualização pode ser um elemento chave por proporcionar uma maior cobertura contra ataques através do isolamento do ambiente a ser monitorado, da rápida inicialização, parada e recuperação de máquinas virtuais contendo componentes do IDS [7].

Para melhorar a proteção, um componente que monitore constantemente os recursos da nuvem poderia ser adicionado a um IDS baseado em VM, possibilitando assim acompanhar a expansão da nuvem, sendo que à medida que o ambiente de nuvem disponibilizasse recursos para os usuários, o IDS aumentaria a quantidade de sensores para acompanhar o crescimento da nuvem. Além disso, uma definição das superfícies de ataques em que o IDS deve atuar é necessária para definir que tipo de ações maliciosas serão repelidas ou mitigadas pelo IDS.

## 1.2 Solução Proposta

A solução proposta baseia-se na proteção das máquinas virtuais do ambiente de computação em nuvem contra usuários internos que possam utilizar alguma máquina virtual para realizar atividades maliciosas ou que apresentem comportamentos suspeitos. O monitoramento das máquinas virtuais é feito através de sensores de IDS espalhados pelo ambiente de nuvem a ser protegido.

A finalidade é proteger as VMs para evitar que usuários maliciosos tenham acesso ao conteúdo de outras VMs ou possam fazer um uso abusivo para efetuar ataques a outras VMs.

A Figura 1.3 mostra o modelo conceitual do IDS.

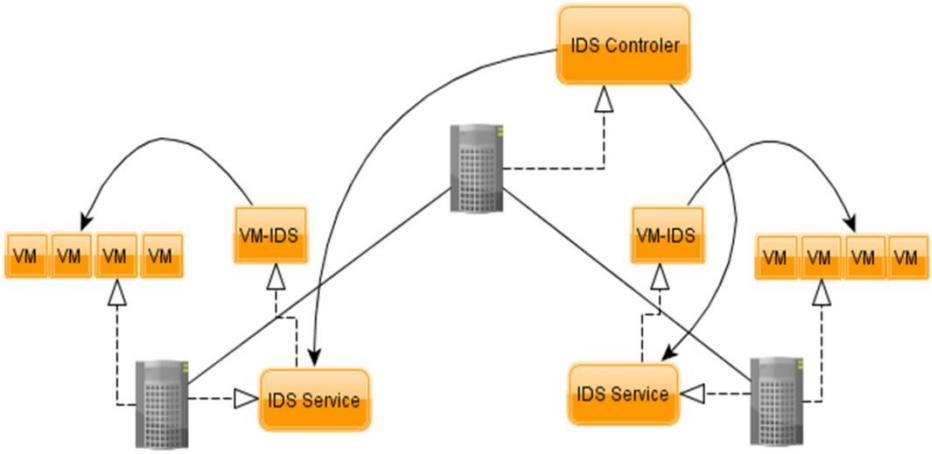



**Figura 1.3:** Modelo Conceitual do IDS Proposto

## 1.3 Metodologia Utilizada

O desenvolvimento deste trabalho consiste na seguinte metodologia:

- Inicialmente, a pesquisa bibliográfica sobre sistemas de detecção de intrusão ou IDS (*Intrusion Detection System*), Computação em nuvem, segurança em sistemas de Computação em nuvem, virtualização, virtualização em IDS e o sistema IDS-NIDIA;

- Montagem de um ambiente de testes, composto por um microcomputador utilizando o sistema operacional linux e com o hipervisor kvm instalado, disponibilizando assim a criação de várias máquinas virtuais para que os testes sobre o IDS possam ser realizados. Outro ambiente de testes foi utilizado com dois microcomputadores com o midleware opensource Eucalyptus instalado, formando assim uma nuvem privada IaaS. Desta forma as funcionalidades do IDS poderão ser testadas nesses ambientes simulando uma nuvem real;
- Implementação do IDS no sistema operacional da nuvem com a finalidade de proteger as máquinas virtuais contra ataques internos;
- Possibilidade de adaptação ao IDS-NIDIA;
- Teste do IDS.

## 1.4 Objetivos gerais e específicos

O objetivo deste trabalho é propor um modelo para detecção de intrusão em computação em nuvem que monitore as tentativas internas de invasões, as mudanças no comportamento ou o mau uso de recursos, especificamente de máquinas virtuais dos usuários.

No sentido de alcançar o objetivo geral pretendido, buscar-se-á atingir os seguintes objetivos específicos:

1. Propor um modelo para detecção de intrusão para computação em nuvem que esteja de acordo com as normas nacionais [39] e internacionais de segurança, mencionadas em [66], e que possa, futuramente, ser realizada uma integração com os agentes do IDS NIDIA [8];

2. Implementar um protótipo que atue a nível de infraestrutura (IaaS) do middleware de computação em nuvem;

3. Prover segurança nas máquinas virtuais dos usuários de IaaS;

4. Implementar o IDS com a linguagem de programação JAVA e posteriormente adaptar os módulos para que sejam compatíveis com o IDS NIDIA;

5. O IDS deve atuar, baseado em estudos sobre as superfícies de ataques em computação em nuvem [9], cobrindo ataques na superfície entre o usuário e o

ambiente de nuvem, impedindo assim que atacantes passem por usuários legítimos e obtenham controle sobre os recursos da nuvem. Com isso, pode ser evitado ataques aos serviços hospedados nos provedores de nuvem (por meio da superfície de ataque entre provedor e os serviços). Assim, ao proteger a interface entre usuário e provedor, evita-se que ataques sejam executados na interface entre provedor e serviços;

6. O IDS deve focar na segurança das máquinas virtuais dos usuários de IaaS;
7. Contribuir para aumentar o nível de segurança em ambientes de computação em nuvem.

## 1.5 Estrutura da dissertação

Este trabalho está organizado da seguinte forma:

No Capítulo 2, a teoria sobre os principais conceitos em computação em nuvem, virtualização, segurança em computação em nuvem, sistemas de detecção de intrusão além dos trabalhos relacionados que serviram de base para o trabalho é apresentada.

No Capítulo 3, o modelo do IDS para computação em nuvem proposto neste trabalho será apresentado.

No Capítulo 4, serão apresentados a implementação do IDS proposto, os testes e resultados.

No Capítulo 5, apresentamos a conclusão deste trabalho.

# 2. Fundamentação Teórica

Neste capítulo, são apresentados alguns conceitos importantes sobre Computação em nuvem, virtualização, Segurança em computação em nuvem, sistemas de detecção de intrusão e, por fim, sistema de detecção de intrusão aplicados à Computação em nuvem. Estes conceitos foram de extrema importância para o desenvolvimento deste trabalho.

## 2.1. Computação em Nuvem

A computação em nuvem pode ser definida como uma infraestrutura de processamento de dados na qual as aplicações, dados e poder de processamento estão disponíveis remotamente através da internet. A nuvem propõe uma nova forma de utilização dos recursos de TI, disponibilizando aos usuários acesso a um *array* de aplicações e serviços, ao mesmo tempo em que abstrai as complexidades envolvidas na entrega dos mesmos. A mudança na localização de dados e programas, ao saírem do desktop dos usuários para a nuvem, proporciona uma mudança geográfica na computação, onde o processamento é realizado remotamente em *Data Centers* que disponibilizam um *pool* de recursos de TI, como poder de processamento, armazenamento e software, tudo sob demanda aos usuários [10].

Sendo assim, uma das principais propostas do paradigma de computação em nuvem é a nova forma de utilização dos recursos computacionais, onde o consumo destes recursos, assim como no fornecimento de água e eletricidade, passa a ser cobrado pela utilização. Dessa forma, o usuário, ao contratar os serviços de um provedor, efetua o pagamento de acordo com o consumo, terceirizando a administração de recursos de TI.

Por se tratar de um novo modelo, a computação em nuvem não tem ainda suas especificações e definições padronizadas, resultando em várias definições e conceitos, como em [1][11], onde a computação em nuvem pode também ser definida como um conjunto de recursos computacionais virtualizados (máquinas virtuais) disponibilizado para usuários através da Internet.

Outra definição é que a nuvem pode ser vista como um modelo de computação distribuída que deriva características da computação em grades, no que diz respeito à provisão de informação sob demanda para múltiplos usuários concorrentes e

utiliza-se da virtualização para oferecer recursos por meio de serviços aos usuários os quais necessitam apenas de um browser e conexão de internet para consumir tais recursos [12]. Assim, a virtualização tornou-se um elemento chave por oferecer um conjunto de recursos de TI (armazenamento, software, poder de processamento, entre outros) como serviços aos usuários, que precisam somente de acesso à web para utilizá-los [13].

Uma das definições mais amplamente aceita por vários pesquisadores é a do NIST (National Institute of Standards and Technology), que define a computação em nuvem como um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços [14].

Com o aumento na utilização de cada vez mais recursos computacionais, principalmente hardware e software, em diversas áreas, a computação em nuvem apresenta-se como uma boa alternativa para as empresas por proporcionar serviços de TI com pagamento baseado no uso.

### 2.1.1 Arquitetura da computação em Nuvem

A computação em nuvem pode ser concebida como um modelo em camadas, onde na base da estrutura da nuvem temos a infraestrutura física, composta por servidores, equipamentos de rede e sistemas operacionais. Logo em seguida temos uma camada de virtualização e máquinas virtuais. Depois, temos uma camada formada por ambientes operacionais e programas de desenvolvimento funcionando sobre as máquinas virtuais e por fim uma camada de software disponível para usuários finais. A visualização em camadas pode ser vista na Figura 2.1.

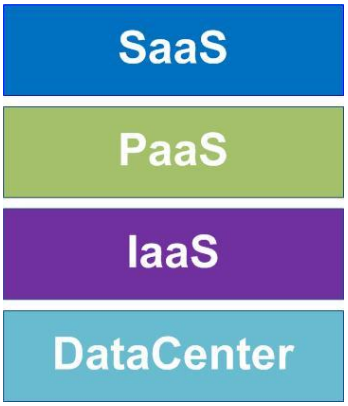



**Figura 2.1:** Visualização em Camadas da Computação em Nuvem

O NIST define um modelo de computação em nuvem composto por cinco características essenciais, três modelos de serviços e quatro modelos de implantação [14], como podemos ver na Figura 2.2. A seguir veremos uma descrição desse modelo.

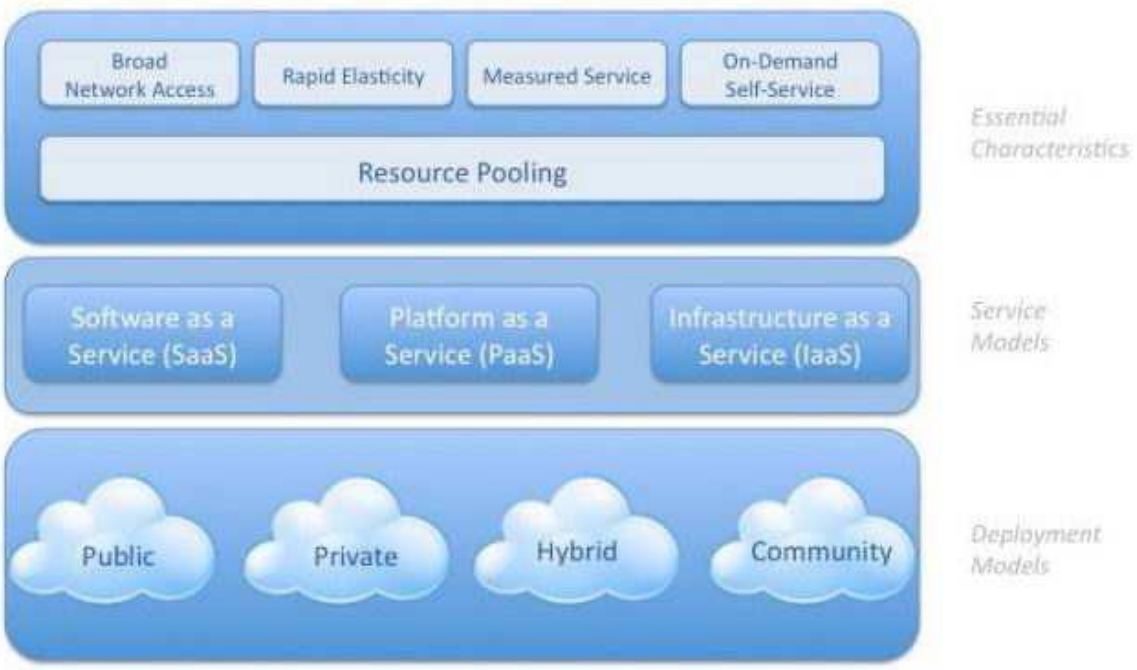



**Figura 2.2:** Modelo da Computação em Nuvem definido pelo NIST [14]

#### 2.1.1.1 Características essenciais da computação em Nuvem

As principais características de computação em nuvem são auto-serviço sob demanda, ou on-*demand self-service*, virtualização de recursos, independência de localização com acesso aos recursos via Internet, elasticidade e modelo de pagamento baseado no consumo [13], [14].

O auto-serviço sob demanda proporciona a alocação de recursos para o usuário da nuvem, sem a necessidade de intervenção do provedor, conforme sua necessidade. Os recursos de TI, como hardware e software, podem ser automaticamente reconfigurados dentro da nuvem e essas modificações devem ocorrer de forma transparente para os usuários.

A virtualização é uma das principais tecnologias da computação em nuvem, onde a máquina virtual é a unidade básica das plataformas de computação em nuvem [15]. A virtualização de recursos, conseguida através do uso de tecnologias já estabelecidas como *virtual machines*, virtualização de memória, virtualização de armazenamento e virtualização de rede (VPN), desatrela os serviços de

infraestrutura dos recursos físicos (hardware, rede). Com essa abstração, a localização real dos recursos é tratada na camada mais baixa da computação na nuvem (IaaS), sendo transparente para as demais camadas.

A independência de localização é conseguida com a total disponibilidade dos serviços, tornando-os acessíveis de qualquer lugar que se tenha acesso à infraestrutura de rede, onde a nuvem apresenta-se como o único ponto para acesso à todas as necessidades de recursos de TI dos usuários.

A elasticidade é a capacidade de alocar e remover rapidamente grandes quantidades de recursos de TI em tempo de execução. Recursos podem ser requisitados de uma forma elástica e rápida, dando a impressão de serem ilimitados. A virtualização auxilia na característica de elasticidade da computação em nuvem, através da criação de várias instâncias de recursos virtuais sob um único recurso real [10].

O pagamento baseado no consumo é outra característica central da computação em nuvem, onde os usuários necessitam pagar ao provedor apenas os serviços que foram utilizados.

#### 2.1.1.2 Modelos de Serviço da Computação em Nuvem

A computação em nuvem pode ser classificada de acordo com o modelo de serviço que oferece, como visto na Figura 2.3. Esses modelos são descritos usando a taxonomia XaaS, que foi primeiramente usada por Scott Maxwell em 2006 [16], onde podemos substituir o “X” por “S” de Software, “P” de plataforma ou “I” de infraestrutura.

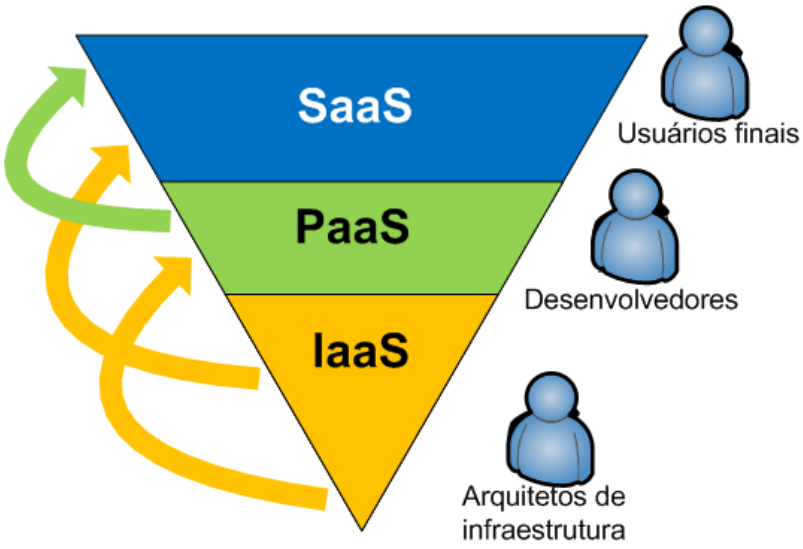



**Figura 2.3:** Modelos de Serviços Oferecidos pela Computação em Nuvem adaptado de [81]

Os três modelos de serviços da Computação em nuvem são:

- IaaS-Infraestructure *as a Service* (Infraestrutura como Serviço), fornece recursos computacionais como processamento, armazenamento e rede através da utilização da virtualização com a disponibilização de várias máquinas virtuais em um único hardware compartilhado por meio do hipervisor. O termo IaaS se refere a uma infraestrutura computacional provisionada através de técnicas de virtualização de hardware. Dependendo das requisições das aplicações, esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo o provisionamento de recursos. Tem como foco a terceirização de infraestrutura e redução de desperdícios, sendo seu público-alvo os arquitetos de infraestrutura. Como exemplos de IaaS temos o Amazon EC2 (*Elastic Cloud Computing*) [17] e o Eucalyptus (*Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*) [18].

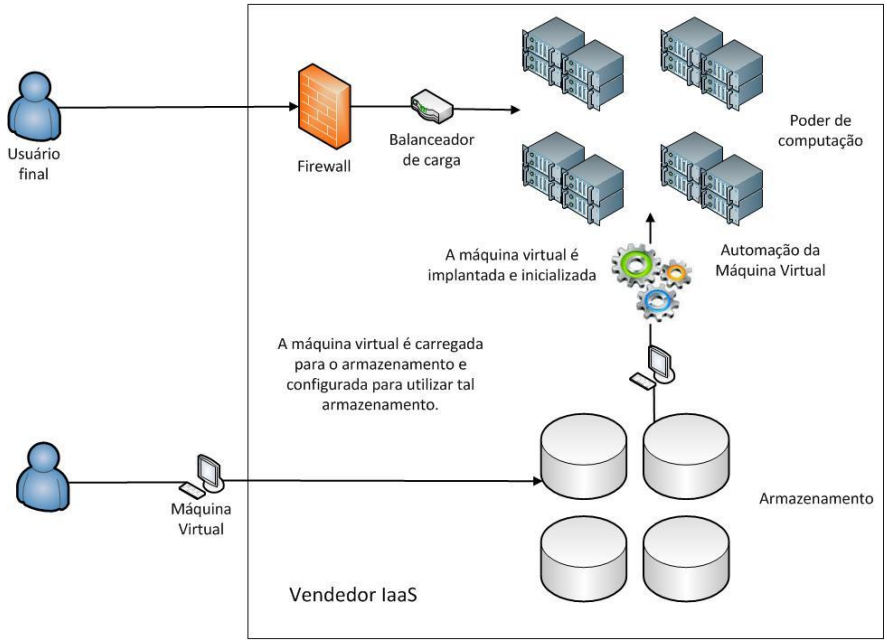



**Figura 2.4:** IaaS- Infraestrutura as a Service (Infraestrutura como serviço) adaptado de [81]

- PaaS (Plataforma como Serviço), é o serviço que permite ao usuário utilizar a infraestrutura da nuvem para criar e implantar suas próprias aplicações usando linguagens, bibliotecas e ferramentas disponibilizadas pelo provedor. Proporciona um ambiente de computação em camadas como serviço para criação, teste e hospedam de aplicativos em nuvem, tendo como público-alvo desenvolvedores. PaaS facilita a implantação de aplicações ao fornecer todas as facilidades necessárias para suportar o

ciclo de vida completo de construção e entrega de aplicações e serviços web disponíveis pela Internet. Podemos citar como exemplos de PaaS o Google appEngine [19] e Microsoft Azure.

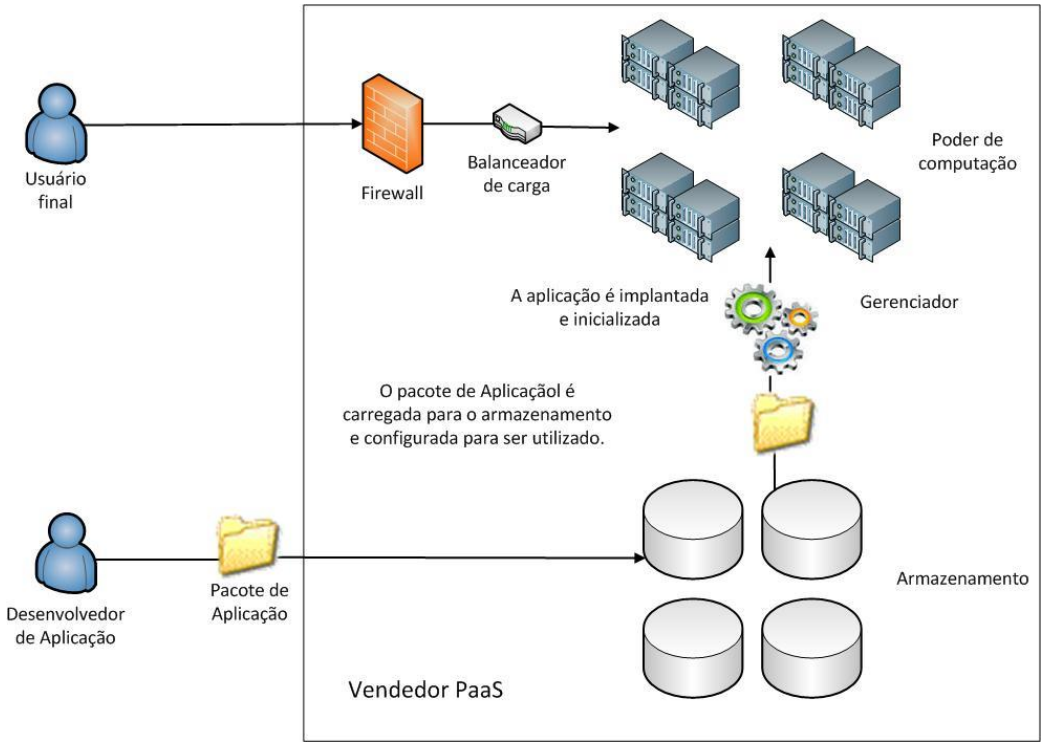



**Figura 2.5:** PaaS-Plataform as a Service (Plataforma como serviço) adaptado de [81]

- SaaS (Software como Serviço), é um modelo de entrega de software no qual a hospedagem dos programas e dados é centralizada na nuvem. SaaS é tipicamente acessado por clientes através de *web browsers* em máquinas com poucos recursos e provê aplicações à nuvem para serem consumidas sob demanda, onde as aplicações “rodam” no *browser* do usuário, não sendo necessárias instalações e configurações de software. Com a localização do software na Web, os clientes podem acessá-lo a qualquer momento e de qualquer lugar, permitindo, por exemplo, uma maior mobilidade para funcionários de determinadas empresas que precisam constantemente mudar de local de trabalho ou atuam a maior parte do tempo fora das dependências físicas da empresa, o que implica numa maior autonomia e integração entre funcionários e setores de uma mesma empresa. Outra vantagem é redução de custos com instalações, atualizações e aquisição de licenças de software. Alguns exemplos de SaaS são os serviços de CRM (Custom Relationship Management) on-line da Salesforce [20], webmail e o Google Docs [19].

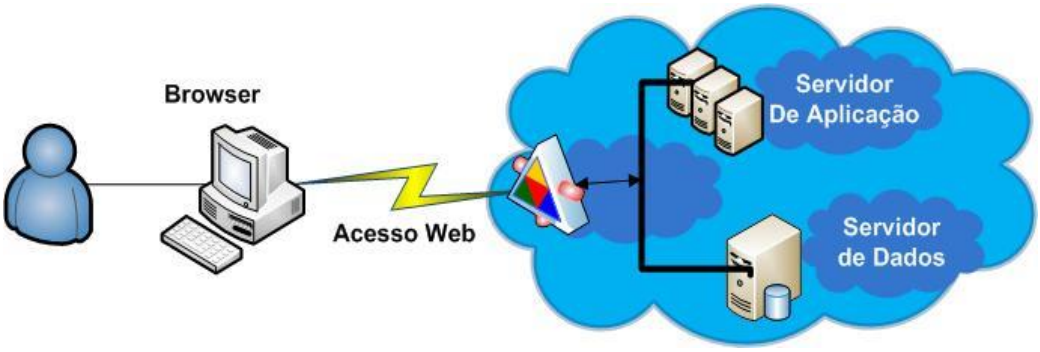



**Figura 2.6:** SaaS-Software as a Service (Software como serviço) adaptado de [81]

### 2.1.1.3 Modelos de Implantação da computação em Nuvem

Os modelos de implantação, dependendo da forma como os recursos são organizados e como estão disponíveis aos usuários, podem ser classificados como:

- Nuvem pública, onde o acesso é disponibilizado para o público em geral, podendo pertencer e ser gerenciada por uma organização privada, acadêmica, governamental, ou qualquer combinação entre elas. São serviços de nuvem prestados por terceiros (fornecedor). Eles existem além do firewall da empresa, e são totalmente hospedados e gerenciados pelo provedor da nuvem.
- Nuvem privada, quando a infraestrutura é provisionada para uso exclusivo de uma única organização. Estas nuvens existem dentro do firewall da empresa e são gerenciadas por ela.
- Nuvem comunitária, possui a infraestrutura provisionada para uso exclusivo de uma comunidade de usuários específica (organizações com interesses comuns).
- Nuvem híbrida, onde a infraestrutura da nuvem é composta por qualquer tipo de combinação entre infraestruturas distintas (privada, pública ou comunitária), tornando-se uma única nuvem.

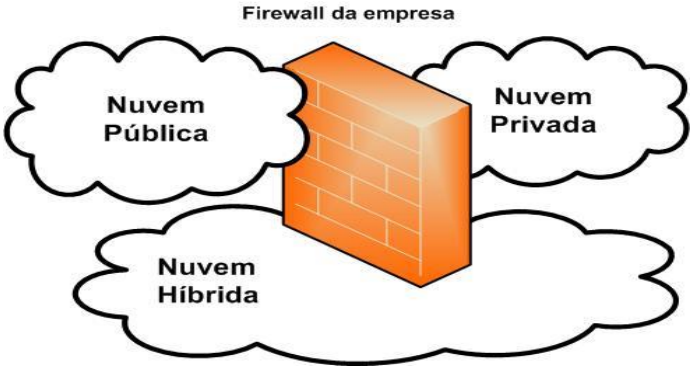



**Figura 2.7:** Nuvem Pública, Nuvem Privada e Nuvem Híbrida

### 2.1.1.4 Papéis na Computação em Nuvem

É de extrema importância a definição de responsabilidades, acesso e perfil para os diferentes tipos de usuários que utilizam ambientes de Computação em Nuvem. Para um melhor entendimento do funcionamento do ambiente de nuvem, os atores do modelo podem ser classificados conforme os papéis que desempenham [21]. Estes papéis são mostrados na Figura 2.8.

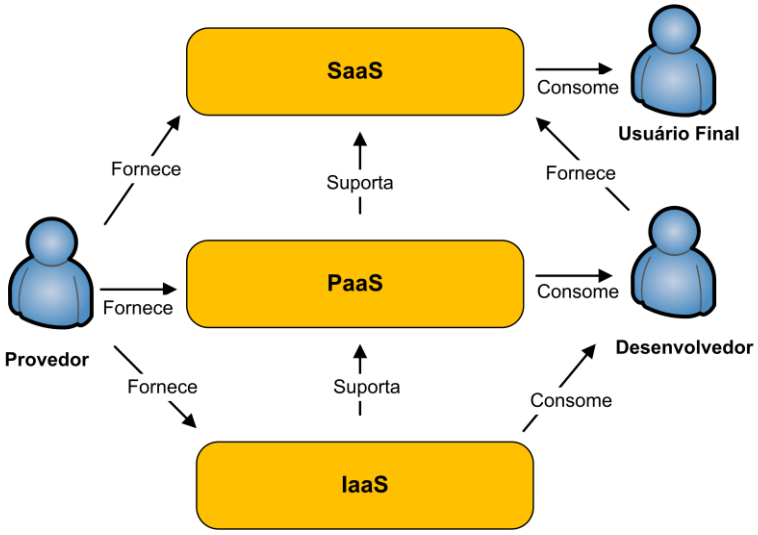



**Figura 2.8:** Papéis na computação em nuvem adaptado de [10]

De acordo com a figura, podemos observar os papéis exercidos pelos atores:

- O provedor: tem por função disponibilizar, gerenciar e monitorar toda a estrutura para a solução de computação em nuvem, abstraindo dos desenvolvedores e usuários finais a complexidade e também a responsabilidade de administrar os recursos e sistemas subjacentes da nuvem. O provedor fornece serviços nos três modelos de serviços;

- O desenvolvedor: utiliza os recursos fornecidos e provê serviços para os usuários finais;

- O usuário final: consome os recursos ofertados pela nuvem.

De acordo com o interesse, os atores podem assumir diferentes papéis ao mesmo tempo, contudo somente o provedor fornece suporte a todos os modelos

de serviços. A organização em papéis auxilia na definição dos atores e de seus diferentes interesses.

### 2.1.2 Desafios na Implantação da computação em nuvem

Como todo novo modelo, a computação em nuvem apresenta alguns desafios a serem enfrentados que podem dificultar a sua adoção, como podemos ver a seguir:

- Segurança e privacidade: a utilização dos serviços de computação em nuvem vem despertando grande interesse nos últimos anos, com isso surge também a preocupação com a segurança destes ambientes. A segurança e a privacidade são os principais desafios a uma ampla adoção do paradigma de computação em nuvem, pois, falhas de segurança em qualquer um dos componentes podem impactar os demais componentes de segurança e consequentemente a segurança de todo o sistema poderá entrar em colapso [22]. A eficiência de mecanismos tradicionais de segurança estão sendo reconsideradas, pois a características do modelo de computação em nuvem podem diferir das outras arquiteturas tradicionais [23]. Mecanismos de proteção precisam ser adaptados para esse novo ambiente;
- Disponibilidade dos serviços: ao mover seus dados e aplicativos para a nuvem, usuários devem ter garantias de acesso a qualquer momento e em qualquer lugar. A nuvem deve proporcionar alta disponibilidade, utilizando-se de redundância e técnicas de balanceamento de carga, visando atender aos requisitos e necessidades dos usuários;
- Gargalo na conexão: quanto mais aplicações são transferidas para ambientes de computação em nuvem, maior será a necessidade de largura de banda para transportar dados entre provedores e consumidores de serviços de nuvem. A infraestrutura de Internet disponível atualmente ainda é insuficiente, porém, quanto mais perto do provedor o consumidor estiver melhor será a qualidade de serviço. Assim, investimentos e estudos em novas tecnologias de rede para melhorar o QoS na conexão de Internet são necessários, pois melhores taxas de transferência de dados e tempos de resposta aceitáveis são essenciais para que a computação em nuvem possa demonstrar um desempenho satisfatório sem ser prejudicada pela infraestrutura da Internet.

Além disso, o protocolo HTTP, devido às limitações do seu modelo baseado em pedido-resposta *stateless* (sem estado), não foi originalmente projetado para algumas interações e tarefas na nuvem. Assim, melhorias no protocolo de comunicação são necessárias [24];

- Padronização: a portabilidade e a interoperabilidade entre nuvens e aplicativos em diversas nuvens podem ser problemáticas devido a grande diversidade de tecnologias envolvidas. Diversas empresas de desenvolvimento de software tem grande interesse nessa portabilidade, contudo, padronizar as diversas interfaces e serviços tem se tornado uma grande desafio.

A grande vantagem ao se utilizar a computação em nuvem é permitir a contratação de novos recursos à medida que estes se tornem necessários, reduzindo custos com infraestrutura e manutenção.

## 2.2 Virtualização

Com o passar dos anos a informática vem evoluindo rapidamente, com um grande aumento no poder de processamento dos computadores. Porém, esse poder de processamento não tem sido utilizado de forma eficiente, ocasionando uma subutilização dos recursos computacionais. Para a resolução desse problema, passou-se a utilizar cada vez mais a tecnologia de virtualização, visando a diminuição da ociosidade de processamento. O uso da virtualização representa a criação de máquinas virtuais que possibilitam que sistemas operacionais executem simultaneamente em um único equipamento físico [25].

A virtualização não é uma tecnologia nova, originou-se na década de 60 com o desenvolvimento do sistema operacional experimental M44/44X pela IBM, introduzindo, assim, a tecnologia de hipervisor em seus mainframes, o que serviu de base para desenvolver vários sistemas comerciais com suporte a virtualização, como o OS/370 [26] [27] [28].

O hipervisor ou monitor de máquina virtual (*Virtual Machine Monitor* - VMM) atua como uma camada de abstração entre o hardware e os diversos sistemas operacionais em execução nas VMs, permitindo particionar um único sistema computacional (chamado hospedeiro) em vários outros (denominados *guest*). Um ambiente de máquina virtual é fornecido pelo hiperisor, também conhecido como “sistema operacional para sistemas operacionais” [29]. Cada máquina virtual oferece

um ambiente completo muito similar a uma máquina física, possuindo seu próprio sistema operacional, aplicativos e serviços de rede, sendo possível interconectar (virtualmente) cada uma dessas máquinas formando uma rede virtual (VN). Uma máquina virtual é uma eficiente e isolada cópia de uma máquina real [30]. O hipervisor disponibiliza uma interface (multiplexando o hardware) idêntica ao hardware subjacente e controla as VMs, abstraindo as complexidades de hardware e de sistema para que as VMs possam executar aplicações ou um "sistema convidado" acreditando ter controle exclusivo de um ambiente convencional com acesso direto ao hardware.

A virtualização proporciona, mesmo que os diversos ambientes virtuais apresentem funcionalidades diferentes [31], características comuns como compatibilidade de software, isolamento, encapsulamento, rápida recuperação, capacidade de migração entre diferentes hosts e execução em múltiplas plataformas [7]. A compatibilidade garante que todo software escrito para uma determinada plataforma possa executar em uma VM que virtualize esta plataforma. O isolamento faz com que processos executando em uma máquina virtual não interfiram em outra máquina virtual ou no monitor de máquinas virtuais. O encapsulamento é a propriedade da virtualização que permite controlar e manipular a execução do software na máquina virtual. A rápida recuperação, que é o retorno de uma VM ao estado de execução após uma pausa ou desligamento, permite a restauração de um estado do sistema na máquina virtual e a migração e execução em múltiplas plataformas são obtidas devido ao monitor de máquinas virtuais ter acesso e controle a todas as informações sobre processos rodando em suas máquinas virtuais.

A aplicação dessas características pode ser bastante útil em sistemas de proteção como Sistemas de Detecção de Intrusão ou IDS (*Intrusion Detection Systems*), possibilitando um isolamento do IDS ou do sistema monitorado e protegendo, assim, o acesso aos níveis de sistema, minimizando os impactos de possíveis ataques.

### 2.2.1 Tipos de virtualização

Existem várias maneiras de implementar a virtualização. Podemos classificar o tipo de virtualização em relação à modificação do sistema operacional convidado em:

- Virtualização total: pode ser definida como uma camada de software que simula os dispositivos de hardware de um sistema computacional. A virtualização total tem o objetivo de fornecer para as máquinas virtuais uma cópia do *hardware* subjacente, de forma que o sistema operacional e as aplicações executem como se tivessem um hardware exclusivo, não percebendo a virtualização. A virtualização realizada dessa forma traz a vantagem de poder utilizar sistemas operacionais convidados sem modificações, porém, pode apresentar algumas inconveniências devido ao sistema convidado poder executar operações em modo privilegiado, gerando resultados incorretos e necessitando de um maior controle pelo hipervisor, sendo que todas as instruções devem ser testadas antes, o que implica em um maior custo de processamento.

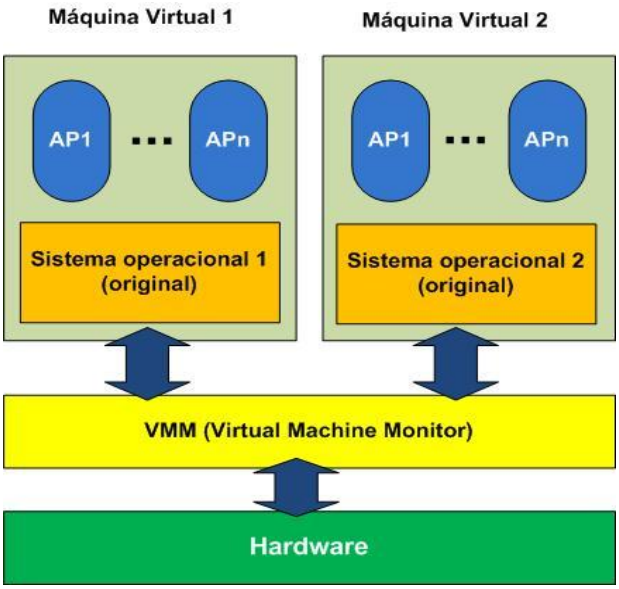



**Figura 2.9:** Virtualização Total

- Para-virtualização: Na Para–virtualização o sistema operacional é modificado para chamar[3] o *hypervisor* sempre que executar uma instrução que possa alterar o estado do sistema. Com isso, as instruções executadas nas máquinas virtuais vão diretamente para o *hardware* original, sem a necessidade da realização de teste pelo *hypervisor,* gerando um ganho significativo no desempenho de processamento. Outro ponto positivo na para-virtualização é que os dispositivos de *hardware* são acessados por *drivers* da própria máquina virtual, sem a necessidade da utilização dos *hardwares*

[3] Emprega-se normalmente o termo hypercall, ou seja, a substituição da chamada de sistema realizado pelo hipervisor.

genéricos, conseguindo utilizar a capacidade total de recursos oferecidos pelos dispositivos. Aparentemente a para-virtualização apresenta um ganho em desempenho significativo frente à virtualização total. Com a implementação de instruções de virtualização nos processadores *Intel* e *AMD* que favorece a virtualização total essa disparidade tem diminuído muito.

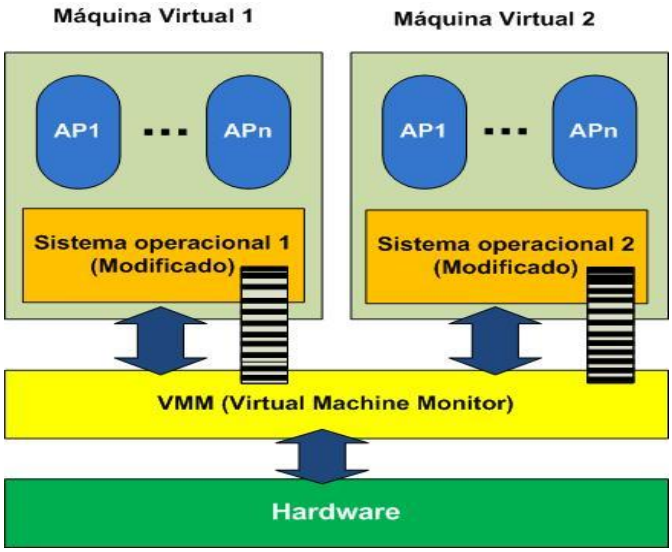



**Figura 2.10:** Para-virtualização

Podemos também classificar a virtualização de acordo com o tipo de hipervisor:

- Hipervisor hospedado ou hipervisor tipo 1: executam diretamente no hardware da máquina real, atuando como controlador de hardware e monitor de sistemas operacionais convidados. Este tipo de hipervisor multiplexa os recursos de hardware de uma forma que cada máquina virtual os vê como um conjunto de recursos próprios, comportando-se como uma máquina física completa capaz de executar seu próprio sistema operacional. Como exemplos temos o VMWare ESXi [32], IBM Power VM [33] e o Xen [34].
- Hipervisor não-hospedado ou hipervisor tipo 2: executam como um processo em um sistema operacional comum e permitem a execução de sistemas operacionais guest. O hipervisor se utiliza dos recursos do sistema operacional subjacente para disponibilizar recursos virtuais aos sistemas convidados em execução. Geralmente um hipervisor tipo 2 suporta apenas uma máquina virtual executando uma instância de sistema operacional convidado, necessitando serem iniciados mais hipervisores caso mais máquinas virtuais sejam necessárias. Alguns exemplos dessa abordagem são o VMWare Workstation [35], Parallels Desktop [36], QEMU [37] e o VirtuaBox [38].

## 2.3 Segurança em Computação em Nuvem

Com o surgimento da Internet, houve um considerável aumento no número de ataques e tentativas de ataques a sistemas computacionais, o que motivou a busca por melhores estratégias de segurança. Ataques, quando bem sucedidos, podem causar grandes prejuízos financeiros e de imagem para empresas, instituições e pessoas físicas.

No cenário atual, o rápido crescimento no campo da computação em nuvem também tem contribuído para aumentar a preocupação com a segurança. A computação em nuvem é rodeada por questões de segurança como a privacidade dos dados armazenados na nuvem, a utilização indevida do recursos da nuvem por usuários maliciosos além da administração dos dados dos usuários ser feita por terceiros. Assim, para uma implantação bem sucedida da computação em nuvem, é preciso um bom gerenciamento da segurança.

A Segurança da Informação é a área responsável por proteger os ativos da informação, ou seja, alterações indevidas, acesso não autorizado e indisponibilidade da mesma. Desta forma, tal área de conhecimento possui três propriedades básicas a serem atingidas [39]:

- Confidencialidade: somente usuários devidamente autorizados podem ter acesso à informação. Logo, ela tem por objetivo proteger o conteúdo e limitar o acesso à informação, ou seja, a permissão de ler ou escrever apenas é dada a quem realmente tem direito ou permissão;
- Integridade: garante que a informação não será destruída ou corrompida, além de o sistema ter um desempenho correto. A informação só pode ser alterada por quem tem direitos, o que garante que ela não seja modificada no percurso da origem ao destino;
- Disponibilidade: os usuários devem poder acessar a informação a qualquer momento, desde que tenha permissão para isso. Os serviços/recursos do sistema devem estar disponíveis sempre que forem necessários.

Existem várias pesquisas em andamento na área de segurança para computação em nuvem. Devido ao grande interesse das organizações e das pessoas em implantar ou utilizar serviços de nuvem, vários grupos e organizações estão desenvolvendo soluções de segurança e padrões para a computação em

nuvem. O *Cloud Security Alliance* (CSA) [4] apresentou o documento intitulado "*Secuity Guidance for Critical Areas of Focus in Cloud Computing*" (Guia de Segurança para Áreas Críticas Focando em Computação em Nuvem) em dezembro de 2009, onde identifica treze áreas mais preocupantes classificadas em três seções principais, que são: seção 1- arquitetura da nuvem; seção 2 – governança na nuvem e seção 3 – operando na nuvem [40]. O ENISA (*European Network and Information Security Agency*) em sua pesquisa intitulada "*Cloud Computing: Benefits, Risks and Recommendations for Information Security*" (Computação em Nuvem: Benefícios, Riscos e Recomendações para Segurança da Informação) enumera oito riscos específicos de computação em nuvem que são: perda de governança, *lock-in*[5], falha de isolamento, risco de conformidade, interface de gerenciamento comprometida, proteção dos dados, dados deletados de forma incompleta ou insegura e colaboradores internos mal intencionados [41]. No relatório chamado "*Assensing the Security Risk of Cloud Computing*" (avaliando os riscos de segurança da computação em nuvem), publicado pelo Gartner em junho de 2008, são identificados sete questões especificas de segurança: acesso de usuário privilegiado, conformidade legal, localização dos dados, segregação dos dados, recuperação, apoio à investigação, e viabilidade a longo prazo [41,42].

Os trabalhos citados são amplamente citados por vários pesquisadores na área de segurança em computação em nuvem e servem como guia para usuários que pretendem usar serviços de um provedor de nuvem. De acordo com os trabalhos, antes de contratar um provedor, os usuários devem conhecer os principais riscos associados à computação em nuvem. O CSA, através dos resultados de suas pesquisas, elaborou em março de 2010 o documento *The Top Threats to Cloud Computing* (Principais Ameaças para Computação em Nuvem), descrevendo sete áreas de segurança críticas consideradas de grande importância para as organizações que usam serviços de nuvem [43]. A seguir veremos mais detalhes desses riscos de segurança.

[4] Cloud Security Alliance (CSA) é uma organização sem fins lucrativos com a missão de promover a utilização das melhores práticas para a prestação de garantia de segurança dentro de Cloud Computing, e promover educação sobre o uso da computação em nuvem auxiliando a proteção de todas as formas de computação.

[5] É o aprisionamento tecnológico resultante da falta de padronização entre provedores de Computação em Nuvem, tornando os clientes dependentes do vendedor de serviços e produtos, impedindo o cliente de trocar de provedor de nuvem sem implicar custos adicionais substanciais.

### 2.3.1 Principais ameaças em computação em nuvem

As sete principais ameaças para Computação em Nuvem são apontadas pelo CSA como sendo:

- **Uso Abusivo e transgressivo da computação em nuvem**

Neste tópico, é relatado o fato de alguns provedores de IaaS não manterem um controle adequado sobre o uso de contas com período de teste gratuito. Essas contas podem ser utilizadas por hackers, spammers e outros tipos de pessoas engajadas em atividades ilícitas, onde geralmente é apenas requisitado para o cadastro um cartão de crédito válido.

Como sugestão para evitar o problema, o CSA recomenda que sejam realizados procedimentos mais rigorosos de registro e checagem de identificação de novos usuários e a utilização de programas de prevenção a possíveis falsificações em cartões de crédito, que são usados para habilitar contas com período de teste gratuito. Outra recomendação é a inspeção do tráfego de rede dos usuários e o monitoramento de blacklist públicas de endereçamentos IP de usuários [43].

Porém, devido às leis de privacidade, pode-se não ter permissão de monitorar o que os usuários estão fazendo. Portanto, mesmo com a recomendação do CSA para um monitoramento avançado, as leis de privacidade restringem ações nesse sentido.

- **APIs inseguras**

Alguns tipos de interfaces de software, também chamadas de APIs, disponibilizadas aos usuários para o gerenciamento ou a interação com os serviços de nuvem, podem expor vulnerabilidades. Essas vulnerabilidades podem ocorrer devido a fragilidades nas APIs ou por serem amigáveis em demasia.

São apontadas como soluções para resolver o problema a análise do modelo de segurança da API, uso de autenticação forte e controle de acesso com encriptação de transmissão [43].

- **Colaboradores internos mal intencionados**

Os empregados de uma empresa provedora de serviços de computação em nuvem podem, em uma atitude maliciosa, obter acesso a dados e serviços confidenciais dos clientes, utilizando do acesso privilegiado que possuem. A preocupação com essa ameaça é agravada com usuários e serviços sob um único

ambiente e também pela falta de transparência dos processos e procedimentos dos provedores. Como exemplo, podemos citar o fato dos provedores não revelarem como controlam os acessos físicos e lógicos ao seus ativos, como monitoram seus funcionários e como é feita a politica interna de segurança.

Tal situação pode proporcionar uma oportunidade para que vários tipos de agentes maliciosos possam obter informações confidenciais ou obter controle sobre serviços na nuvem, com pouco ou nenhum risco de serem detectados.

A sugestão apontada pelo CSA é reforçar a gerência e controle restritivo da cadeia de fornecimento e especificar uma qualificação de recursos humanos como parte do contrato de nível de serviço (SLA[6] – Service Level Agreement). É recomendado também que se tenha transparência nas práticas de segurança da informação e nos relatórios de conformidade, além da verificação dos processos de notificação de falhas de segurança.

- **Problemas relacionados ao compartilhamento tecnológico**

Para proporcionar seus serviços de forma escalável os ambientes de computação em nuvem se utilizam de infraestrutura compartilhada por meio da virtualização. Geralmente a virtualização utiliza um hipervisor para criar máquinas virtuais e sistemas operacionais, mas falhas no hipervisor podem conceder acesso indevido a algum usuário que pode controlar a plataforma impactando os outros usuários.

Como soluções, CSA aponta a implantação de melhores práticas de segurança para instalação e configuração de servidores, monitorar atividades e mudanças não autorizadas no ambiente de nuvem, uso de autenticação forte e controle de acesso para operações administrativas, garantia de cumprimento do acordo de nível de serviço relacionado à correções de vulnerabilidades e varredura de vulnerabilidades e auditoria de configuração.

- **Vazamento ou perda de dados**

Os métodos de economia de custo utilizado pelos provedores de nuvem não podem comprometer os dados dos usuários, pelo menos é o que espera quem

[6] O SLA é um contrato entre um fornecedor de serviços de TI e um cliente especificando, em geral em termos mensuráveis, quais serviços o fornecedor vai prestar. Níveis de serviço são definidos no início de qualquer relação de outsourcing e usados para mensurar e monitorar o desempenho de um fornecedor.

utiliza os serviços de um provedor de nuvem. Existem várias formas de comprometer dados, como excluir ou apagar os dados sem uma cópia de segurança, perder a ligação entre um dado gravado e seu contexto, gravar dados em mídia não confiável e permitir que partes não autorizadas obtenham acesso a dados sensíveis. Em ambientes de computação em nuvem a ameaça de comprometimento de dados sensíveis de usuários é potencializada pelo número de riscos e desafios únicos para esse ambiente, como também pela interação entre riscos e desafios. As características de arquitetura ou de operação da nuvem também comtribuem para essa ameaça.

Um controle forte de acesso em APIs, encriptação e proteção de integridade de dados em trânsito, implantação de práticas como uso de chaves fortes, armazenamento, gerenciamento e destruição de chaves, obrigação contratual do provedor em destruir completamente os dados de um cliente antes de disponibilizar a mídia a outro usuário e a especificação em contrato de estratégias de retenção e backup de dados são os direcionamentos feitos pelo CSA.

- **Sequestro de contas, serviços e tráfego**

O sequestro de contas ou de serviços é geralmente feito através do roubo de senhas e credenciais. Vários tipos de ataques como phishing, fraude e exploração de vulnerabilidades, podem ser utilizados para conseguir credenciais que são reutilizadas, possibilitando a um atacante poder espionar as atividades e transações do verdadeiro dono da credencial, além de manipular dados, retornar informações falsas e redirecionar o usuário legitimo a um site falsificado.

Para remediar o problema, é proposto pelo CSA a proibição do compartilhamento de credenciais de acesso entre usuários e serviços, utilizar fortes técnicas de autenticação quando possível, monitoramento pró-ativo para detecção de atividades não autorizadas, e compreensão das políticas e de segurança e SLAs do provedor de nuvem.

- **Perfil de risco desconhecido**

Os benefícios oferecidos pela computação em nuvem como, redução ou mesmo eliminação de manutenção em hardware e software, redução de custos com hardware e licença, além de outros benefícios permitem que as empresas foquem seus esforços diretamente em seus negócios. Porém, deve haver cautela em adotar

a computação em nuvem, comparando os benefícios com as preocupações de segurança, pois essa adoção não garante a eficácia dos procedimentos de segurança usados pela própria empresa, implicando em riscos desconhecidos. As sugestões do CSA são divulgar registros e dados aplicáveis, divulgação total ou parcial da infraestrutura da nuvem e monitoramento e alerta de informações necessárias.

### 2.3.2 Superfícies de ataque

Para a construção de um cenário de Computação em Nuvem é necessário utilizar três tipos de participantes: usuários de serviços, serviços ou instâncias de serviços e o provedor de nuvem.

Neste cenário, devem ocorrer interações entre os três tipos de participantes, como exemplos de interação podemos citar um usuário requisitando um serviço ou um serviço requerendo mais ciclos de CPU ao sistema de infraestrutura. As interações em um cenário de computação em nuvem ocorrem com a participação de pelo menos duas instâncias, pertencentes a tipos de participantes diferentes.

Da mesma forma, podemos descrever as tentativas de ataque em um cenário de computação em nuvem como sendo um conjunto de interações entre os três tipos de participantes, onde por exemplo, entre um usuário e um serviço temos os mesmos ataques existentes fora do cenário de computação em nuvem, como injeção de SQL e ataques de DoS. Com isso, temos que considerar que na segurança de computação em nuvem, a lista de participantes em um ataque pode incluir qualquer um dos três tipos de participantes do cenário.

Neste cenário, cada um dos três tipos de participantes deve prover uma interface específica para os outros participantes. Por exemplo, o provedor de nuvem deve disponibilizar para cada instância de serviço uma interface específica (API) com a qual o serviço possa interagir. De maneira semelhante, uma instância de serviço deve disponibilizar ao usuário uma interface específica (API). Portanto, cada participante do cenário deve disponibilizar uma interface aos outros participantes, onde podemos considerar ao todo seis interfaces. Na Figura 2.11 podemos ver um detalhamento dessas interfaces que segundo [9], podem ser consideradas com superfícies de ataque.

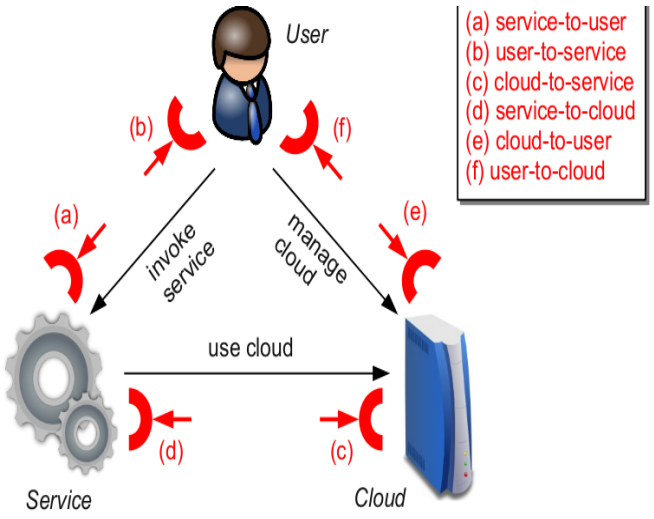



**Figura 2.11:** Superfícies de Ataque em Computação em Nuvem [9]

As superfícies de ataques categorizadas em [9] são:

- Superfície de ataque **a** : é ofertada de uma instância de serviço para um usuário. Pode ser considerada como uma interface comum de servidor para cliente, sendo vulnerável a todos os tipos de ataques a arquitetura cliente/servidor como *buffer overflow* e SQL *injection*;
- Superfície de ataque **b** : é ofertada de um usuário para uma instância de serviço. Como exemplo desse tipo de interface temos os ambientes que os programas de usuários proveem ao servidor (navegadores web), o que a torna alvo de ataques baseados em navegadores. Como exemplos de ataques baseados em navegadores web temos SSL *certificate spoofing*, ataques a browser cache e ataques de *phishing* no e-mail de usuários;
- Superfície de ataque **c:** é a interface situada entre a instância de serviço e o sistema de nuvem. Provê os recursos do sistema de nuvem ao serviço, sendo que esta superfície cobre todos o tipos de ataques que um serviço pode lançar contra o sistema de nuvem que o hospeda. Como exemplo de ataques a esta superfície, podemos citar os ataques que levam à exaustão de recursos como denial of service e ataques ao hipervisor;
- Superfície de ataque **d:** é a superfície ofertada ao sistema de nuvem pela instância de serviço. Essa superfície pode sofrer todos os tipos de ataques que um provedor de nuvem possa realizar contra um serviço que esteja executando em sua infraestrutura, como por exemplo a redução de recursos para derrubar um serviço em execução. Outra forma de ataque

pode ser relacionado à privacidade através da exploração dos dados de um serviço em processamento ou por interferência maliciosa adulterando dados sendo processados;

- Superfície de ataque **e** : é a interface provida ao usuário pelo provedor de nuvem. Tem a função de possibilitar ao usuário o controle dos serviços ofertado pelo provedor. Como exemplo de ataque a esta superfície temos o *impersonating attack*;
- Superfície de ataque **f** : é a interface ofertada pelo usuário ao provedor de nuvem. Como ataques temos ataques de pishing onde é oferecido ao usuário um falso site de pagamento do provedor, possibilitando o controle dos serviços do provedor e direcionando o custo ao usuário. Este ataque é conhecido com ataque de *billing*.

O nosso trabalho concentra-se em cobrir ataques na superfície **e,** entre o usuário e o provedor de nuvem. Dessa forma, podemos evitar também ataques em outras superfícies como na superfície **d**, onde o alvo são os serviços hospedados nos provedores de nuvem.

## 2.4 Sistemas de detecção de intrusão - IDS

Com a popularização dos microcomputadores nas décadas de 80 e 90 e o desenvolvimento e amplo acesso às redes de computadores e à Internet, cada vez mais os recursos computacionais passaram a fazer parte das empresas e da vida das pessoas, facilitando tarefas e melhorando a produtividade. No entanto, com o avanço das redes de computadores e da Internet surgiram inúmeras ameaças virtuais que têm como alvo a informação, que passa a ser o principal ativo de muitas organizações. Por este motivo é necessário implantar mecanismos de proteção, pois a cada dia surgem novas ferramentas automatizadas ou código-fonte de explorações de falhas em sistemas para promover acesso a servidores de rede, comprometendo sua segurança.

O sistema de detecção de intrusão ou IDS (*Intrusion Detection System*) tem como objetivo melhorar a segurança em uma rede de computadores. O IDS engloba processos de monitoramento, identificação e notificação de ocorrências de atividades suspeitas ou maliciosas.

O IDS tenta reconhecer um comportamento ou uma ação intrusiva para alertar um administrador ou automaticamente disparar contramedidas [44], atuando junto ao

sistema operacional ou em uma rede de computadores buscando identificar atividades maliciosas. Atua também como ferramenta de segurança que, assim como outras medidas, tais como antivírus, firewalls e sistemas de controle de acesso, se destinam a reforçar a segurança dos sistemas de informação e comunicação [45].

Um sistema de detecção de intrusão é formado basicamente por três componentes que possuem as funções de coletar, analisar e mostrar informações sobre o sistema [46]. Esses componentes são:

- Sensor: possui a função de capturar informações do sistema;
- Analisador: é o principal componente do IDS, seu objetivo é analisar os dados obtidos pelo sensor para, por meio da análise, alertar se houve ou não uma intrusão;
- Interface com o usuário: apresenta de forma estruturada os dados coletados e analisados para o administrador.

Um outro exemplo de funcionamento de um IDS pode ser representado pelo modelo do *Common Intrusion Detection Framework* – CIDF [47], que mostra o fluxo de informações e as suas funcionalidades básicas, como podemos ver na Figura 2.12.

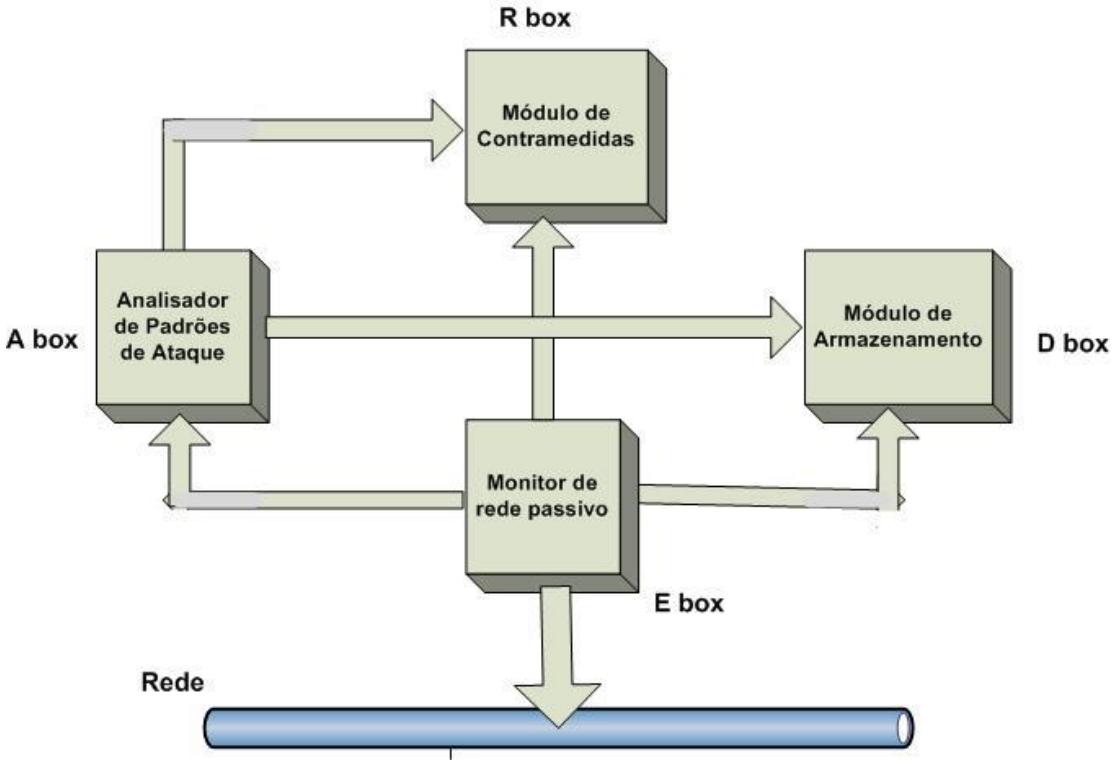



**Figura 2.12:** *Common Intrusion Detection Framework* com capturador de pacotes na rede [47].

Os componentes que formam o CIDF são:

- *E boxes* (*event generators* - geradores de eventos) – sensores que tem a finalidade de capturar eventos e fornecer informações sobre os mesmos para o resto do sistema. São considerados como eventos a presença de um pacote transitando na rede, o registro de alguma atividade no log, etc.;
- *A boxes (analysers* - analisadores) – mecanismo que analisa as informações coletadas pelos *E boxes* sobre os eventos. O A boxe verifica se algo suspeito está em andamento;
- *D boxes* (*database components* – mecanismos de armazenamento) – tem a função de manter as informações provenientes dos geradores de eventos e dos mecanismos de análise para que estejam disponíveis aos operadores do sistema no futuro, como, por exemplo, dados de um atacante reincidente;
- *R boxes (response boxes* – módulos de resposta) – módulo de contramedida responsável por efetuar uma ação em resposta a uma falha de segurança detectada. Por exemplo, interromper uma conexão TCP (*Transmission Control Protocol)*.

### 2.4.1 Classificação

De acordo com os esquemas de classificação de IDS em [5], [48], [49] e [8], as técnicas usadas para detecção de intrusos podem ser classificadas da seguinte forma:

**Quanto à arquitetura**

- Centralizada: composta por elementos sensores e analisadores, no qual a tarefa principal é concentrada em um único elemento, apresentando assim o inconveniente de um ponto único de falhas. Entretanto, como ponto positivo, proporcionam uma administração facilitada;
- Hierárquica: formada por camadas, onde as camadas superiores delegam tarefas a camadas subordinadas a elas. Podendo ser organizadas por uma

camada de captura de dados, uma camada de análise e por um coordenador responsável por todo o IDS. A falha em uma camada pode comprometer todo o sistema.

- Distribuída: possuem componentes do IDS que cooperam entre si para efetuarem a detecção, sem a presença de um gerenciador central.

**Quanto ao método de detecção**

- Detecção baseada em assinaturas: identifica padrões de ataques conhecidos para encontrar possíveis tentativas de invasão [50]. Estas assinaturas são formadas por um conjunto de regras que caracterizam o intruso, possuindo a vantagem de uma detecção imediata e praticamente impossibilitando a ocorrência de falsos positivos.
- Detecção baseada em anomalia: classifica atividades fora de um padrão considerado normal da rede, como as anomalias do tráfego de rede [45] [50], e comportamento anormal do sistema, identificando atividades suspeitas por apresentarem desvios de comportamento, o que poderia indicar a presença de atividades maliciosas. A vantagem deste método está na detecção de ameaças desconhecidas, no entanto, pode produzir uma alta taxa de falsos positivos, devido ao comportamento imprevisível dos usuários.
- Híbrida: utiliza as duas abordagens anteriores, ampliando, assim, a possibilidade de sucesso na detecção de ataques.

**Quanto ao local de coleta de dados**

- Sistemas de detecção baseado em rede ou NIDS (Network-based Intrusion Detection): capturam dados do tráfego de rede para análise, procurando por assinaturas de ataques conhecidos e anomalias nas atividades em hosts monitorados na rede. Ele se localiza em um ponto com visibilidade total sobre o tráfego de rede a monitorar.
- Sistemas de detecção baseado em host ou HIDS (Host-based Intrusion Detection): monitoram a atividade local do host, utilizando os arquivos de registro de eventos do sistema operacional como base de dados para serem analisados. O objetivo é identificar ataques e tentativas de acesso indevido à própria máquina. Neste caso, o IDS localiza-se na máquina a monitorar.

- Híbridos: utilizam a combinação de sensores que capturam o tráfego de rede e de sensores que capturam os registros de eventos do *host* para serem analisados aumentando a capacidade de detecção. Segundo [51], a maior parte dos sistemas de detecção utiliza a análise baseada em rede e em *host*.

**Quanto ao comportamento ao detectar ações intrusivas**

- Passivo: O IDS apenas alerta o administrador sobre a intrusão, sem que nenhuma ação seja tomada;
- Ativo: O IDS, ao detectar uma ação intrusiva, efetua ações para proteger o sistema.

## 2.5 Trabalhos Relacionados

Um sistema de detecção de intrusão deve estar disponível em qualquer local da nuvem, monitorando os seus diversos nós físicos e virtualizados. Como a nuvem possui um ambiente distribuído, o ideal é que, para uma eficaz proteção, o IDS também seja distribuído e utilize a virtualização para acompanhar a elasticidade da nuvem. Com o objetivo de se adequar às características da computação em nuvem, alguns trabalhos foram propostos, como em [52], apresentando uma arquitetura que provê a segurança através de três tipos de serviços: serviço de segurança de VM, serviço de segurança de rede virtual e serviço de gerência de política de controle de acesso.

No trabalho de Ando et. al. [53] é proposto um analisador de *log* em ambientes de computação em nuvem usando raciocínio automático/automatizado. Em provedores de computação em nuvem, o monitoramento da VM (V*irtual Machine*) é importante para detectar incidentes de segurança. Na descrição do trabalho é apresentada a forma de como monitorar as máquinas virtuais e como o raciocínio automático/automatizado se torna mais eficiente na recuperação de uma grande quantidade de informações de *log*.

A virtualização é utilizada para solucionar vulnerabilidades no hipervisor Xen em [54], onde é apresentado um sistema de monitoramento e detecção de integridade para gerenciar máquinas virtuais em ambientes distribuídos, tendo seu foco em garantir a confiabilidade das VMs de gerenciamento que manipulam diretamente o hardware.

Em [55], é proposta uma arquitetura para proteger o VMM (V*irtual Machine Monitor*), atuando como uma camada de filtragem ao examinar as chamadas de sistema feitas pelo hipervisor.

Em [56], é proposto um sistema de detecção de intrusão para ambientes de computação em grade e em nuvem. O sistema é implementado ao nível de middleware da nuvem, ficando protegido de intrusos pelas características de isolamento da virtualização. O IDS possui uma arquitetura distribuída, de forma que cada nó do ambiente de nuvem é monitorado por uma parte do sistema de detecção de intrusão, e na ocorrência de um ataque, um alerta será enviado para outros nós do ambiente.

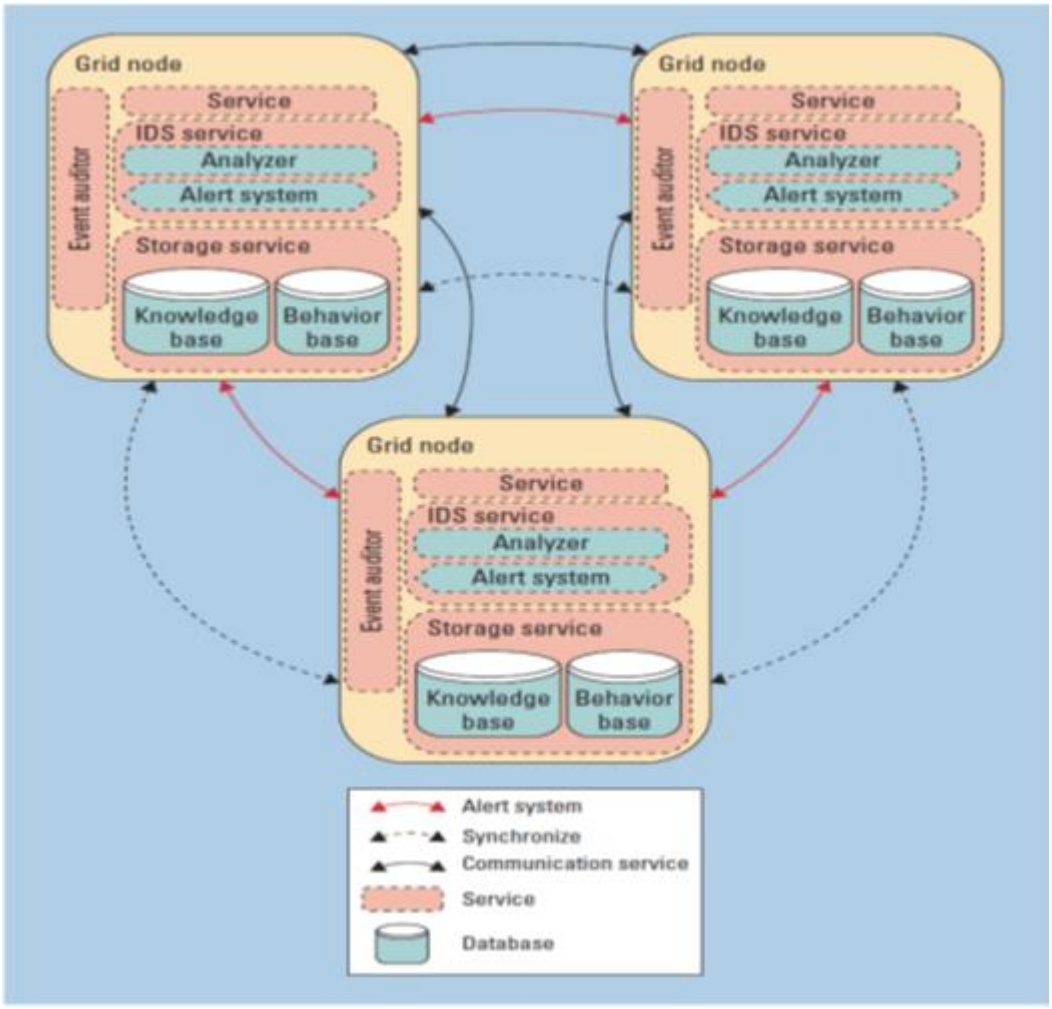



**Figura 2.13:** Arquitetura do Grid and Cloud Computing Intrusion Detection System (GCCIDS) [56]

A Figura 2.13 detalha o compartilhamento de informações entre o componente IDS service e os outros elementos participantes da arquitetura, que são:

- *Node*: recursos que são acessados de uma maneira homogênea através do *middleware*.
- *Service*: provê sua funcionalidade no ambiente por meio do *middleware*, facilitando a comunicação.
- *Event Auditor*: é um elemento chave no sistema. Captura dados de várias fontes, como *log* de sistemas, serviços e mensagens provenientes dos nós.
- *Storage Service*: Armazena os dados que o *IDS service* deve analisar. É importante por garantir que todos os nós tenham acesso aos mesmos dados.

O sistema captura informações de auditoria de "*logs*" e oferece um *middleware* como parte de comunicação de segurança para cada nó e também para o ambiente de grade e nuvem como um todo.

Para obter segurança, duas técnicas são utilizadas: a detecção de intrusão baseada em comportamento, obtida por meio de redes neurais artificiais do tipo "feed-forward"; e a detecção de intrusão baseada em conhecimento, para identificar ataques conhecidos.

No trabalho proposto em [57], os arquivos de *log* dos sistemas de detecção de intrusão são analisados usando algoritmos com Ha*doop M*apReduce, proporcionando relatórios eficientes para os administradores entenderem ataques rapidamente.

Em [58], é proposta uma arquitetura para detecção de intrusão em ambientes de computação em nuvem. A solução proposta cria instâncias separadas de IDS para cada usuário e utiliza um controlador único para gerenciar as instâncias. Nesta arquitetura de IDS, tanto assinatura quanto aprendizagem podem ser utilizados como métodos de detecção. A figura 2.14 apresenta a arquitetura proposta.

Na arquitetura proposta, o IDS é composto por instâncias classificadas como "mini IDS" criadas por um controlador central e único chamado de IDS Controller. As instâncias são implantadas entre cada usuário e o provedor de serviços da nuvem. Segundo os autores, a principal vantagem é a redução da carga de trabalho, pois a mesma será dividida entre as várias instâncias de IDS que estarão habilitadas a realizar seu trabalho de uma melhor maneira do que em um IDS único para toda a nuvem. Sempre que algum usuário quiser acessar serviços da nuvem, uma instância

de IDS será fornecida pelo IDS Controller, sendo responsabilidade do controlador criar instâncias de IDS para cada usuário.

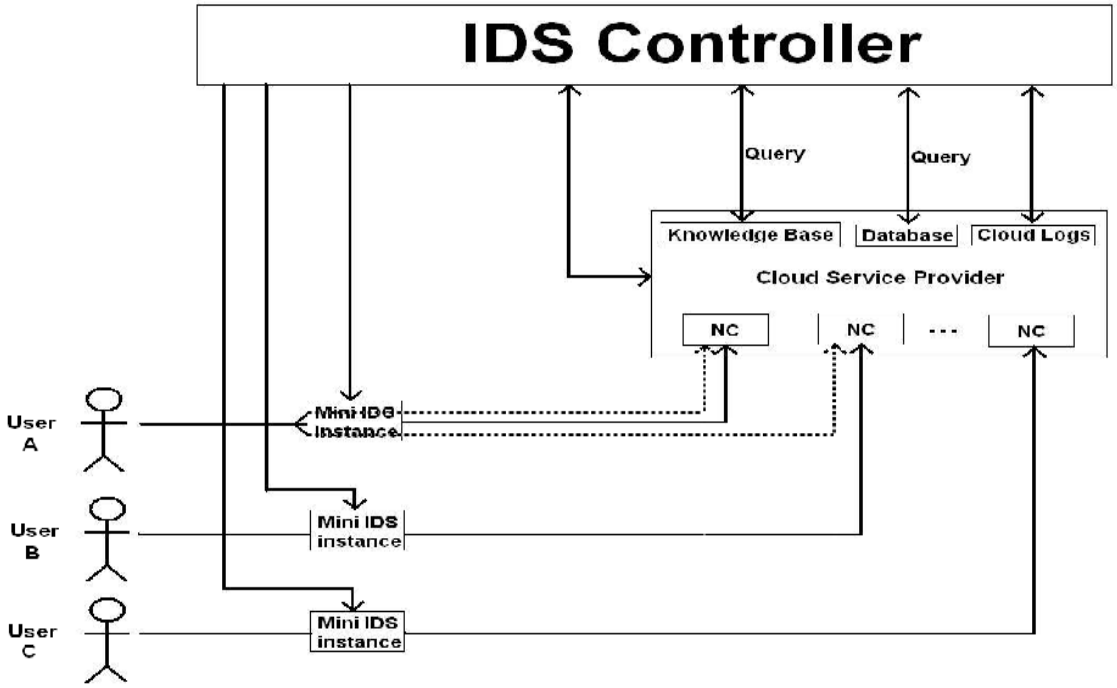



**Figura 2.14:** Intrusion Detection System in Cloud Computing Environment [58]

Portanto, todas as atividades do usuário serão monitoradas pelos mini IDS que, ao término da sessão do usuário, enviará um registro para o controlador de IDS com todas as atividades realizadas pelo usuário. O controlador, por sua vez, armazena esses registros na nuvem, sendo que na próxima vez que o usuário iniciar uma sessão, o IDS controller perguntará à base de conhecimento informações a respeito daquele usuário. A base de conhecimento é armazenada na nuvem e contém informações sobre os padrões das atividades dos usuários baseado em informações contidas nos log da nuvem. Esta base de conhecimento pode utilizar redes neurais ou pode ser estática. Sempre que uma instância de IDS é fornecida para um usuário particular, informações sobre suas atividades anteriores são requeridas pelo IDS controller à base de conhecimento. Esse padrão de atividades pode ser usado para detectar qualquer intrusão a partir do padrão do usuário, com isso é possível aplicar diferentes regras para diferentes usuários. Na arquitetura há um relacionamento de um-para-um entre usuário e instância de IDS e muitos-para-muitos entre instância de IDS e node controler. A arquitetura inclui mais alguns termos que são Agents, Directors e Notifiers. Informações provenientes de fontes de dados como arquivos

de registro, processos, rede, etc, são capturadas pelos Agensts que, localizados dentro das instancias de IDS, tem função de envia-las aos Directors. Os Directors, localizados no IDS controller, fazem a análise das informações, a qual determina se um ataque está acontecendo. Em caso da possibilidade de um ataque o Notifier toma as ações necessárias.

O trabalho realizado em [59] apresenta uma arquitetura extensível de IDS que consiste em vários sensores inseridos em máquinas virtuais controlados por um componente central que analisa os resultados coletados. Diferentes sensores de IDS podem ser utilizados, sendo que a troca de mensagens entre eles é feita com o padrão IDMEF (Intrusion Detection Message Exchange Format). A arquitetura é baseada nas características das máquinas virtuais, como isolamento e rápida recuperação em caso de comprometimento, e também utiliza o padrão IDMEF para a troca de mensagens entre os sensores e a unidade central de gerenciamento. A utilização do IDMEF proporciona uma padronização nas informações de alerta e ainda possibilita a utilização de diferentes sensores de IDS.

O IDS VM management system é formado por vários IDS sensor VMs e por uma unidade central de gerenciamento chamado de IDS Management Unit. Cada componente IDS sensor VM é composto por sensores de IDS inseridos em máquinas virtuais. Usuários podem controlar o gerenciamento do IDS interagindo diretamente e configurando os componentes principais. A estrutura do IDS VM management system é formado pelos seguintes componentes:

- As máquinas virtuais, chamadas de VM IDS, que hospedam os sensores de IDS e o Event Gatherer.

- A unidade central de gerenciamento que controla as VM IDS e é formada por quatro componentes: Event Gatherer, Event Database, Analysis Componente e IDS Remote Controler.

Na figura 2.15 podemos ver os elementos dessa estrutura. Os sensores de IDS são processos independentes que rodam dentro das VMs, podendo ser tanto NIDS quanto HIDS, e identificam comportamentos maliciosos produzindo alertas que são transmitidos à unidade central de gerenciamento através de um componente de alerta conectado em sua saída de dados chamado Event Gather.

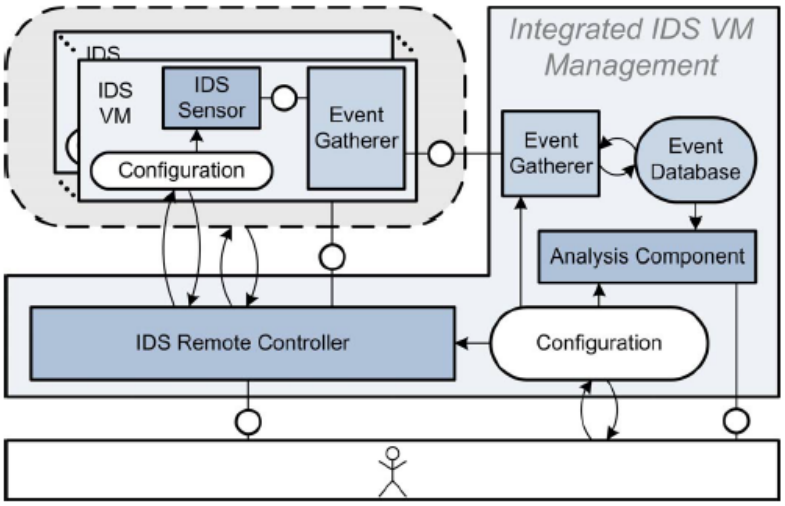



**Figura 2.15:** IDS Management Architecture [59]

O Event Gather está presente tanto no lado do sensor quanto na unidade de gerenciamento, tendo a função de coletar todos os eventos vindos dos diversos tipos de sensores e padronizar as diversas mensagens de saída para IDMEF, além de realizar a comunicação lógica entre os sensores e a unidade de gerenciamento. O gatherer consiste de vários plug-ins:

- Receivers: são usados para a leitura dos alertas e depois os converte para o padrão IDMEF.
- Senders: usados para escrever alertas para o destino, como a rede, uma base de dados ou um arquivo.
- Handlers: são utilizados para modificar alertas em processamento.

A figura 2.16 mostra os componentes do gather.

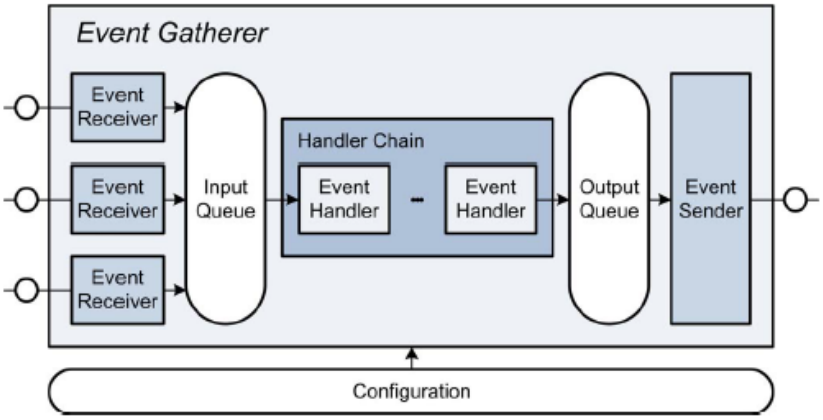



**Figura 2.16:** Even Gatherer [59]

O componente Event Database é uma base de dados passiva que armazena informações sobre todos os eventos recebidos e pode ser acessado através do Analysis Component. Cada evento é armazenado permanentemente na base Event

Database. Um gatherer pode ser uma instancia em execução de um componente de gerenciamento de IDS que aceite conexões e escreva eventos em uma base de dados.

O componente Analysis tem a função de representar os eventos recuperados e analisa-los, correlacionando eventos isolados que possam compor cenários de ataques mais complexos, portanto tendo acesso à base Event database.

O IDS Remote Controller controla e configura remotamente todos os sensores de IDS conectados, tendo acesso a cada uma das configurações dos sensores e podendo ser controlado e configurado pelo usuário. O controlador remoto pode ser um software de monitoramento e controle remoto. Para gerenciar os IDS nas maquinas virtuais o controlador remoto pode comunicar-se com as IDS VMs e seus sensores encapsulados. Três importantes características de gerenciamento das máquinas virtuais, como controle de VM, monitoramento de VM e configuração de VM, são integradas internamente ao controlador remoto que possui as operações de: ligar, desligar, parar, resumir, recuperar e atualizar VMs, podendo também ler e modificar as configurações dos sensores dos IDS. Através do sistema é possível visualizar informações de seus sensores, e também de seu ambiente virtualizado como os status das máquinas virtuais (se uma VM está em execução ou não, qual a carga de processamento da VM, e outras informações). Vários parâmetros das IDS VM em execução podem ser configurados diretamente através gerenciamento de IDS, tais como sistema básico de arquivos, espaço em disco utilizável, ou memória. A recuperação de VMs é usada como mecanismo de prevenção de ataques para VMs comprometidas. Outra possível prevenção de ataque pode ser uma parada temporária de VM.

A proposta de [60] é disponibilizar um serviço escalável e ajustável aos usuários da nuvem, provendo a habilidade de monitorar e reagir a ataques em várias VMs existentes dentro da rede virtual privada dos usuários. Para tanto IDSaaS (Intrusion Detection System as a Service) foi implementado utilizando o serviço EC2(Elastic Cloud Compute) da Amazon web service [17]. IDSaaS cria um ambiente de rede virtual dos usuários através do serviço VPC (Virtual Private Cloud) da Amazon, onde instancias de VM do tipo EC2 são criadas para armazenar e executar os seus componentes visando a segurança a nível de infraestrutura (IaaS) através da disponibilidade de mecanismos de detecção totalmente controlados pelos usuários. IDSaaS é um IDS com o modelo detecção baseado em assinatura e usa a rede

como fonte de coleta de dados. É escalável, portável, sob demanda, controlado pelo usuário e disponibilizado através do modelo de pagamento por uso. Tem como alvo o nível de infraestrutura da nuvem, onde sua primeira tarefa é monitora e registrar atividades suspeitas no tráfego de rede entre máquinas virtuais dentro de uma rede virtual pré-definida na nuvem pública. A figura 2.17 mostra o layout do IDSaaS que tem seus componentes construídos e empacotados no formato AMI (Amazon Machine Images). O serviço VPC é utilizado para criar duas sub-redes, uma pública, onde residem as VMs com o IDSaaS, e uma privada que mantém as aplicações de negócios protegidas.

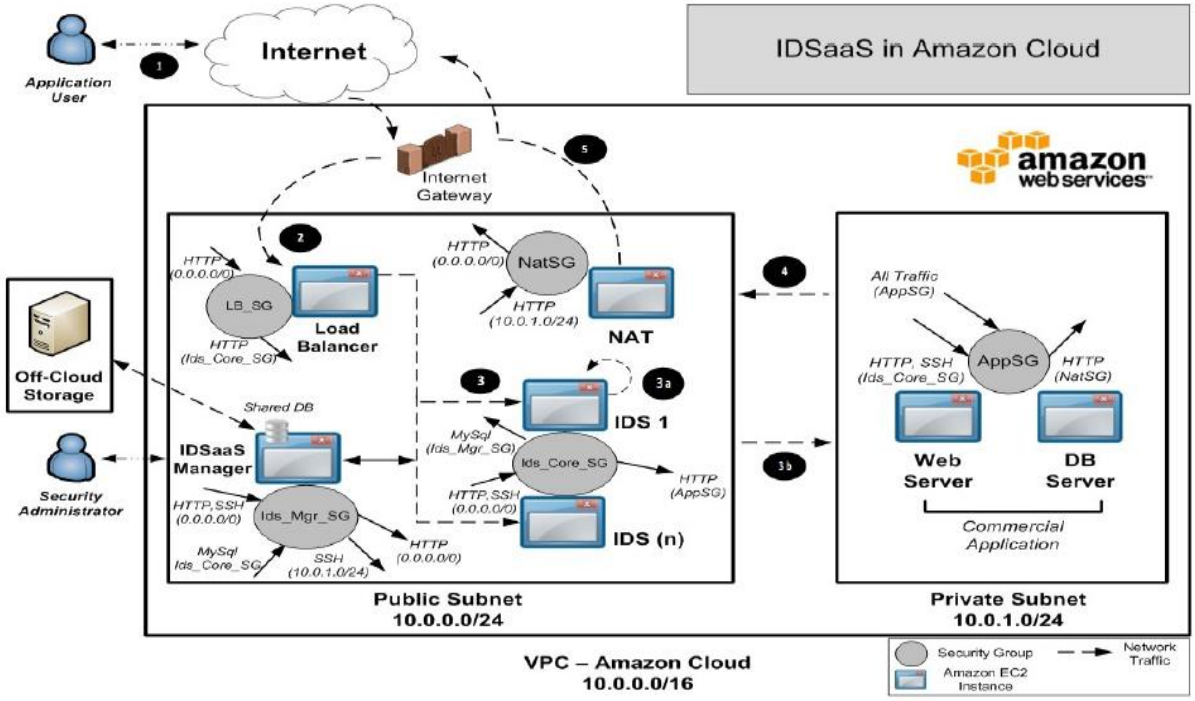



**Figura 2.17:** IDSaaS: Intrusion Detection System as a Service in Public Clouds [60]

As VMs contendo IDSaaS podem ser de três tipos: IDSaaS Manager, IDSaaS Core e LoadBalancer. IDSaaS Manager é o ponto de acesso do administrador de segurança, onde várias tarefas de monitoramento podem ser feitas como configurar outras VMs tanto na sub-rede pública quanto na privada. O IDSaaS Manager contem a base de dados Event Database. IDSaaS Core é a porta de acesso para as VMs contendo aplicações de negócio na sub-rede privada, inspeciona todo tráfego usando o mecanismo de detecção de intrusão, além disso outras réplicas podem ser criadas para distribuir o tráfego evitando pontos únicos de falha. LoadBalancer aumenta a disponibilidade do sistema IDSaaS na nuvem fazendo o balanceamento de tráfego entre as várias VMs executando o IDS core.

Um framework para IDS baseado em nuvem é proposto em [61]. O objetivo é lidar com ataques como masquerade attacks (onde ameaças se fazem passar por usuários legítimos), Host-based attacks (que podem ser uma consequência de masquerade attacks) e Network-based attacks. CIDS também resume os intensivos alertas de IDS de rede, enviado relatórios resumidos ao administrador da nuvem. A solução atua no nível do middleware da nuvem utilizando-se de seus mecanismos, como por exemplo, o sistema de troca de mensagens e introspecção de memória. CIDS monitora e protege o ambiente de execução dos usuários (que residem em máquinas virtuais), além de manter seus próprios componentes protegidos de ameaças que possam afetar as VMs, pois os componentes do CIDS estão localizados fora das máquinas virtuais. Essa proteção é possível graças a característica de isolamento das VMs.

CIDS é escalável, sem coordenador central e usa uma solução P2P. Sua arquitetura distribui a carga de processamento por vários locais da nuvem. As tarefas de usuários são isoladas através da execução em VMs monitoradas. CIDS utiliza tanto uma base de dados de conhecimento, quanto uma base de dados de comportamento como técnicas de análise, além de coletar eventos e auditar informações das VMs para aumentar a cobertura de ataques.

A figura 2.18 mostra a arquitetura do framework onde cada nó inclui um sistema de auditoria que monitora mensagens entre nós e o log de sistema do middleware. Cada nó possui também uma base de dados com assinaturas de ataques conhecidos (conhecimento) e outra com padrões de comportamento anormal (comportamento).

Também são coletados logs e eventos das VMs. Através da troca de informações das bases de dados de conhecimento e comportamento pelos componentes de auditoria presentes em cada nó, CIDS pode detectar usuários não legítimos (designados de masquereders) que acessem a partir de vários nós e também detectar ataques de host e rede.

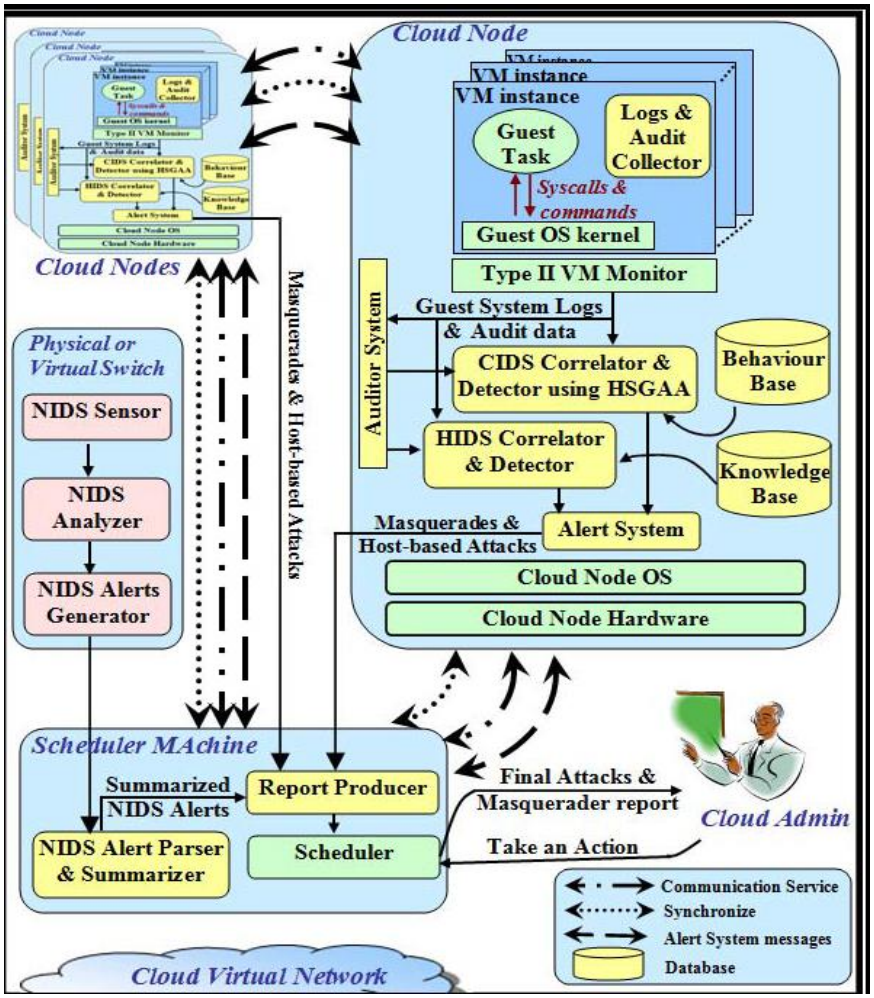



**Figura 2.18:** Cloud Intrusion Detection System [61]

Os componentes do framework CIDS são:

- Cloud node: contém os recursos que são acessados homogeneamente através do middleware da nuvem.
- Guest task: sequencia de ações e comandos submetidos pelo usuário a uma instância de VM.
- Logs & audit collector: localizado dentro da VM atua como um sensor tanto para o detector do CIDS quanto para o detector do HIDS. Coleta os logs, dados auditados e sequências de ações e comandos de usuários.
- Instancia de VM: encapsula o sistema a ser monitorado usando VMM (Virtual Machine Monitor). Os mecanismos de detecção são implementados fora da VM, ficando fora do alcance de invasores. Uma única instância de VM de monitoramento pode observar várias VMs.
- Type II VMM: CIDS usa VMM do tipo II, implementada como um processo no sistema operacional hospedeiro na máquina física.

- The audit System: implementa três funções principais que são monitorar mensagens trocadas entre os nós extraindo delas o comportamento dos usuários; monitorar o sistema de logging do middleware no próprio host e coletar todos os dados auditados e eventos do middleware, como login de usuário ou tarefas submetidas.
- CIDS correlator and detector: correlaciona comportamento dos usuários e os analisa de acordo com uma heurística própria.
- HIDS correlator and detector: correlaciona log de usuários e assinaturas coletadas de várias fontes.
- Behavior-based database: base de dados de perfis e históricos de usuários da nuvem.
- Knowledge-based database: base de dados de regras e assinaturas de ataques conhecidos.
- Alert System: usa os mecanismos de comunicação do middleware para alertar outros nós no caso dos componentes de detecção (CIDS ou HIDS) encontrarem algum padrão de ataque.
- Parser and Summarizer: sumariza os alertas enviados pelo NIDS.
- Report producer: coleta alertas de qualquer IDS da nuvem e envia um relatório sobre os ataques ao escalonador da nuvem.

Cada nó físico tem dois detectores de IDS um CIDS e um HIDS, onde cada nó participa na detecção da intrusão identificando eventos locais que poderiam representar violação de segurança e trocando seus dados auditados com outros nós. Por ter seu próprio analisador e detector, os nós realizam localmente suas tarefas de análise e detecção reduzindo a troca de informações entre os vários nós, diminuindo a complexidade de analisar dados provenientes de vários locais. Isso, no entanto, aumenta a sobrecarga de processamento dentro dos nós. Essa sobrecarga é reduzida como o uso da abordagem HSGAA (Heuristic Semi-Global Alignment Approach) que detecta masquerades attacks baseado no algoritmo de alinhamento Semi-Global de Smith Waterman [62].

A tabela I apresenta uma análise comparativa relacionada às características relevantes das principais arquiteturas de IDS apresentados nesta seção. Esta comparação leva em conta a arquitetura do sistema proposto, bem como se as

características de VM são utilizadas na detecção de intrusos ou para proteção do sistema.

| Características | Arquitetura Distribuída | Arquitetura Centralisada |
|---|---|---|
| Uso das características das VM para detecção de intrusão | [52] [61] [56] | [58] [59] [60] |
| Uso das características das VM para proteção do IDS | [61] [56] | |
| Sensores de IDS inseridos na VM | [52] | [58] [59] [60] |
| Sensores de IDS localizados fora das VM | [61] [56] | |
| Uso da Virtualização para proteger ou corrigir problemas no hipervisor | | [54] [55] |

**Tabela I:** IDS baseados em VM para Computação em Nuvem

As soluções propostas por [58] e [59] utilizam uma arquitetura centralizada, preocupando-se com eventuais gargalos ocasionados pelo processamento das informações coletadas em diversos nós. Para evitar a queda de desempenho da nuvem, a análise das informações é feita localmente, diminuindo o envio de informações ao elemento central, que se encarrega de analisar um conjunto reduzido de dados ou apenas apresentar alertas ao administrador do sistema.

A disponibilidade aos usuários da nuvem de um IDS como serviço é proposto em [60], onde é utilizada a estrutura da Amazon, com os serviços EC2 e VPC para criar as VMs e redes virtuais usadas na arquitetura. A estrutura de uma nuvem pública facilita na construção de um IDS para nuvem, porém as aplicações e dados de usuários protegidos pelo IDS ficam expostas a um maior número de ameaças e também a possíveis vulnerabilidades ainda não encontradas.

Outra possibilidade é apresentada em [61] onde uma arquitetura distribuída sem elemento central é proposta, balanceando a carga de trabalho pelos nós da nuvem e evitando um ponto único de falha por não ter elemento central, entretanto, a troca

constante de informações entre os nós para manter a consistências das bases de dados locais, pode reduzir o desempenho do sistema.

## 2.6 Síntese do capítulo

Neste capítulo, foram apresentadas as seções sobre Computação em Nuvem, virtualização, segurança em Computação em nuvem, sistemas de detecção de intrusão e trabalhos relacionados.

Os principais conceitos, características e modelos de implantação na Computação em Nuvem foram abordados. Na seção sobre virtualização foi mostrado a importância na Computação em Nuvem, características e os principais tipos de virtualização utilizados. Na seção de segurança em Computação em Nuvem foram apresentados os principais estudos realizados sobre as principais ameaças em Computação em Nuvem e também sobre superfícies de ataque na nuvem. Na seção de sistemas de detecção de intrusão, as características e a classificação desses sistemas foram apresentadas.

E por fim, os principais trabalhos realizados na área de sistemas de detecção de intrusão para Computação em Nuvem foram apresentados, onde podemos observar, apesar de alguns trabalhos utilizarem a característica de elasticidade no mecanismo de detecção de intrusão, a ausência de um mecanismo que monitore constantemente os seus recursos da nuvem, possibilitando que o IDS acompanhe a expansão ou a redução de recursos disponibilizados pela nuvem. Com isso, os recursos da nuvem poderiam ser poupadas quando uma demanda baixa de recursos fosse requisitada à nuvem, tornado o sistema de detecção mais eficiente.

# 3 Modelo Proposto

Este capítulo apresenta a proposta de uma arquitetura para um sistema de detecção de intrusão para ambientes de computação em nuvem, tendo como base a aplicação da tecnologia de virtualização. Uma visão geral da arquitetura e do seu funcionamento será apresentada, bem como as principais interações entre os módulos. Ao final do capítulo, a integração da arquitetura proposta ao Sistema NIDIA será apresentada.

## 3.1 Arquitetura proposta

A arquitetura proposta nesta dissertação tem como objetivo prover segurança aos usuários de ambientes de computação em nuvem em nível de infraestrutura, ou seja, protegendo as máquinas virtuais que compõem a camada de IaaS da nuvem que serve de base às outras camadas (PaaS e SaaS). Para proteger as VMs dos usuários, foi utilizado como base para a concepção do IDS os resultados da pesquisa sobre as principais ameaças à computação em nuvem realizada pelo CSA [43], especificamente as ameaças de uso abusivo e transgressivo da Computação em Nuvem e sequestro de contas, serviços e tráfego.

O uso abusivo e transgressivo da Computação em Nuvem ocorre quando os usuários da nuvem fazem um uso abusivo dos serviços que contratam, como por exemplo, hospedar websites falsos ou instanciar máquinas virtuais em um mesmo hardware de uma outra máquina virtual de uma possível vítima. Já no sequestro de contas, serviço e tráfego, acontece o roubo das credenciais de um usuário legítimo da nuvem, possibilitando a um invasor se passar por um usuário legal e cometer ações maliciosas contra outros usuários ou mesmo efetuar ataques fora da nuvem.

A pesquisa sobre superfícies de ataque na Computação em Nuvem realizada por [9], também serviu de base para a construção do IDS. Nesse estudo, podemos observar que através da interface fornecida pelo provedor ao usuário, por meio de um acesso não autorizado ou mesmo por abuso dos recursos da infraestrutura, podem-se executar ataques contra serviços executando na infraestrutura do provedor, no intuito de parar os serviços ou obter acesso às informações contidas em tais serviços. Portanto, uma potencial ameaça em ambientes de computação em nuvem parte internamente, das próprias máquinas virtuais em execução, sendo que

estas máquinas virtuais  podem ou não efetuar atividades consideradas suspeitas ou maliciosas, o que depende do perfil dos usuários proprietários das contas no provedor de nuvem. A arquitetura proposta é mostrada na Figura 3.1 e na Figura 3.2 temos uma visualização da arquitetura com diversos nós do IDS sendo controlados pelo IDS_Admin.

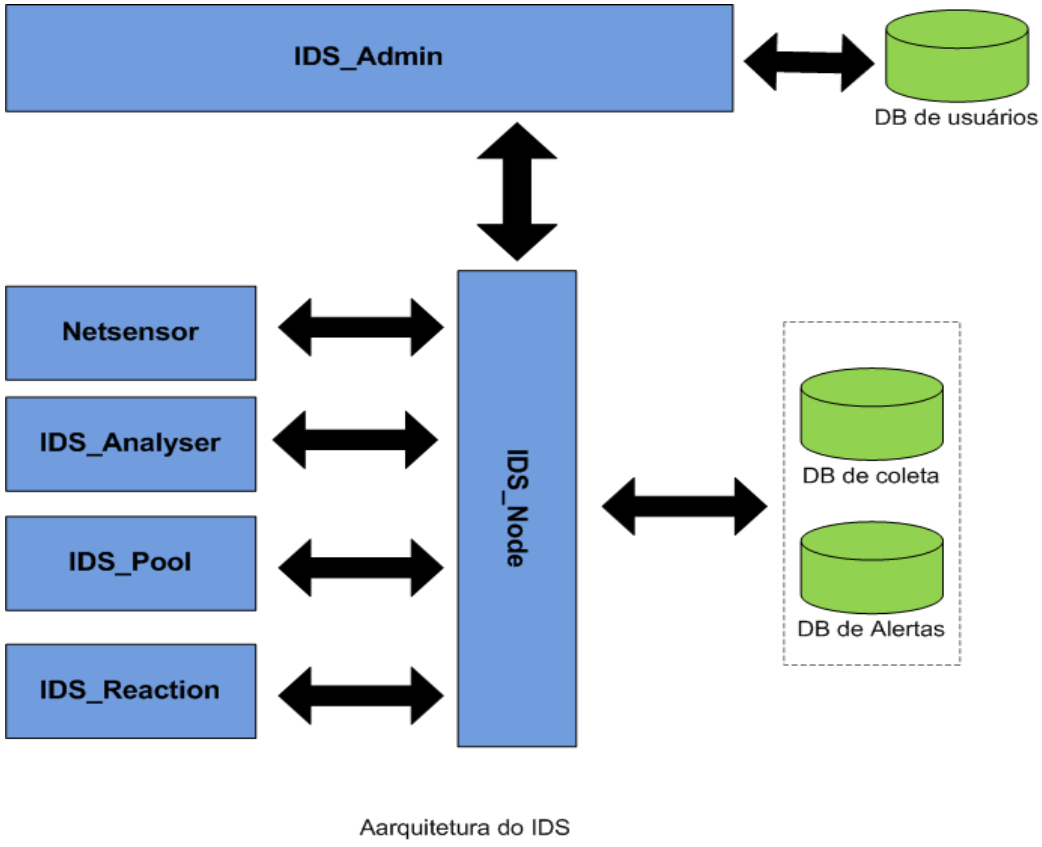



**Figura 3.1**: Arquitetura do IDS Proposto

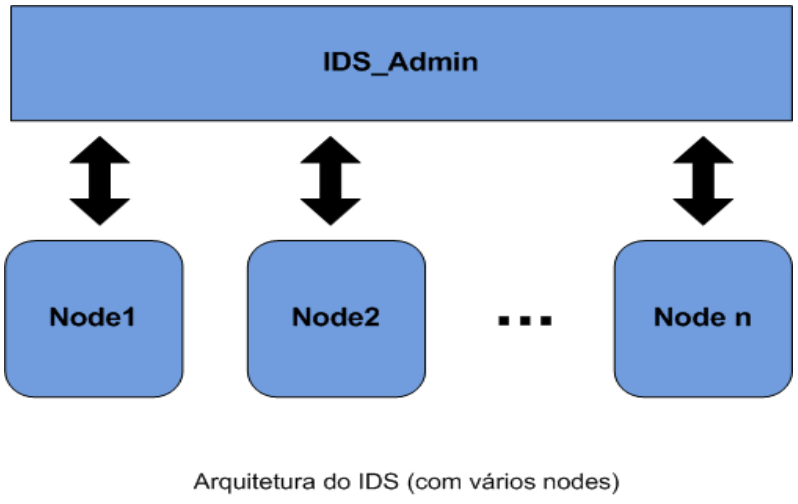



**Figura 3.2:** Arquitetura do IDS (com vários nodes)

Na Figura 3.1 temos a arquitetura do IDS que se aplica a cada nó da nuvem, onde através de seus componentes coleta informações das máquinas virtuais efetuando a detecção de atividades suspeitas e posteriormente enviando alertas ao módulo de controle, denominado IDS_Admin, já na Figura 3.2 é apresentado a expansão em uma arquitetura de nuvem com vários nós, sendo que o IDS_Admin é o elemento central que controla os vários módulos do IDS e cada componente denominado Node (Node1, Node2,...,Noden) engloba os elementos IDS_Node, Netsensor, IDS_Analyser, IDS_Pool e IDS_Reaction, o que torna a arquitetura do IDS mista entre centralizada e distribuída. Os detalhes de cada componente, bem como seu funcionamento serão detalhados nas seções seguintes.

### 3.2 Ambientação do IDS

A metodologia de construção do IDS foi feita, tendo por base os ambientes *open source* de infraestrutura como serviço como o EUCALYPTUS [18] utilizando o hipervisor KVM [72], onde existe um ponto de entrada principal da nuvem chamado *Cloud controller* e os componentes de execução de máquinas virtuais chamados de *node controller*. Assim, um módulo central é encarregado de criar e iniciar módulos responsáveis por monitorar o comportamento das VMs em cada *node controller* recebendo alertas e solicitações de contramedidas.

### 3.3 Funcionamento do IDS

O IDS utiliza uma estrutura centralizada e hierárquica onde um componente denominado de IDS admin inicializa os módulos em cada *node* da arquitetura de nuvem, controlando e recebendo informações de cada componente IDS Node através de conexões de rede . Um componente local, denominado IDS Node, é instanciado em cada *node* da nuvem e tem a função de monitorar as máquinas virtuais dos usuários. Esse monitoramento é feito através de sensores de IDS inseridos nas máquinas virtuais, que capturam informações das VMs dos usuários pelo tráfego das vlans. Com essas informações, é feita uma análise local, através de assinatura, caso seja encontrado um padrão de ataque, um alerta é enviado para o controlador central e uma contramedida é efetuada (desligamento da VM, por exemplo), caso não seja encontrado um padrão, as informações são encaminhadas ao componente localizado no node da nuvem para uma análise mais completa, utilizando o método de anomalia, ainda pode ser encaminhado para um IDS com uma base de intrusão mais abrangente como o NIDIA. A instanciação dos sensores nas máquinas virtuais é feita de acordo com a alocação das mesmas pelos usuários

da nuvem, acompanhando, assim, o crescimento ou a redução de consumo de recursos (propriedade de elasticidade da Computação em Nuvem).

O funcionamento da arquitetura proposta pode ser visualizado no diagrama de atividades mostrado Figura 3.3. O objetivo do diagrama de atividades é mostrar uma visão simplificada do funcionamento interno de um sistema, com suas funções ou processos [63]. As atividades de um sistema são modeladas de acordo com o fluxo ações executadas pelo sistema. Quando usado para a modelagem de software, as atividades normalmente representam um comportamento invocado como resultado da chamada de um método. Quando usado para modelagem de negócios, as atividades podem ser desencadeadas por eventos externos, tais como uma ordem a ser colocada das ações desenvolvidas [64,65].

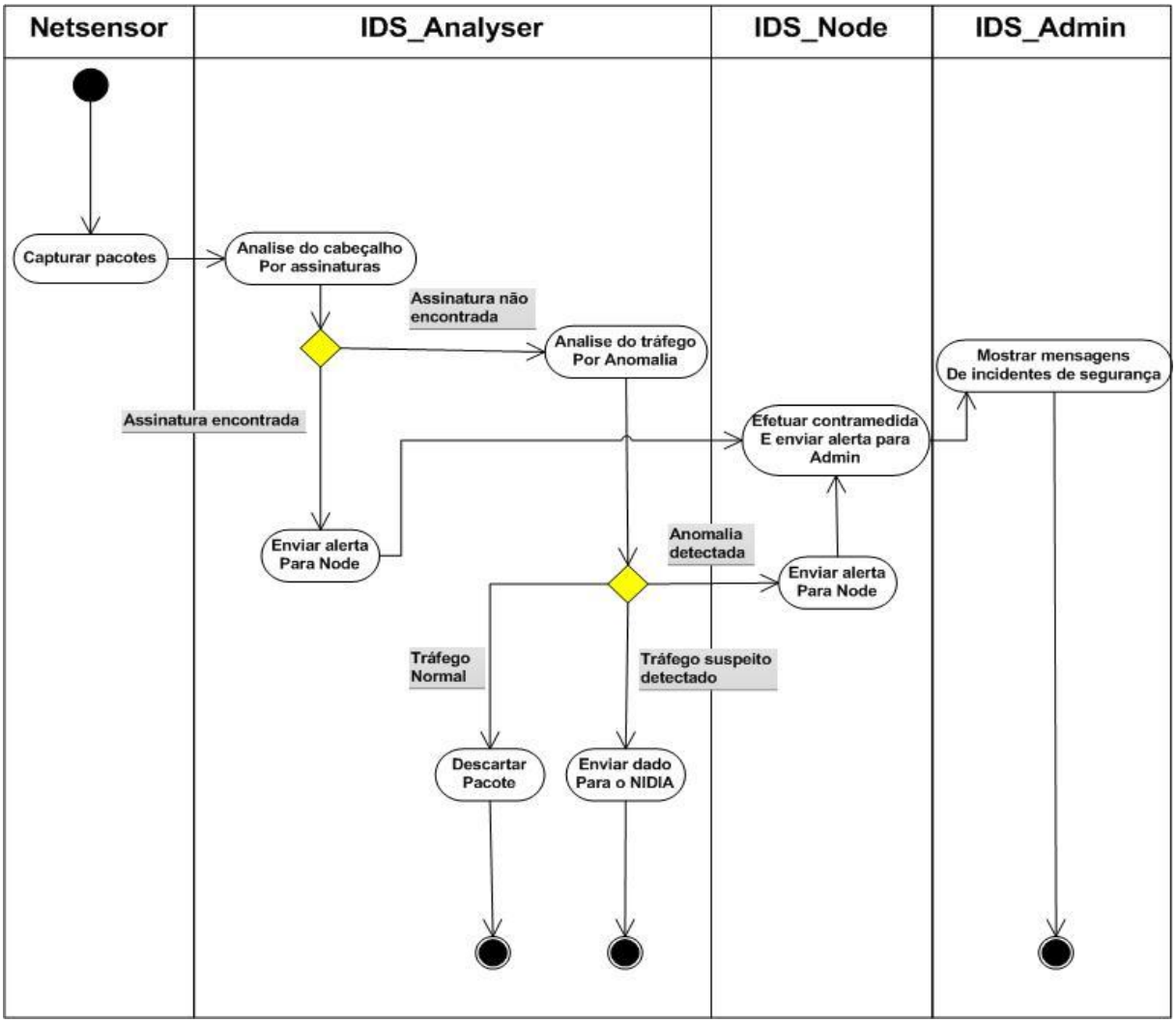



**Figura 3.3:** Funcionamento do IDS – Diagrama de Atividades

Com a captura de pacotes pelo sensor, é iniciado o fluxo de atividades para o sistema de detecção de intrusos. Após sua captura, estes pacotes são formatados e

encaminhados para a análise de assinaturas onde são comparados com um padrão pré-definido no intuito de identificar ataques conhecidos. No caso de ser encontrado um padrão, um alerta é enviado ao componente Node que identificara o ataque e enviará um relatório ao componente Admin e ainda acionará o componente de contramedidas. Caso, não seja encontrado nenhum padrão de ataque, os pacotes são encaminhados para a análise por anomalia, onde o tráfego de rede é comparado com um padrão capturado anteriormente em um período considerado livre de ataques. Nesta etapa três situações podem ocorrer: na primeira situação pode não ser identificado nenhuma anomalia, sendo que os pacotes são descartados, na segunda possibilidade, uma anomalia pode ser encontrada, onde um alerta é enviado ao Node e este encaminha também um alerta ao Admin e posteriormente uma contramedida é efetuada e em um terceira situação, é detectado uma suspeita de ataque, sendo que neste caso os pacotes considerados suspeitos podem ser enviados a um outro IDS, como o NIDIA [8].

O componente Node, de acordo com os resultados recebidos pelos analisadores, ativa o componente de contramedidas e envia um relatório de incidentes de segurança ao componente Admin, que exibe as informações de incidentes e ações efetuadas ao administrador do sistema.

## 3.4 Componentes do IDS

A seguir serão apresentados os componentes que formam o IDS proposto, bem como seu funcionamento.

### 3.4.1 IDS Admin

O IDS admin é o 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 nodes do sistema de nuvem. Mantém informações sobre  o comportamento do IDS em toda a nuvem, mostrando uma visão geral do comportamento do sistema, disponibilizando uma interface funcional para o administrador da nuvem. Através da base de dados de usuários da nuvem, tem a função de identificar a que conta de usuários pertencem as máquinas virtuais envolvidas em incidentes de segurança reportadas ao IDS admin pelos diversos nós da nuvem.

### 3.4.2 IDS Node

Tem a função de instanciar e controlar os sensores e analisadores localizados nas máquinas virtuais, monitorando assim o ambiente virtual da nuvem. Desta forma, é responsável por receber os alertas de segurança provenientes da análise de assinatura e encaminhá-los ao agente admin. Ao término da análise de assinatura, o agente node encaminha os pacotes capturados para uma análise de detecção de anomalias, a fim de encontrar tráfegos suspeitos, eventos indesejados ou padrões que indiquem intrusão. Conforme as informações periodicamente recebidas pelo agente Pool sobre a existência ou não de máquinas virtuais de usuários em execução, instancia ou remove os sensores e analisadores do ambiente virtual no intuito de tanto poupar recursos do ambiente de nuvem, quando da ausência de VMs executando, quanto de acompanhar o crescimento da nuvem, quando mais clientes solicitam VMs.

### 3.4.3 IDS Pool

Monitora o ambiente virtual da nuvem, buscando por máquinas virtuais ativas. Para realizar esta tarefa comandos do sistema operacional hospedeiro, ou comandos do próprio middleware de solução de computação em nuvem podem ser utilizados por este módulo do IDS. Ao encontrar máquinas virtuais em execução, envia uma mensagem ao agente node para que sejam ativados os sensores do IDS. Da mesma forma, quando nenhuma máquina virtual é encontrada em execução é enviando uma mensagem ao agente node para que desative os sensores, no intuito de poupar os recursos da nuvem.

### 3.4.4 IDS Sensor

A detecção é iniciada a partir deste componente, que tem a função de capturar os pacotes que trafegam na rede na qual se encontram as máquinas virtuais dos usuários da nuvem. O sensor interage com a rede de forma passiva, atuando como um *sniffer* e interferindo o mínimo possível, para não comprometer a performance do ambiente das máquinas virtuais e não corromper o tráfego. Além da captura, é feita uma filtragem dos pacotes capturados, selecionando apenas as informações dos pacotes que interessam para a análise a ser feita posteriormente. Os pacotes são gravados em uma base de dados, ficando assim disponíveis para que seja feita a análise tanto de assinatura quanto de anomalia. Devido ao fato dos ambientes de

computação em nuvem possuírem uma forma te atuação na qual terceirizam sua infraestrutura para outros clientes, a questão da privacidade dos clientes deve ser levada em consideração, pois leis de privacidade podem impedir o monitoramento dos dados ou dos recursos utilizados pelos clientes da nuvem, mesmo que seja para fins de inspeção e segurança, pois com tais procedimentos seriam realizados acessos não autorizados aos dados dos usuários. Segundo [66], as leis de privacidade podem dificultar no processo de adicionar medidas de segurança para a Computação em Nuvem. Assim, para que a captura dos dados feita pelo sensor, não vá de encontro com as leis de privacidade, somente as informações contidas no cabeçalho dos pacotes IP são armazenadas para serem analisadas posteriormente pelos demais componentes do IDS.

### 3.4.5 IDS Reação

É o componente responsável por efetuar uma contramedida caso uma ação maliciosa seja detectada.

As medidas tomadas vão desde uma pausa em uma máquina virtual suspeita de apresentar comportamento suspeito, até o desligamento ou a destruição de uma máquina virtual detectada como maliciosa. Ao ser detectada uma atividade maliciosa, o componente de reação é acionado pelo agente node, desencadeando uma ação programada para a intrusão e depois finalizando a sua execução.

### 3.4.6 Análise assinatura

O analisador de assinatura localiza-se no ambiente virtualizado criado pelo IDS. É responsável por analisar os dados gerados pelo sensor, contribuindo para o processo de detecção ao aplicar regras de detecção, podendo assim identificar ações maliciosas de alguma máquina virtual, antes de enviar as informações ao componente node dos IDS. Se for encontrado um padrão de ataque, uma mensagem de alerta deve ser enviada ao componente Node, e mesmo que não seja detectada uma ação maliciosa, os pacotes capturados são enviados ao agente node para que seja feita uma análise mais detalhada. Devido ao fato mencionado na seção 3.4.4 sobre as leis de privacidade, somente o conteúdo do cabeçalho dos pacotes IP são analisados, onde são procurados padrões de ataques já conhecidos, como possíveis pré-ataques como portscan. Os ataques baseado em cabeçalho utilizam-se de vulnerabilidades na pilha de protocolos TCP/IP e podem ser

identificados com a utilização de um filtro de pacotes ou regras especificas de conteúdo de cabeçalho pelo IDS.

### 3.4.7 Análise anomalia

O analisador de anomalia localiza-se fora do ambiente virtualizado, mais especificamente no componente Node. É responsável por analisar os dados gerados pelo sensor e que foram encaminhados pelo componente de análise de assinatura para ser feita uma nova análise. A análise de anomalia é realizada de tal forma que um alerta é emitido para o node em caso de confirmação de comportamento anômalo em alguma VM ou, em caso de suspeita, as informações podem ser enviadas para serem analisados por um IDS com uma base de dados de intrusão de maior abrangência, como o NIDIA. A análise é feita com base na variação da quantidade de tipos de pacotes capturados na rede. Segundo o trabalho realizado por Zaidi [67], e também aplicado em [68], a variação da quantidade dos tipos de pacotes capturados entre valores definidos em um limite inferior e um limite superior, será utilizada como base para determinar se o tráfego de rede é considerado normal, se estiver abaixo do limite considerado inferior, anormal, se for maior que o limite superior, ou suspeito caso esteja entre o limite inferior e o limite superior.

## 3.5 Integração com o IDS NIDIA

Esta seção descreve como seria aplicada a estrutura do IDS proposto nesta dissertação ao modelo do IDS apresentado por [8], o NIDIA[7].

### 3.5.1 O Sistema NIDIA

O Projeto NIDIA foi criado na Universidade Federal do Maranhão- UFMA em 2001 [8], possuindo como objetivo a melhoria das técnicas de detecção de intrusão em redes de computadores. Constitui-se na proposta de um IDS baseado em agentes, capaz de detectar incidentes de segurança através da análise feita em dados capturados de logs de hosts e de pacotes de tráfego de rede, podendo assim gerar um índice de suspeita de ataque e, ainda efetuar contramedidas [69].

O Projeto NIDIA, por ser inspirado no modelo lógico do CIDF (*Common Intrusion Detection Framework*) [47], possui agentes com função de geradores de eventos (agentes sensores), mecanismos de análise dos dados (agentes de monitoramento

[7] *Network Intrusion Detection System based on Intelligent Agents.*

e de avaliação de segurança), mecanismos de armazenamento de histórico e um módulo para realização de contramedidas (agente controlador de ações) [70]. Além disso, existem agentes responsáveis pela integridade do sistema e pela coordenação das atividades do IDS como um todo e pela atualização das bases de conhecimento [8].

A arquitetura do NIDIA é composta por camadas [69], como podemos ver na figura 3.4. As atividades de cada camada são executadas pelos comportamentos dos agentes, presentes nas camadas. A comunicação entre as camadas se comunicam por meio dos agentes que trocam informações importantes, contribuindo assim para desempenhar suas funções. A seguir apresentamos resumidamente uma descrição das funcionalidades das camadas e dos agentes e demais elementos que compõem o NIDIA:

- **Camada de Monitoramento –** é a camada que tem por finalidade captar a ocorrência de eventos e fornecer informações sobre o mesmo para o resto do sistema. Esta camada é composta pelos agentes SMA (System Monitoring Agent) que funcionam como “sentidos” receptores do sistema. Os agentes SMA realizam uma pré-formatação nos dados obtidos antes de envia-los ao agente de avaliação. Os agentes SMA dividem-se em dois tipos:

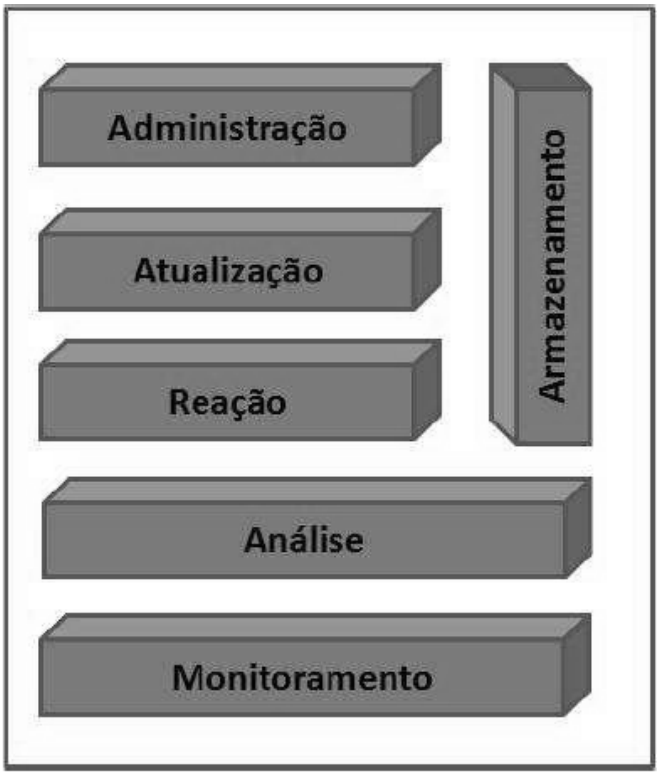



**Figura 3.4:** Arquitetura do NIDIA [8]

1. Agente Sensor de Rede: responsável por capturar os pacotes que trafegando na rede, atuando em pontos estratégicos da rede e funcionando como monitores de rede passivo, trabalhando em modo promíscuo.
2. Agente Sensor de Host: coletam informações em tempo real de um host em particular e enviam para análise.

- **Camada de Análise –** Tem a função de analisar os eventos oriundos da camada de monitoramento. Os eventos coletados são formatados para que padrões de ataques possam ser identificados e posteriormente confirmados como um verdadeiro ataque. Para isso, são utilizados bases de conhecimento, como a base de dados de padrões de intrusão (IIDB, Incidents of Intrusion and Forensic Information Database) e a base de estratégias (STDB, Strategy Database). Nesta camada estão localizados os agentes SEA (Security Evaluation Agent) que são responsáveis pela análise dos eventos coletados pela camada de monitoramento, emitindo uma notificação de intrusão.
- **Camada de Reação –** Na detecção de um problema de segurança, é nesta camada que são tomadas contramedidas. Para isso, de acordo com o parecer emitido pelo SEA, são utilizadas as bases de dados de estratégia (STDB) e ações (RADB, Reaction Database) para efetuar uma contramedida. Nesta camada estão localizados os agentes:
    1. Agente SCA (System Controller Agent), que realizam ações sobre os eventos previamente notificados, utilizando para esta tarefa as bases de conhecimento, como IIDB ( base de intrusos e intrusão), a base de dados de incidentes de intrusão e informação forense (DFDB) além da base de estratégias (STDB).
    2. Agente de contramedida (CMA, Counter-Measure Agent), responsável por decidir qual a contramedida a ser efetuada, sendo que essas contramedidas são mapeadas em ações enviadas aos módulos atuadores [71].
    3. Agente Atuador (AA, Actuator Agent), executa as ações definidas pelos agentes de contramedidas.

- **Camada de Atualização –** A atualização das bases de informações fica a cargo desta camada. Qualquer camada pode consultar diretamente as bases de informação, no entanto somente a camada de atualização poderá realizar inserções. Possui também a responsabilidade de manter a integridade e consistência das informações armazenadas. Nesta camada estão localizados os agentes SUA (System Updating Agent) que atualizam as bases DFDB, IIDB, RADB e STDB.
- **Camada de Administração –** A administração e a integridade de todos os agentes do NIDIA é de responsabilidade desta camada. Nesta camada estão localizados os agentes MCA (Main Controller Agent), além do agente de gerenciamento.
- Camada de Armazenamento – Tem por finalidade manter de forma persistente informações provenientes pelas demais camadas e por armazenar informações que estas geram. Nesta camada estão localizadas as seguintes bases de dados utilizadas pelo NIDIA:
  1. STDB, tem a função de armazenar as estratégias para a política de segurança.
  2. RADB, armazena informações necessárias a tomada de contramedidas em caso de atividade suspeita.
  3. IIDB, base de padrões de ataque utilizados para a detecção.
  4. DFDB, base de dados responsável por armazenar os incidentes de intrusão, que pode servir de fundamento para investigar e provar a culpa de um determinado atacante.

### 3.5.2 Integração do IDS proposto ao NIDIA

A solução proposta pode auxiliar o NIDIA na detecção de ameaças internas, sendo útil em ambientes de IaaS para computação em nuvem. A proposta de integração para os dois modelos é facilitada devido ao fato das arquiteturas serem organizadas em camadas, onde propomos a inserção dos componentes do IDS proposto nas camadas do NIDIA e, adaptando estes componentes para trabalharem com os agentes NIDIA. Na proposta de integração as camadas do NIDIA foram organizadas para trabalharem em conjunto com os componentes do IDS proposto, conforme a Figura 3.5, onde representamos em cinza escuro os elementos do NIDIA, enquanto que os componentes em cinza claro representam o IDS proposto.

O componente IDS_admin poderia ser inserido para atuar junto com as camadas de administração e atualização para gerenciar os componentes instalados nos diversos nós físicos da nuvem e dar suporte aos agentes do NIDIA responsáveis pelo gerenciamento dos demais agentes, onde repassaria informações sobre os locais da nuvem onde ocorreram intrusões ou suspeitas de ações maliciosas além de detalhes dos dados coletados para realização de análises mais especializadas pelos agentes do NIDIA. O componente IDS_node poderia ser inserido logo abaixo do IDS_admin e também atuando no mesmo nível da camada de reação, onde repassaria ao IDS_admin os alertas e informações provenientes dos vários nós e máquinas virtuais da nuvem, além de em caso de contramedidas, trabalharia em conjunto com a camada de reação ao acionar contramedidas direcionadas ao eventos maliciosos detectados.

As camadas de análise e monitoramento trabalhariam em conjunto com o ID_analyser, o Netsensor e o IDS_pool, onde as duas camadas responsáveis por analisar as informações recebidas (análise no NIDIA e IDS_analyser no IDS para nuvem) trabalhariam em conjunto para identificar potenciais ameaças. O netsensor atuaria no caso especifico em coletar informações provenientes das máquinas virtuais, enquanto que a camada de monitoramento ficaria encarregada do tráfego.

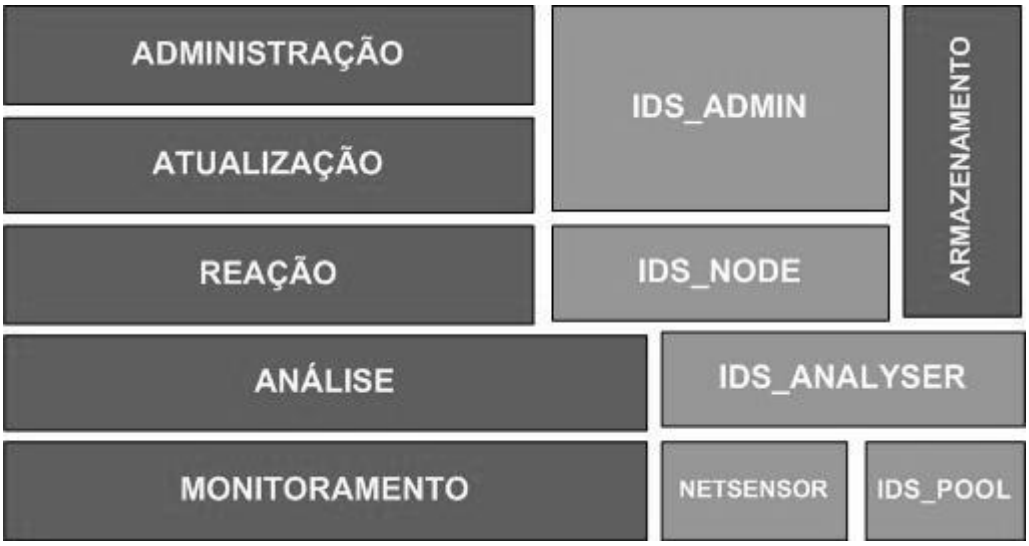



**Figura 3.5:** Proposta de integração ao NIDIA

dos nós físicos que compõem a estrutura de nuvem, sendo que o IDS_pool seria o componente responsável por iniciar a coleta de dados no ambiente virtual, dependendo da existência ou não de máquinas virtuais instanciadas. A integração proposta potencializa a proteção do ambiente computacional utilizado como

provedor de nuvem auxiliando na detecção em ambientes virtualizados, protegendo assim os usuários da nuvem contra ameaças internas, possivelmente lançada a partir de VMs maliciosas.

### 3.6 Análise comparativa

Apresentou-se neste capítulo uma proposta de sistema de detecção de intrusão para ambientes de computação em nuvem. O IDS proposto tem como objetivo a proteção das máquinas virtuais de clientes de nuvem de ataques internos, provenientes de VMs maliciosas, e tendo por requisito a proteção contras as ameaças relatadas em [43], especificamente o uso abusivo dos recursos da nuvem e sequestro ou roubo de contas.

A Tabela II apresenta uma comparação do IDS proposto com os principais IDSs para nuvem apresentados anteriormente, onde podemos observar que o uso das propriedades da virtualização é utilizado no modelo proposto de forma a acompanhar a elasticidade do ambiente de nuvem, poupando seus recursos quando necessário, enquanto que nos demais trabalhos esta propriedade é utilizada para aumentar a capacidade do próprio IDS, não se preocupando com um possível aumento no consumo de recursos da nuvem.

| características / IDS | Arquitetura | | Forma de proteção do IDS | | | Captura pacotes nas VMs | Localização | | | Uso da Elasticidade | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cent | Distr | Por nó | Por VM | Por serviço | | Nó físico | VMs | Nó físico e nas VMs | Aumentar capacidade do IDS | Poupar recursos da nuvem |
| GCCIDS [56] | | x | x | | | | x | | | | |
| Intrusion Detection System in Cloud [58] | x | | | | x | | x | | | x | |
| IDS VM system [59] | x | | | x | | x | | x | | x | |
| IDSaaS [60] | x | | | | x | x | | x | | x | |
| CIDS [61] | | x | x | x | | x | | | x | x | |
| IDS proposto | x | | x | x | | x | | | x | x | x |

**Tabela II:** Comparativo entre IDS baseados em VM para Computação em Nuvem

O IDS utiliza as propriedades das máquinas virtuais para proteção do sistema, atuando através da instanciação do IDS por cada nó do sistema e também em cada VM, onde são inseridos elementos do IDS para capturar e analisar informações. A elasticidade da computação em nuvem é explorada através do componente IDS_node que, por meio do monitoramento do ambiente virtual, instancia sensores nas VMs, aumenta ou reduz a quantidade de sensores de acordo com o crescimento ou a redução de VMs solicitadas à nuvem pelos usuários, promovendo um usos eficiente dos recursos.

A função de cada componente foi mostrada, assim como a arquitetura proposta e também uma possível integração com o IDS NIDIA.

# 4 Implementações Parciais e Resultados

Este capítulo aborda a implementação do protótipo para o modelo proposto na pesquisa. Com o objetivo de verificar o funcionamento do modelo proposto, foram implementados os mecanismos para a detecção de intrusão, utilizando a linguagem de programação Java.

Primeiramente, para avaliar a solução proposta foram criados dois ambientes de testes, simulando cenários que permitissem a coleta de resultados. Após isso, foi necessário conceber, implementar e implantar os componentes do sistema de detecção de intrusão utilizando a metodologia proposta no trabalho. Por fim, foram realizados os ataques necessários e os comportamentos do IDS observados para anotação dos resultados e avaliação da solução proposta.

A construção do ambiente de testes e os detalhes da implementação de cada um dos componentes do IDS são apresentados a seguir.

## 4.1 Ambiente de validação dos cenários

Para a realização do trabalho foram criados no LABSAC (Laboratório de Sistemas e Arquiteturas Computacionais) da UFMA (Universidade Federal do Maranhão) dois ambientes de testes que simulam uma infraestrutura de nuvem IaaS. Para o primeiro ambiente, foi utilizado um microcomputador denominado EDUCLOUD, no qual instalamos o sistema operacional Linux Ubuntu Server 12.04, o software de virtualização KVM (Kernel-based Virtual Machine) [72], a biblioteca open source Libvirt [73], o gerenciador gráfico para o ambiente virtualizado Virt-Manager [74] e demais dependências necessárias.

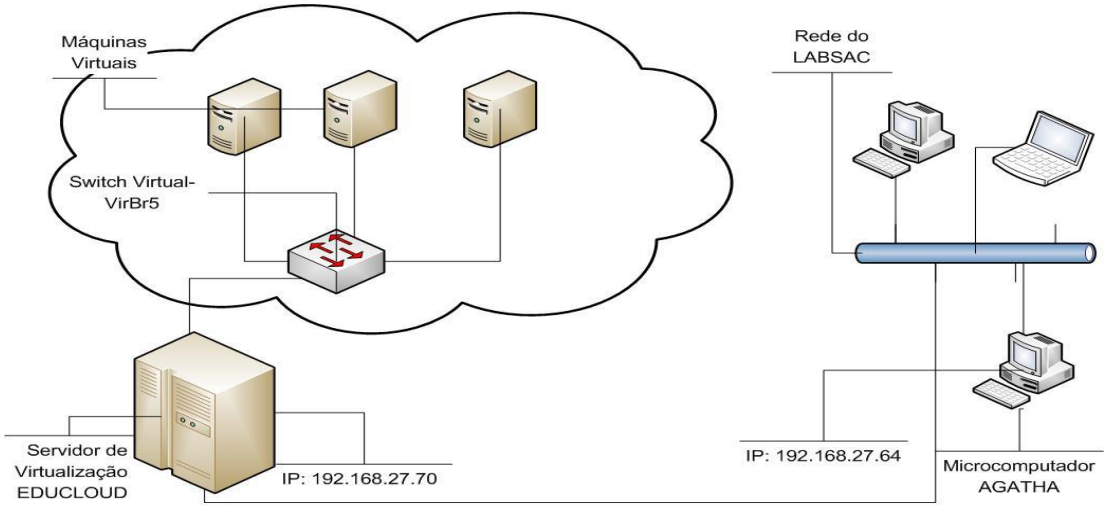



**Figura 4.1:** 1º Ambiente Montado para Realização de Testes

Na Figura 4.1 podemos visualizar esse ambiente de teste, que utiliza as máquinas virtuais criadas e as conexões entre elas para simular as operações em um ambiente virtual, o qual serve de base ao funcionamento de uma nuvem.

Para o segundo ambiente de teste utilizamos dois microcomputadores, onde foram instalados o sistema operacional Linux CentOS 6.3 e o software open source para implementação de infraestrutura como serviço EUCALYPTUS (Elastic Utility Computing Architecture for Linking Your Programs To Usefull Systems) [18], conforme mostra a figura 4.2.

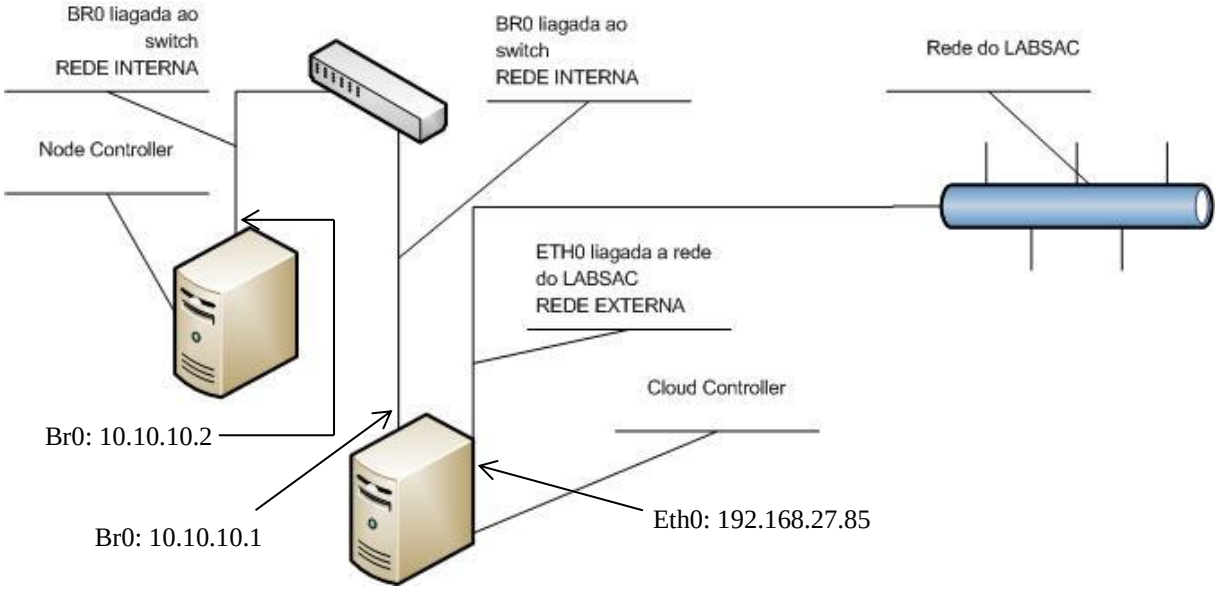



**Figura 4.2:** 2º Ambiente Montado para Realização de Testes

A configuração de hardware e softwares utilizado para a montagem dos ambientes de teste 1 e 2 são detalhados no Quadro 4.1 e Quadro 4.2 respectivamente.

| Nome da Máquina | Endereço IP | Configuração | Softwares instalados |
|---|---|---|---|
| EDUCLOUD | 192.168.27.70 | Pentium Core i7, 8GB RAM, HD de 500 GB | Linux Ubuntu 12.03 Server, KVM, Virt-Manager |
| AGATHA | 192.168.27.64 | Pentium Core i7, 8GB RAM, HD de 500 GB | Linux Ubuntu 12.03 Desktop, Java, Eclipse, Virt-anager |

**Quadro 4.1:** Configuração de Hardware e Software Utilizado no 1º Ambiente de Teste

| Nome da Máquina | Endereço IP | Configuração | Softwares instalados |
|---|---|---|---|
| CLOUDLABSAC | Eth0:192.168.27.85<br>Br0: 10.10.10.1 | Pentium Core i5, 8GB RAM, HD de 500 GB, 2 interfaces de rede ethernet. | Linux CentOS 12.03 Server, Eucalyptus 3.01 com os módulos cloud controller, cluster controller, storage controller e Walrus. |
| NODELABSAC | Br0: 10.10.10.2 | Pentium Core i7, 8GB RAM, HD de 500 GB, 1 interface de rede ethernet. | Linux CentOS 12.03 Server, KVM, Eucalyptus 3.01 com o módulo node controller. |

**Quadro 4.2:** Configuração de Hardware e Software Utilizado no 2º Ambiente de Teste

## 4.2 Implementação do IDS proposto

O principal objetivo da solução proposta é a detecção de ataques contra máquinas virtuais hospedadas na nuvem, ataques estes efetuados internamente, a partir de alguma VM maliciosa. Para isso, o monitoramento do ambiente de rede das VMs deve ser realizado, combinado com o uso estratégico de análise por assinaturas e anomalias. Este monitoramento baseia-se em informações coletadas do tráfego direcionado às VMs em modo promíscuo, sendo que a detecção de ataques é baseada no cabeçalho dos pacotes IP, devido o ambiente de teste simular uma nuvem, onde por questões de leis de privacidade, segundo [66], as informações dos clientes, representadas pelo conteúdo dos pacotes, não devem ser acessadas.

Utilizamos ataques do tipo DoS (*Denial of Service*) na avaliação do protótipo implementado, sendo que o mesmo pode ser estendido para outros tipos de ataques. Os detalhes da implementação dos componentes do IDS são apresentados a seguir.

### 4.2.1 Diagrama de classes

As classes, métodos e os atributos implementadas para o IDS podem ser visualizados pelo do diagrama de classes, como mostrado na figura 4.3, onde, podemos observar também, o relacionamento entre as classes. Os dados dos objetos são armazenados nos atributos das classes e as funções são realizadas pelos métodos ou operações implementados nas classes [64].

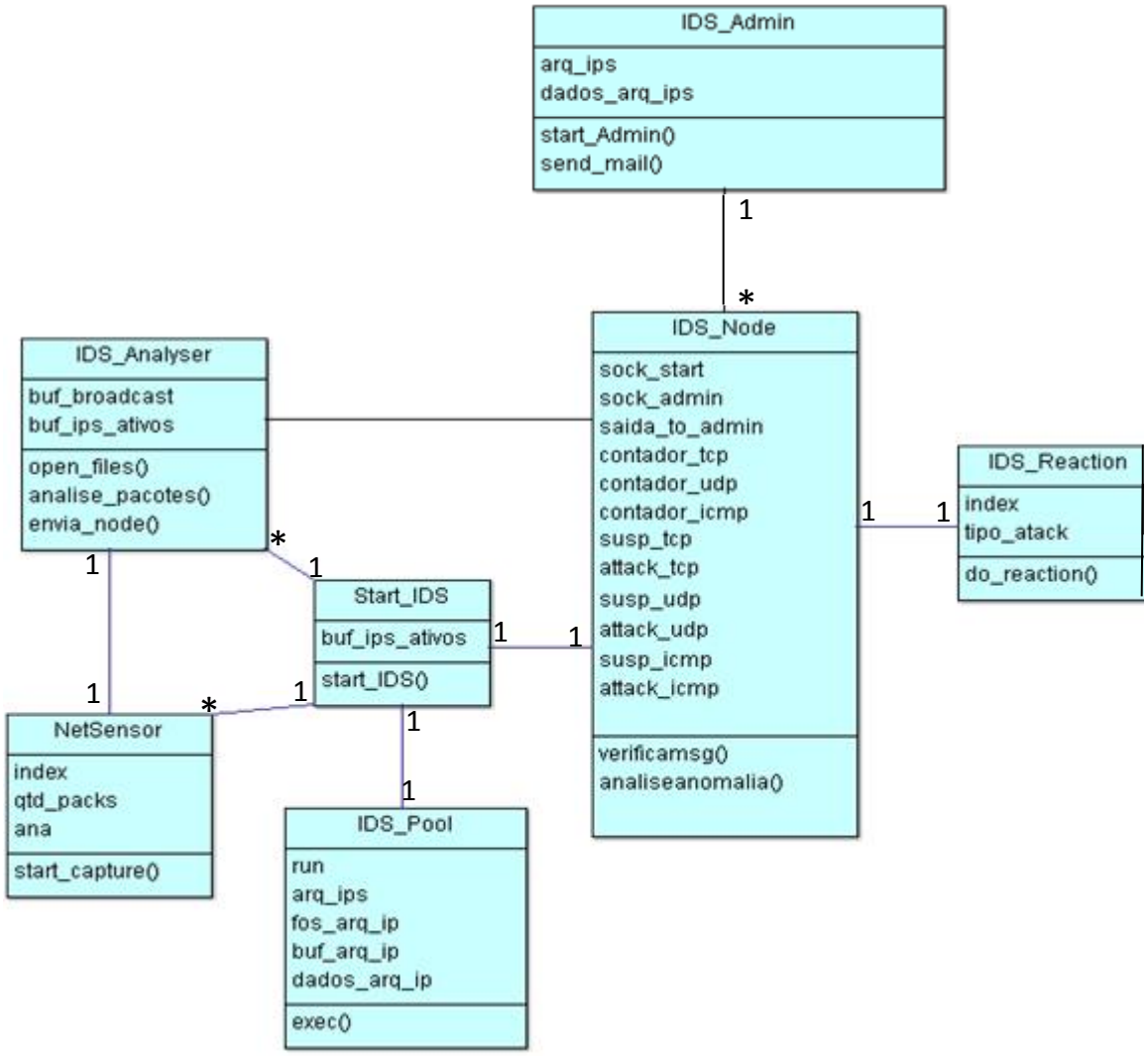



**Figura 4.3:** Diagrama de Classes do IDS Proposto

O diagrama é composto pelas seguintes classes: Netsensor, IDS_Analyser, Start_IDS, IDS_Node, IDS_Admin, IDS_Pool e IDS_Reaction. No diagrama podemos observar o relacionamento de 1 para n entre as classes IDS_Admin e IDS_Node, onde para um ambiente de nuvem existirá apenas um IDS_Admin que ficará responsável por gerenciar o IDS, mantendo uma visão geral da arquitetura e

coordenando os vários IDS_Node, os quais ficam responsáveis pelo monitoramento em cada nó físico da nuvem. Para a demonstração da funcionalidade da solução proposta foram implementadas as classes Netsensor, IDS_Anlyser, IDS_Node, IDS_Pool e a classe IDS_Admin foi implementada parcialmente, somente a interface com o usuário, foi também implementada a classe start_IDS responsável por inicializar o sistema. Os detalhes das classes implementadas são mostrados a seguir.

### 4.2.2 Implementação da classe Start_IDS

A classe Start_IDS é responsável por inicializar o IDS propriamente dito, iniciando o IDS_Pool para verificar se existe alguma máquina virtual ativa com o objetivo de decidir se o IDS será inicializado ou não. Se houver máquinas virtuais é instanciado um objeto da classe Netsensor para efetuar a captura dos pacotes da rede e posteriormente realizar a análise, como podemos observar no código 4.1, a seguir.

**Código 4.1 – Start_IDS**

```
29  // executa o pool para realizar a verificação de ips ativos
30  Thread trpool = new Thread(new Pool());
31  trpool.start();
32  trpool.join();
33
34  String ip;
35      if (buf_ips_ativos.ready()) {
36          if ((ip = buf_ips_ativos.readLine()) != null) {
37              if (ip != null) {
38              while (cliente.isConnected()) {
39                  Netsensor net = new Netsensor(
40                          Integer.parseInt(args[0]),
41                          Integer.parseInt(args[1]));
42                  Thread tn = new Thread(net);
43                  tn.start();
44                  tn.join();
45              }
46          }
47          Thread trpool2 = new Thread(new Pool());
48          trpool2.start();
49          trpool2.join();
50      }
51  }
```

### 4.2.3 Implementação da classe Netsensor

A classe Netsensor é um sensor de rede individual para ser executado em cada VM a partir do momento em que esta é ligada. Os objetos que representam os pacotes capturados na rede são definidos  por esta classe que, após realizada a

captura, instancia um objeto da classe IDS_Analyser através do método analisar_pacote(), para que seja feita a análise nos pacotes capturados. Para a realização da captura dos pacotes foi utilizada a Jpcap [75] para implementar as funcionalidades da biblioteca libpcap [76].

A biblioteca de captura de pacotes libpcap trabalha junto ao kernel do sistema operacional fornecendo um rápido acesso as cópias de pacotes. A Jpcap é um conjunto de classes Java que auxiliam na implementação de mecanismo para a captura e o envio de pacotes de rede, permitindo ao programador trabalhar em um nível de abstração maior, podendo manipular diversos tipos de pacotes, como Ethernet, Ipv4, Ipv6, ARP/RARP, TCP, UDP e ICMPv4.

A classe Netsensor captura somente pacotes IPv4 dos tipos tcp, udp e icmp, sendo adicionados aos pacotes os atributos *index* que representa o índice da placa de rede selecionada para a captura, *qtd_packets* que mostra a quantidade de pacotes de um determinado tipo que foram capturados, *tipo* que diz qual é o tipo do pacote e o atributo *date* que indica a data e a hora que o pacote foi capturado. Podemos observar na figura 4.4 um exemplo da representação de um pacote TCP no arquivo capturado. Os campos são separados por ponto e vírgula, no primeiro campo é informado a data e a hora da captura, no segundo é mostrado o tipo do pacote, no terceiro e quarto campo temos o endereço IP e a porta de origem respectivamente, no quinto e sexto campo são mostrados o endereço IP e a porta de destino respectivamente, no sétimo, oitavo e nono campo são mostrados os flags syn, ack e fin de uma conexão tcp respectivamente nesta ordem.

```
17/05/2013 16:23:55;tcp;10.10.10.174;135;10.10.10.173;135;false;false;false
```

**Figura 4.4:** Pacote TCP capturado

A seguir são exibidos os códigos dos principais métodos da classe que são start_capture(), receivePacket() e o método main().

**Código 4.2 método start_capture()**

```
46  private void start_capture() {
47      try {
48          Handler_packets handler = new Handler_packets();
49          handler.main(null);
50      } catch (Exception e) {
51          System.out.println("Erro na captura de pacotes.");
52          e.printStackTrace();
53      }
54  }
```

**Código 4.3 método receivePacket()**

```
1   public void receivePacket(Packet packet) {
2       if (packet instanceof TCPPacket) {
3           TCPPacket tcp = (TCPPacket) packet;
4           msg += String.format("tcp" + sep + tcp.src_ip.getHostAddress()
5                                + sep + tcp.src_port + sep
6                                + tcp.dst_ip.getHostAddress() + sep + tcp.dst_port
7                                + sep + tcp.syn + sep + tcp.ack + sep + tcp.fin);
8       } else if (packet instanceof UDPPacket) {
9           UDPPacket udp = (UDPPacket) packet;
10          msg += String.format("udp" + sep + udp.src_ip.getHostAddress()
11                               + sep + udp.src_port + sep
12                               + udp.dst_ip.getHostAddress() + sep + udp.dst_port
13                               + sep + udp.length);
14      } else if (packet instanceof ICMPPacket) {
15          ICMPPacket icmp = (ICMPPacket) packet;
16          msg += String.format("icmp" + sep
17                               + icmp.src_ip.getHostAddress() + sep
18                               + icmp.dst_ip.getHostAddress() + sep + icmp.type + sep
19                               + icmp.code);
20      }
21      analisar_pacote(msg);
22  }
```

**Código 4.4 método main() da classe Netsensor**

```
1   public void main(String[] args) throws java.io.IOException {
2       NetworkInterface[] devices = JpcapCaptor.getDeviceList();
3       JpcapCaptor captor = JpcapCaptor.openDevice(devices[index],
4                                    65535, true, 20);
5       captor.setFilter("ip", true);
6       captor.loopPacket(qtd_packs, this);
7   }
```

O código 4.2 mostra o método start_capture() que instancia um objeto da classe handler_packets (que implementa a interface packet_receiver do Jpcap), chamando o seu método principal para instanciar um objeto da classe JpcapCaptor que abre os dispositivos de rede disponíveis iniciando a captura dos pacotes. No recebimento do pacote, o método packet_receiver() é chamado para formatar as informações do cabeçalho e, após isso, disponibilizar os pacotes para serem utilizados pela aplicação.

### 4.2.4 Implementação da classe IDS_Analyser

A classe IDS_Analyser faz a análise de cada tipo de pacote recebido através do Netsensor que pode ser um pacote icmp, udp ou tcp, após isso, são aplicadas as regras de detecção e para cada ataque encontrado é criado um alerta, onde esses alertas são armazenados em uma base de dados de alertas, para serem enviados

para o IDS_Node que por sua vez os encaminhará para o IDS_Admin que tem a função de exibi-las ao administrador do sistema.

À medida que os pacotes capturados pelo Netsensor são recebidos pelo analisador, as informações como endereço de origem e de destino, portas de origem de destino, data e hora da captura e demais informações são extraídas para que a análise possa ser efetuada. Inicialmente é executado o método analisa_pacote() que utiliza o método open_files() para abrir os arquivos, em seguida é determinado qual o tipo de pacote a ser analisado, se este é um pacote do tipo tcp, udp ou icmp, para então ser comparado a um dos padrões de ataque usados pela classe, onde são utilizados filtros para testar se os pacotes capturados podem ser identificados com os ataques tradados pela classe, sendo que foram utilizados para fins de teste os filtros para os ataques *land*, *syn flood*, *smurf* e *prob attack.* Os detalhes dos principais métodos da classe Analyser podem ser visualizados no Código 4.5.

**Código 4.5 – método analisa_pacote_tcp()**

```
44  public String analise_pacote_tcp(String[] pack) {
45      String src_ip, src_port, dest_ip, dest_port;
46      String tcp_syn, tcp_ack, tcp_fin, tcp_data;
47      String ip_da_rede = null; // ip atual da rede interna
48      boolean eh_ip_rede = false; // if src_ip eh ip ativo da rede interna
49
50      [...]
51
52      // verifica se o ip de origem eh ip da rede ativo da rede interna
53      while ((ip_da_rede = buf_ips_ativos.readLine()) != null) {
54          if (ip_da_rede != null) {
55              if (src_ip.equals(ip_da_rede)) {
56                  eh_ip_rede = true;
57                  break;
58              }
59          }
60      }
```

**Código 4.6a Filtros usados para land ataque e syn flood**

```
86  /** ------------------------ LANDATTACK --------------------------- */
87  if ((src_ip.equals(dest_ip))) {
88      return (date + ":landattack:" + src_ip);
89  }
90
91  /** ------------------------ SYN FLOODING -------------------------- */
92  if (!eh_ip_rede && (tcp_syn.equals("true"))
93          && ((tcp_ack.equals("false")))) {
94      return (date + ":synflood:" + dest_ip + "/" + dest_port);
95  }
```

**Código 4.6b Filtros usados para icmp flood, prob e smurf**

```
177   /** ------------- ICMP FLOOD ------------ */
178   if (!eh_ip_rede && !eh_ip_broad) {
179       return (date + ":icmpflood:" + dest_ip);
180   }
181
182
183   /** ------------ PROB ATTACK--------------*/
184   if (eh_ip_rede && !eh_ip_broad && type.equals("8")
185           && code.equals("0")) {
186       return (date + ":probattack:" + src_ip);
187   }
188
189   if (!eh_ip_rede && eh_ip_broad) {
190       return (date + ":probattack:" + dest_ip +
191           " src_ip spoofed -> " + src_ip);
192   }
193
194   /** ------------ ATAQUE SMURF --------------*/
195   if (eh_ip_rede && eh_ip_broad) {
196       return (date + ":smurf:" + src_ip);
197   }
```

O Código 4.5 mostra o método analisa_pacotes_tcp, o qual recebe os pacotes formatados pelo Netsensor e aplica os filtros de assinatura retornando um alerta caso tenha sido reconhecido um ataque, comparando o tráfego capturado aos filtros mostrados no Código 4.6, onde podemos identificar os ataques de rede que exploram as vulnerabilidades da pilha TCP/IP, como por exemplo, o ataque *Land* que utilizando apenas um pacote para causar um travamento no sistema (o ataque é caracterizado quando o endereço IP de origem e a porta de origem coincidem com o endereço IP de destino e porta de destino). Outro filtro utilizado é o que identifica o ataque denominado Syn Flooding, que utiliza muitos pacotes para causar algum tipo de indisponibilidade de serviço, através do consumo excessivo de recursos do sistema alvo (ataque identificado na classe quando o IP de origem é inválido e o flag syn ativado). Os outros filtros utilizados são para os ataques icmp flooding (quando o IP de origem é inválido e o IP de destino é um IP válido de uma VM), ataque Smurf (quando um pacote icmp é enviando para o endereço de broadcast da rede das VMs e todas as VMs responderão para o endereço IP da vítima referenciada pelo IP de origem) e ataque Prob (que indica se alguma VM está colhendo informações da rede, provavelmente na tentativa de realizar um pré-ataque).

### 4.2.5 Implementação da classe IDS_Node

A classe IDS_Node é responsável por receber os alertas enviados pelo IDS_Analyser e encaminhá-los ao IDS_Admin, além de receber os pacotes provenientes do IDS_Analyser que não correspondem a nenhum padrão de ataque pré-definido. Ao receber estes pacotes é realizado uma nova análise para verificar se existe alguma anormalidade no tráfego de rede capturado, para isso é aplicado o método proposto por Zaidi [67], onde a quantidade de cada tipo de pacote capturado é comparada a dois limites predefinidos, chamados de $L_{min}$ e $L_{max}$, para decidir se o tráfego será considerado normal ou não. Na situação em que o número de pacotes capturados for menor que o $L_{min}$ o tráfego é considerado normal, se o número de pacotes capturados for maior do que $L_{max}$ o tráfego é anormal, sendo considerado uma atividade maliciosa, então uma alerta e gerado e enviado ao IDS_Admin. Uma terceira situação acontece quando o número de pacotes capturados está entre $L_{min}$ e $L_{max}$, neste caso o tráfego de rede é definido como suspeito e um alerta é enviado ao IDS_Admin para alertar o administrador do sistema de tal situação. Por meio do método anomalia() é realizada a análise de anormalidade do tráfego de rede.

Nos testes realizados no laboratório do LABSAC, os valores para $L_{min}$ e $L_{max}$ foram definidos em 300 e 800, respectivamente. Para a definição desses valores, o tráfego do LABSAC foi observado no período de 9 de maio a 14 de maio de 2013, sendo capturado o tráfego do laboratório por uma hora a cada dia e feito uma média da quantidade de pacotes tcp, udp e icmp, chegando então aos valores utilizados. Esses resultados foram definidos para a realidade de utilização do LABSAC e do ambiente utilizado para os testes do IDS, precisando serem reconsiderados caso sejam aplicados a outros ambientes ou situações. A classe interna TrataAnalyser é chamada toda vez que o Analyser conecta-se ao Node para enviar uma mensagem, os códigos implementados para o IDS_Node, bem como os demais códigos feitos para o protótipo implementado são mostrados no apêndice A. De acordo com código implementado para a classe TrataAnalyser, ao receber uma mensagem proveniente do analyser, é verificado o seu conteúdo, se for "*****" (asterisco), significa que o Netsensor reinicializou, para verificar a situação do ambiente das máquinas virtuais, neste caso os contadores dos tipos de pacotes capturados são "zerados". Caso o conteúdo da mensagem seja um tipo de pacote (protocolo tcp, udp ou icmp), significa que o analisador recebeu um pacote e não verificou nenhuma assinatura de ataque, portanto, enviou a mensagem para a Node a fim de que tal informação sirva

de parâmetro para a análise de anomalia. A terceira situação é quando for um alerta de ataque enviado pelo IDS_Analyser, sendo que o mesmo é enviado ao IDS_Admin através do método saida_to_admin().

### 4.2.6 Implementação da classe IDS_Pool

A classe IDS_Pool tem a função de retornar ao IDS_Node todos os ips ativos das máquinas virtuais que estão executando em uma determinada vlan. Desta forma oferece informações para que a captura de pacotes seja inicializada ou não, objetivando assim diminuir a alocação de recursos da máquina que hospeda o IDS, possibilitando uma economia nos recursos da nuvem. A classe também contribui para o mecanismo de detecção de ataques, sendo um complemento à definição de assinaturas, por exemplo, um evento só é considerado ataque se for realizado por um IP interno, se for externo não será configurado como ataque, sendo assim, a lista dos possíveis atacantes é fornecida pelo IDS_Pool.

O Código 4.7 mostra os detalhes na implementação da classe.

A lista de endereços IP das máquinas virtuais é obtido pelo comando “arp-scan –interface=virbr5 –localnet”, onde virbr5 é o switch virtual que interconecta todas as VMs.

**Código 4.7 IDS_Pool**

```
20  String[] cut;
21  Process process = run
22          .exec("arp-scan --interface=virbr5 --localnet");
23  if (process == null) {
24      System.out.println("Pool dont started.");
25  } else {
26      System.out.println("Pool alive...");
27  }
28  BufferedReader in = new BufferedReader(new InputStreamReader(
29          process.getInputStream()));
30  String line;
31  fos_arq_ip = new FileOutputStream(arq_ips);
32  buf_arq_ip = new BufferedOutputStream(fos_arq_ip);
33  dados_arq_ip = new DataOutputStream(buf_arq_ip);
34
35  while ((line = in.readLine()) != null) {
36      if (line != null) {
37          cut = line.split("\\s");
38          if (cut[0].startsWith("10.")) {
39              dados_arq_ip.writeBytes(cut[0] + "\n");
40              System.out.println(cut[0]);
41              buf_arq_ip.flush();
42          }
43      }
44  }
```

O comando é executado através do processo process, que é realizado no sistema que hospeda as VMs, retornando as VMs ativas, sendo que seus endereços são escritos no arquivo de IPs ativos.

### 4.2.7 Implementação da classe IDS_Admin

A classe IDS_Admin foi implementada parcialmente, realizando somente a exibição dos alertas de ataques provenientes dos IDS_Node localizados nos diversos nós da nuvem. As figuras 4.5, 4.6 e 4.7, mostram os módulos IDS_Node, IDS_Admin, IDS_Pool e Netsensor sendo executados no ambiente preparado para a realização dos testes.

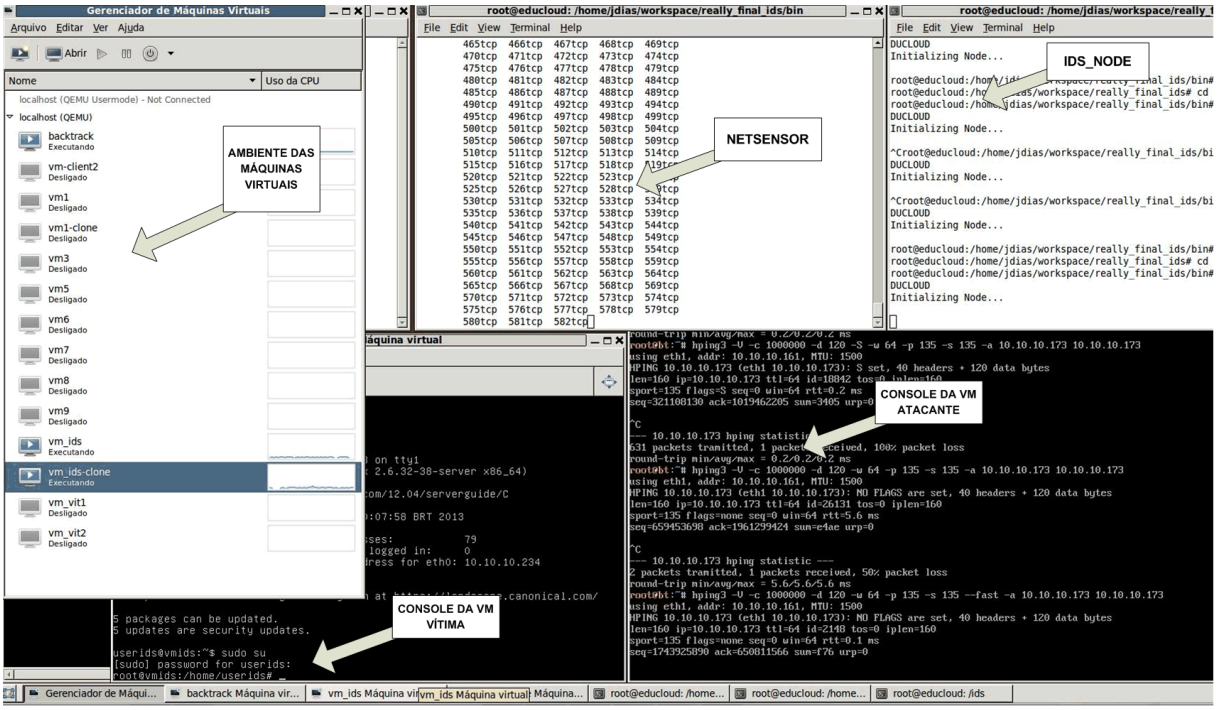



**Figura 4.5:** Netsensor e IDS_Node sendo Executados no Ambiente de Teste

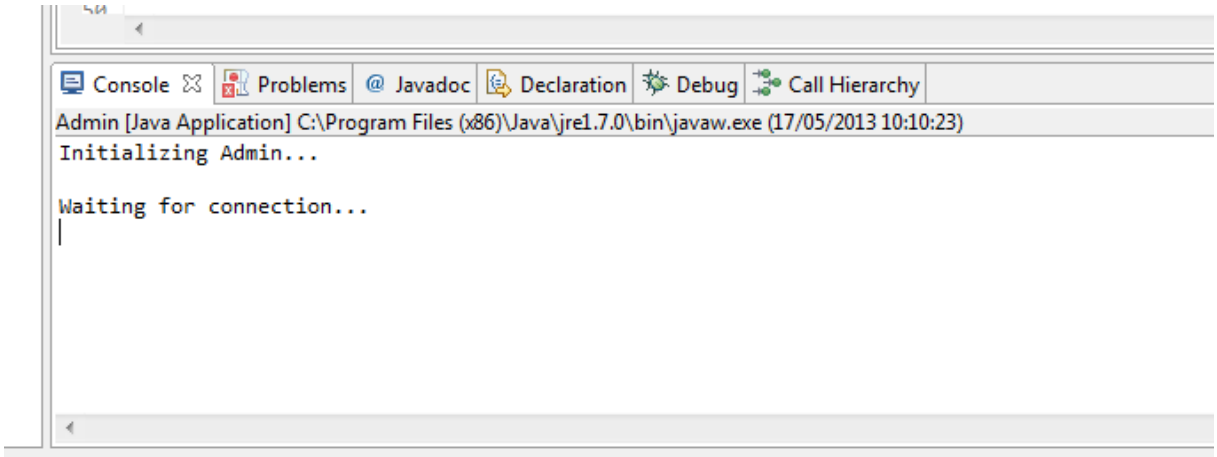



**Figura 4.6:** Inicialização do IDS_Admin no Ambiente de Teste

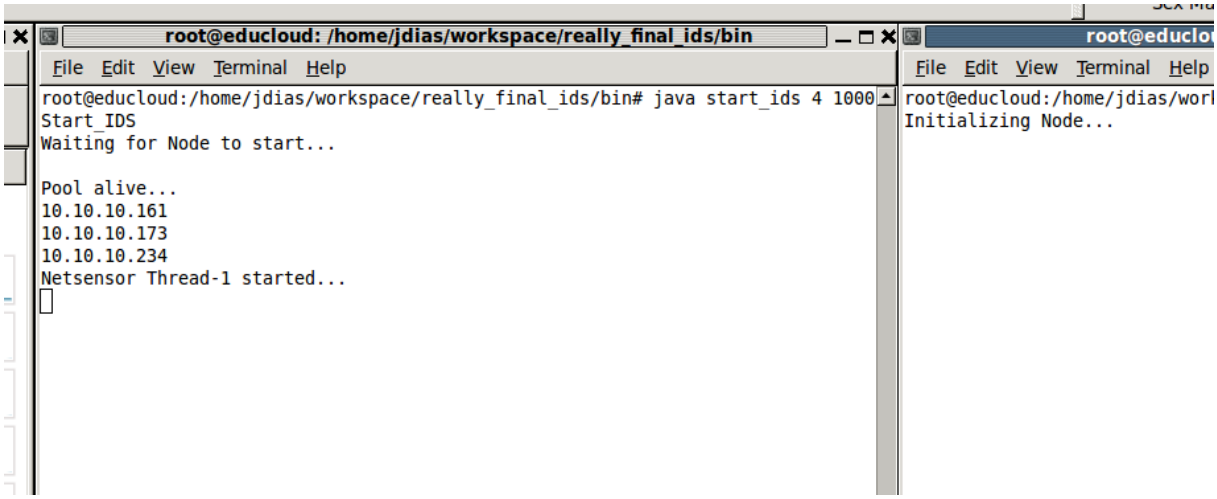



**Figura 4.7:** IDS_Pool sendo Executado no Ambiente de Teste

## 4.3 Testes e Resultados Parciais

Para avaliar o IDS proposto foram realizados ataques de negação de serviço (DoS) a partir de uma máquina virtual interna contra as outras máquinas virtuais existentes no ambiente. O IDS deve alertar ao administrador sempre que uma máquina virtual esteja sobre ataque, tentando identificar o endereço IP do atacante e o tipo do ataque em andamento. Embora a implementação do IDS e o método de ataque escolhido estarem direcionados para a detecção de ataques de DoS, o modelo do IDS proposto pode ser usado também na detecção de outros tipos de ataque.

Para o estudo do comportamento do IDS, foi inserido no ambiente de teste um novo componente, um atacante interno por meio de um máquina virtual instanciada com o sistema operacional linux backtrack 5 [82]. O objetivo dessa VM é realizar ataques contra as outras VMs do ambiente de teste, a fim de que as reações do IDS inserido neste ambiente possam ser observadas. Na máquina AGATHA, está localizado o módulo IDS_Admin, no servidor de virtualização EDUCLOUD, localizam-se os módulos IDS_Node, IDS_Analyser e Netsensor, sendo que a interface de rede escolhida para o Netsensor capturar os pacotes foi a virbr5 (switch virtual), por permitir que todo o tráfego das máquinas virtuais possa ser observado . A partir da captura dos pacotes feita pelo Netsensor na Viirbr5, os dados de alerta serão enviados ao IDS_Node (LABCLOUD: 192.168.27.70), que é o próprio controlador da infraestrutura de virtualização e deverá manter uma comunicação com o módulo IDS_Admin (AGATHA: 192.168.27.64), como podemos ver na Figura 4.8.

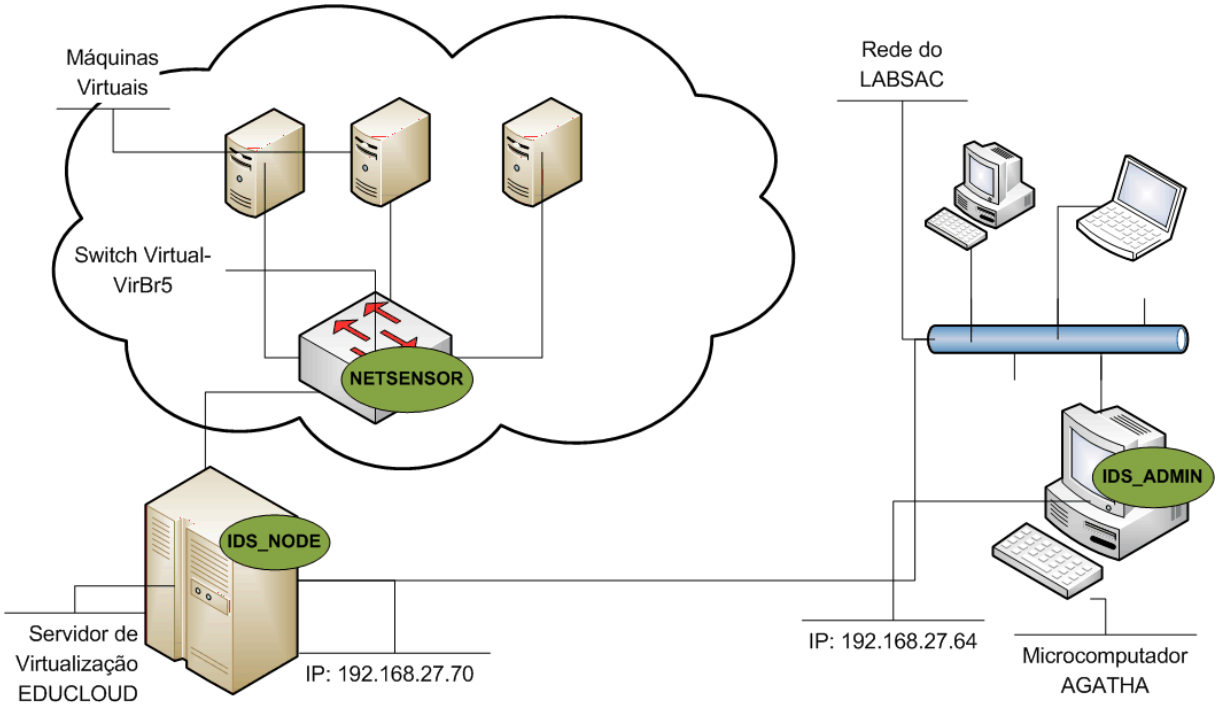



**Figura 4.8:** Componentes do IDS no Ambiente de Teste

Na Figura 4.9 e Figura 4.10, podemos observar as telas de console das máquinas virtuais utilizadas como atacante e vítima, respectivamente, bem como os seus endereços IP.

```
root@bt:~#  uname -a
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux
root@bt:~# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 52:54:00:3b:41:84
          inet addr:10.10.10.161  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe3b:4184/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:184212 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2400542 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8747056 (8.7 MB)  TX bytes:105725491 (105.7 MB)
          Interrupt:10 Base address:0xa000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:5868 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5868 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:422433 (422.4 KB)  TX bytes:422433 (422.4 KB)
```

**Figura 4.9:** Console da VM Atacante

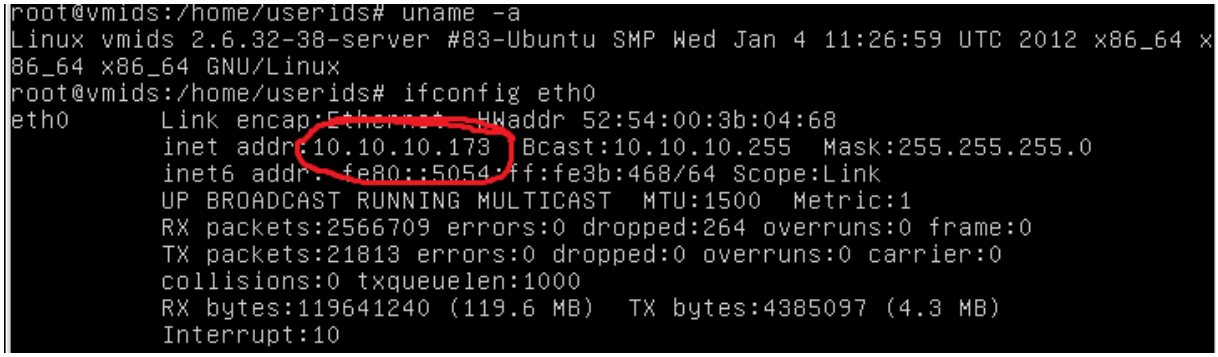



**Figura 4.10:** Console da VM Vítima

Para a realização dos testes foi utilizado o programa Hping [77], que é uma ferramenta de rede utilizada para enviar pacotes TCP/IP modificados para algum endereço alvo na rede, servindo como um excelente teste para a segurança da rede. Com a utilização desta ferramenta foram efetuados comandos simulando o tráfego malicioso contra uma máquina virtual hospedada no ambiente de teste preparado.

O primeiro ataque foi efetuado com o comando ***hping3 -V -c 1000000 -d 120 -S -w 64 -p 135 -s 135 --flood -a 10.10.5.5 10.10.10.173***, o comando simula um ataque do tipo SYN Flooding, enviando pacotes tcp com flag syn habilitado, tendo como endereço de destino o IP da máquina virtual vítima (10.10.10.173) e o endereço de origem inválido, não pertencendo a rede virtual do ambiente de teste. Os parâmetros utilizados no comando são:

- ***–V*** de verbose;
- ***-c 1000000*** que envia um milhão de pacotes;
- ***-d 120*** tamanho dos dados;
- ***-S*** que ativa o flag SYN;
- -w64 janela tcp;
- ***-p 135 –s 135*** que indica a porta TCP de origem e de destino;
- ***--flood*** envia pacotes o mais rápido possível;
- ***-a*** mascara o endereço IP de origem.

Ao efetuar o comando a partir da VM de ataque, o ataque é prontamente identificado pelo IDS, que mostra a mensagem de alerta no console do IDS_Admin, como podemos observar na Figura 4.11.

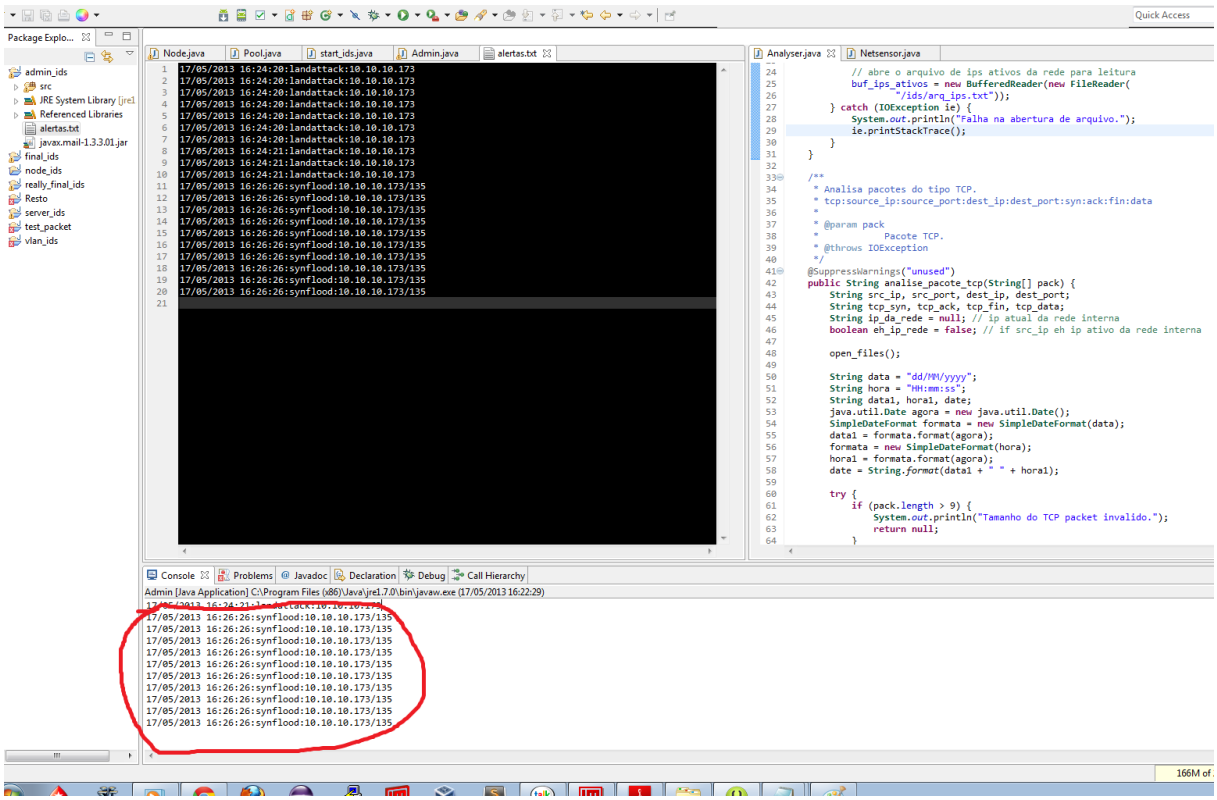



**Figura 4.11:** Ataque Syn Flood Detectado e sendo Exibido pelo IDS_Admin

O segundo teste foi realizado efetuando um ataque com o comando ***hping3 -V -c 1000000 -d 120 -w 64 -p 139 -s 139 --flood -a 10.10.5.5 10.10.10.173***, a partir da máquina virtual utilizada como atacante de IP 10.10.10.161 com destino à VM, denominada de vítima, de IP 10.10.10.173, simulando um ataque Land, onde a porta de origem, é igual porta de destino, como mostra os parâmetros –p 139 –s 139. Como resultado, o IDS também identificou o ataque Land sendo efetuado contra a VM de IP 10.10.10.173, como mostra a figura 4.12.

O comando ***hping3 -a 10.10.5.5 -1 10.10.10.173 –flood*** foi efetuado na VM atacante para simular um ataque do tipo ICMP Flood, o parâmetro ***-1*** foi utilizado para indicar que os pacotes seriam do tipo ICMP. O comando simula um flooding (inundação) de pings com o endereço de origem inválida para uma máquina virtual válida. Na Figura 4.13, podemos ver que o IDS identificou o ataque.

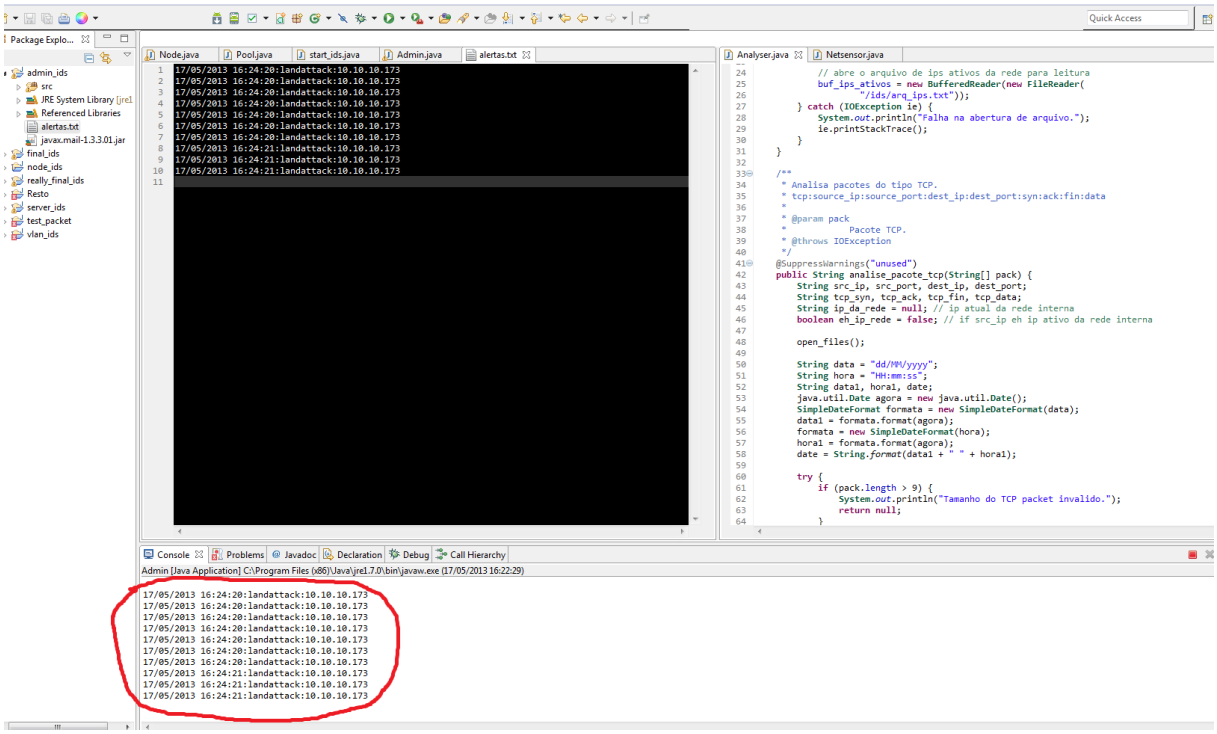



**Figura 4.12:** Ataque Land Detectado e sendo Exibido pelo IDS_Admin

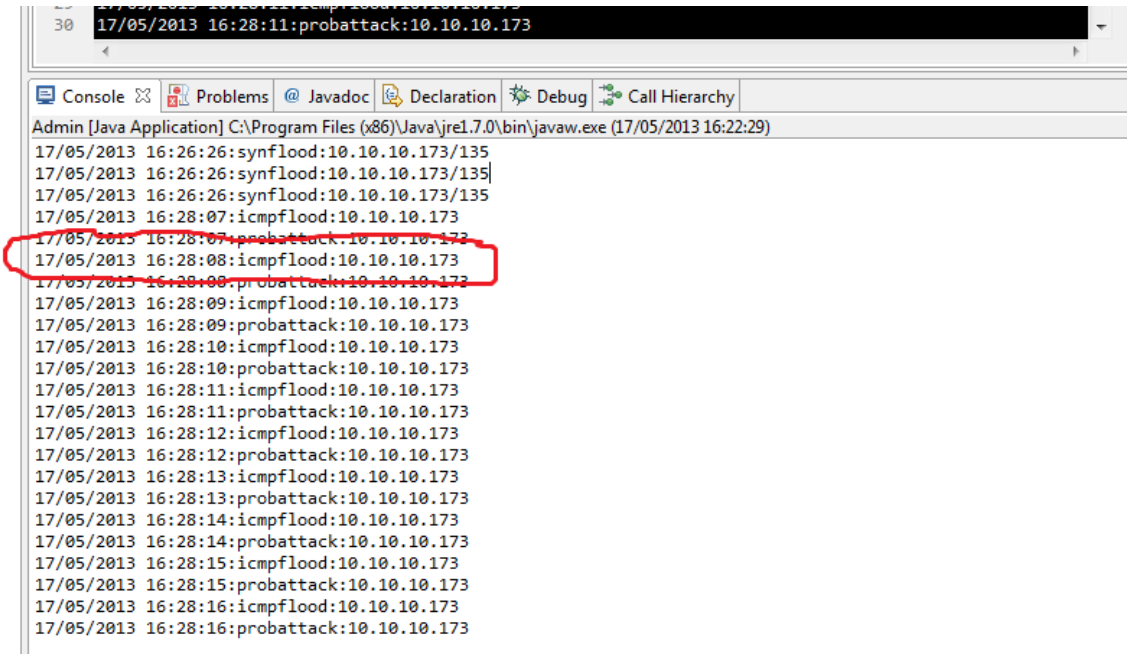



**Figura 4.13:** Ataque ICMP Flood Detectado e sendo Exibido pelo IDS_Admin

A capacidade do IDS em detectar o ataque do tipo Smurf foi testada ao executar na VM de ataque o comando ***hping3 -a 10.10.10.173 -1 10.10.10.255 –flood***. Neste ataque, pacotes do tipo ICMP são enviados para o endereço de broadcast da rede e o endereço de origem é modificado para o endereço da vítima, como podemos observa nos parâmetros –a 10.10.10.173 (IP da VM vítima) e -1 10.10.10.255 – flood

(flooding de pacotes ICMP para o broadcast da rede). Como resultado, podemos observar pela figura 4.14 que o ataque também foi identificado pelo IDS.

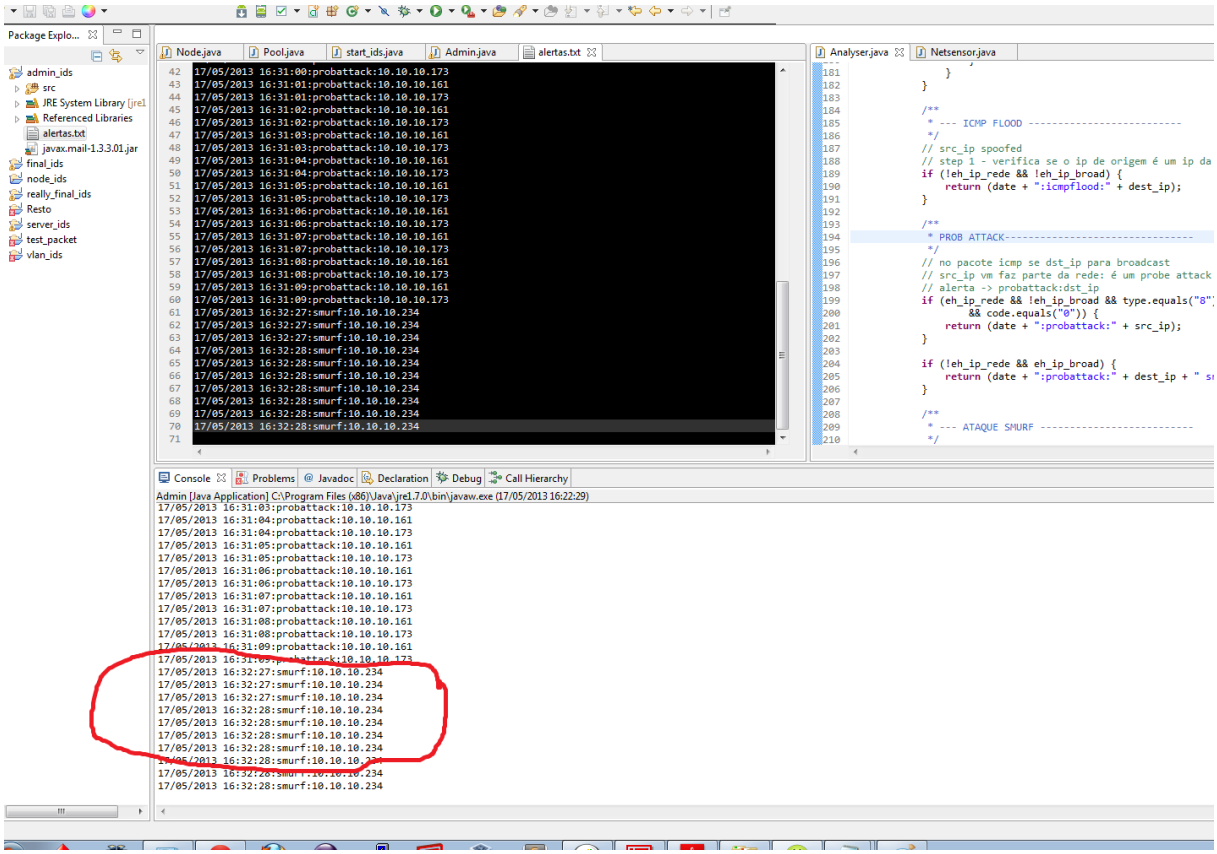



**Figura 4.14:** Ataque SMURF Detectado e sendo Exibido pelo IDS_Admin

O IDS identifica como Prob attack, a tentativa de alguma VM identificar os endereços das outras máquinas na rede, caracterizando uma tentativa de pré-ataque. Para simular essa situação, foi efetuado na VM atacante o comando ***ping 10.10.10.255 -b***, onde a intenção que todas as máquinas da rede respondam ao atacante que assim, identificará todos os Ips ativos na rede, sendo que o IDS identificou essa situação, como mostra a figura 4.15.

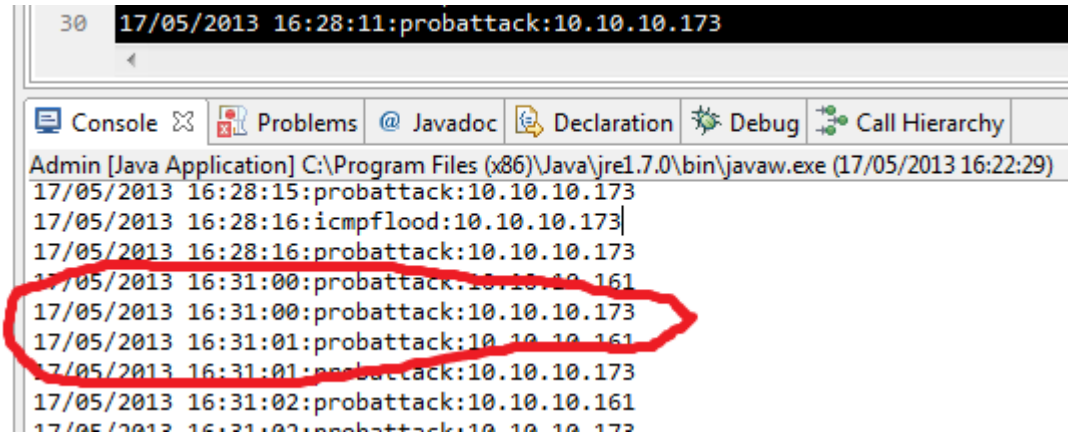



**Figura 4.15:** Prob Attack Detectado e sendo Exibido pelo IDS_Admin

E por fim foram feitos testes tentando simular um tráfego anormal, onde vários pacotes tcp, udp ou icmp foram enviados na tentativa de realizar um ataque DoS em que o tráfego não pudesse se encaixar em nenhum padrão usado no IDS. Para identificar esse tráfego anormal, foi utilizado o método proposto por Zaidi [67] no IDS, como mostrado no capitulo 4, onde a partir dos valores definidos em $L_{min}$ e $L_{max}$, configurados de acordo com o ambiente preparado para os testes, foram efetuados comandos para simular respectivamente tráfego normal, anômalo e suspeito, sendo que o IDS se comportou como o esperado como mostra a figura 4.16.

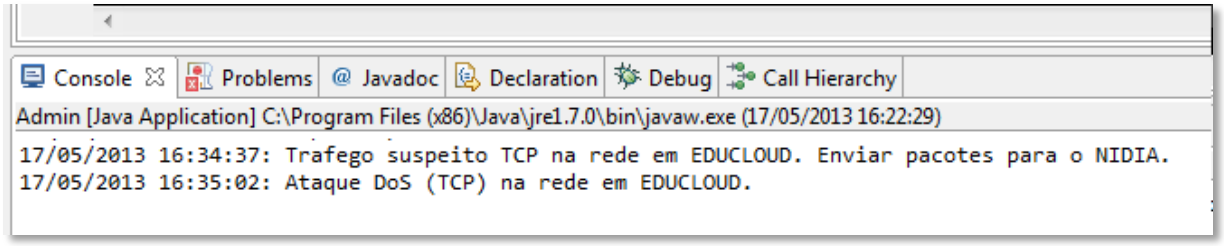



**Figura 4.16:** Tráfego Anormal Detectado pelo IDS

## 4.4 Síntese do capítulo

Neste capítulo, foi apresentado a implementação do protótipo para o modelo de IDS para ambientes de Computação em Nuvem proposto nesta dissertação. Foram também realizados testes nos ambientes montados em laboratório demostrando a potencialidade do protótipo implementado em identificar ataques efetuados contra as máquinas virtuais da nuvem a partir de alguma VM comprometida. As características da virtualização foram utilizadas para proteção das VMs bem como o uso mais eficiente dos recursos do ambiente de nuvem foi demonstrado pelo componente IDS_Pool, que informa aos demais componentes do IDS sobre a existência de VMs em execução. No capítulo seguinte apresentaremos a conclusão do trabalho, bem como as sugestões para trabalhos futuros.

# 5 Conclusão e Trabalhos Futuros

Neste capítulo, apresentamos a conclusão deste trabalho, com as contribuições e sugestões para trabalhos futuros.

## 5.1 Contribuições

No presente trabalho, um modelo para a detecção de intrusão em ambientes de computação em nuvem foi proposto, tendo como objetivos proteger os usuários da nuvem de ameaças internas e também possibilitar uma abordagem de proteção baseada em superfícies de ataque, sendo que a infraestrutura de nuvem, que poderia ser utilizada como superfície de ataque, é protegida de potenciais ameaças.

A arquitetura proposta para o IDS foi descrita, sendo detalhados o funcionamento de cada um dos seus componentes bem como os métodos utilizados na detecção dos ataques. Os estudos do CSA realizados em [43], sobre os principais riscos de segurança em computação em nuvem, especificamente os itens: uso abusivo ou transgressivo da computação em nuvem e sequestro de contas, serviços e tráfego, bem como o trabalho de [9], sobre superfícies de ataque, foram utilizados como norteadores para o funcionamento e aplicação da solução proposta.

O IDS proposto utiliza as características da virtualização como isolamento, rápida parada, rápida inicialização e elasticidade para proteger as máquinas virtuais contra ataques que possam ser efetuados de dentro do ambiente de nuvem, mais especificamente a partir de alguma VM comprometida, denominado no trabalho de VM maliciosa. A instanciação de sensores é feita por VM, onde os pacotes que trafegam nas VMs são capturados e posteriormente analisados para a identificação de ameaças, monitorando assim todo o ambiente virtual, sendo que os demais componentes residem no nó físico, fora do ambiente virtual, estando assim protegidos de possíveis ataques de VMs comprometidas.

Outra contribuição do trabalho é a utilização do componente IDS_Pool que verifica constantemente a existência no ambiente virtual de VMs de usuários em execução, promovendo um uso mais eficiente dos recursos da nuvem ao informar aos outros componentes do IDS quando é necessário instanciar sensores para captura de pacotes, poupando assim o consumo de recursos pelo IDS.

Também foi apresentada a implementação feita para os principais módulos do IDS, onde foram detalhados os principais métodos utilizados na aplicação.

Foi montado um ambiente de testes para validar o funcionamento e a capacidade de detecção de ataques do IDS, sendo que este ambiente serviu para a execução dos testes e visualização dos resultados.

Os resultados obtidos através dos testes aplicados no laboratório criado para representar a infraestrutura de um ambiente de nuvem, confirmaram a capacidade do IDS proposto em identificar ações intrusivas. Com os testes realizados, todos os ataques realizados a partir de uma máquina virtual maliciosa foram detectados pelo IDS.

Apesar de que, somente ataques do tipo DoS foram utilizados, o modelo proposto possibilita que outros tipos de ataques possam ser identificados, necessitando a implementação para os tipos de ataques específicos, sendo que os mecanismos de detecção utilizados na implementação apresentada nesta dissertação, serviram para demonstrar a funcionalidade da mesma.

## 5.2 Trabalhos Futuros

Com o desenvolver deste trabalho, foram identificados algumas possibilidades para trabalhos futuros, dentre as quais podemos sugerir:

- Implementação das contramedidas do IDS, e das funcionalidades do módulos IDS_Admin, para identificação dos usuários da nuvem que utilizem VMs para realizar ataques;
- Reconhecimento de padrões para detecção de anomalia;
- Disponibilizar a solução como serviço, operando na camada SaaS da nuvem, sendo utilizada por usuários finais;
- Desenvolvimento de um sistema de geração de assinaturas de ataques, por meio de honeypots, que trabalhem em conjunto com o IDS no ambiente de nuvem;
- Adaptar o IDS proposto para dispositivos móveis;
- Utilizar Inteligência Artificial na detecção de intrusão e adaptar os componentes do IDS para Sistemas Multi-Agentes.

# REFERÊNCIAS

[1] E. F. Coutinho, F. R. Sousa, D. G. Gomes, e J. N. de Souza, **"Elasticidade em Computação na Nuvem: Uma Abordagem Sistemática,"** 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC, Brasília, 2013.

[2] CISCO, **CloudWatch Summer 2012**. Disponível em: <http://www.cisco.com/cisco/web/UK/assets/cisco_cloudwatch_2012_2606.pdf/>. Acesso em 26 jun. 2012.

[3] CertBR, centro de estudos resposta e tratamento de incidentes de segurança no brasil. **Estatísticas dos Incidentes Reportados ao CERT.br**. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 02 jan. 2013.

[4] M. J. Ranum, **An Internet Firewall**, proceedings of World Conference on Systems Management and Security, 1992. Disponível em: <http://ftp.se.kde.org/pub/security/firewalls/tis/fwalls-slides.ps.gz/>. Acesso em: 12 out. 2012.

[5] F. Sabahi e A. Movaghar, **"Intrusion detection: A survey"**, in Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on, 2008, p. 23–26.

[6] International Data Corporation-IDC. **New IDC Cloud Service Survey: Top Benefits and Challenges**, 2009. Disponível em: <http://blogs.idc.com/ie/?p=730/>. Acesso em: 02 mai. 2012.

[7] X. Zhao, K. Borders, e A. Prakash, **"Virtual machine security systems"**, book chapter, in Advances in Computer Science and Engineering, 2009, pp 339–365.

[8] Christiane F. L. LIMA, **"Agentes Inteligentes para Detecção de Intrusos em Redes de Computadores,"** Dissertação de Mestrado, Universidade Federal do Maranhão - UFMA, São Luís, 2002.

[9] N. Gruschka e M. Jensen, **"Attack surfaces: A taxonomy for attacks on cloud services"**, in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, p. 276–279.

[10] F. R. Sousa, L. O. Moreira, e J. C. Machado, **"Computação em nuvem: Conceitos, tecnologias, aplicações e desafios"**, III Esc. Reg. Comput. Ceará-Maranhão-Piauí Ercemapi, vol. 1, 2009, pp 1-26.

[11] Zhang, S. Zhang, X. Chen, e X. Huo, **"Cloud computing research and development trend"**, in Future Networks, 2010. ICFN'10. Second International Conference on, 2010, p. 93–97.

[12] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, e I. Stoica, **“Above the clouds: A Berkeley view of cloud computing”**, Dept Electr. Eng Comput Sci. Univ. Calif. Berkeley Rep Ucbeecs, vol. 28, 2009.

[13] L. M. Vaquero, L. Rodero-Merino, J. Caceres, e M. Lindner, **“A break in the clouds: towards a cloud definition”**, Acm Sigcomm Comput. Commun. Rev., vol. 39, no 1, p. 50–55, 2008.

[14] P. Mell e T. Grance, **“The NIST definition of cloud computing”**, Nist Spec. Publ., vol. 800, no 145, p. 7, 2011.

[15] X. Tan e B. Ai, **“The issues of cloud computing security in high-speed railway”**, in Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on, 2011, vol. 8, p. 4358–4363.

[16] V. Delgado, **“Exploring the limits of cloud computing”**, Tese de Doutorado, KTH, 2010.

[17] D. Robinson, **Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster**. Emereo Pty Ltd, 2008.

[18] S. Liu, Y. Liang, and M. Brooks. **"Eucalyptus: a web service-enabled e-infrastructure"**. In CASCON ’07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, New York, NY, USA, 2007. ACM, pages 1–11.

[19] E. Ciurana, **Developing with Google App Engine**. Apress, Berkely, CA, USA, 2009.

[20] Salesforce (2009). **"Salesforce"**, disponível em: http://www.salesforce.com/.

[21] A. Marinos e G. Briscoe, **“Community cloud computing”**, in Cloud Computing, Springer, 2009, p. 472–484.

[22] A. M. Jr, M. Laureano, A. Santin, e C. Maziero, **"Aspectos de segurança e privacidade em ambientes de Computação em Nuvem"**. X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Fortaleza, 2010.

[23] D. Zissis e D. Lekkas, **“Addressing cloud computing security issues”**, Future Gener. Comput. Syst., vol. 28, no 3, p. 583–592, 2012.

[24] S. Chaves , **“A questão dos riscos em ambientes de computação em nuvem”**, Dissertação de Mestrado, Universidade de São Paulo, São Paulo, SP, Brasil, 2011.

[25] SCHAFFER, Guilherme (2008) **“Virtualização de desktops: o que é e porque virtualizar”**. Disponível em: <http://www.baguete.com.br/blogs/post.php?id=4,149> Acesso em 18/07/2012.

[26] R. J. Creasy. **"The origin of the VM/370 time-sharing system"**, IBM Journal of Research & Development, Vol. 25, No. 5, pp. 483-490, September 1981.

[27] R. P. Goldberg, **“Architecture of virtual machines”**, in Proceedings of the workshop on virtual computer systems, ACM, 1973, p. 74–112.

[28] R. P. Goldberg e P. S. Mager, **“Virtual Machine Technology: A Bridge From Large Mainframes To Networks Of Small Computers”**, Compcon Fall 79 Proc., p. 210–213, 4.

[29] N. L. Kelem e R. J. Feiertag, **“A separation model for virtual machine monitors”**, Res. Secur. Priv. 1991 Proc. 1991 Ieee Comput. Soc. Symp., p. 78–86, 20.

[30] G. J. Popek e R. P. Goldberg, **“Formal requirements for virtualizable third generation architectures”**, Commun. Acm, vol. 17, no 7, p. 412–421, 1974.

[31] M. A. P. Laureano e C. A. Maziero, **“Virtualização: Conceitos e aplicações em segurança”**, Livro-Texto Minicursos Sbseg, p. 1–50, 2008.

[32] VMWare ESXi. Disponível em: <http://www.vmware.com/products/esxi>. Acessado em: 12/06/2012.

[33] IBM PowerVM. Disponível em: <http://www-03.ibm.com/systems/power/software/virtualization>. Acessado em 21/08/2012.

[34] Xen.org. Disponível em: <http://www.xen.org>. Acessado em 21/07/2012.

[35] VMWare Workstation. Disponível em: <http://www.vmware.com/products/ws>.

[36] Parallels Desktop. Disponível em: <http://www.parallels.com/products/desktop>.

[37] QEMU. Disponível em: <http://www.nongnu.org/qemu>. Acessado em: 23/05/2012.

[38] VirtualBox, I. (2008). **The VirtualBox architecture**. Disponível em: <http://www.virtualbox.org/wiki/VirtualBox_architecture>.

[39] ISO 27002. ABNT NBR ISO/IEC 27002:2005. **“Código de prática para a gestão da segurança da informação”**. Associação Brasileira de Normas Técnicas, 2005.

[40] Cloud Security Alliance (CSA). **Security Guidance for Critical Areas of Focus in Cloud Computing –Version 2.1.** Technical report, Cloud Security Alliance, Dez/2009. Disponível em: <http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf>.

[41] D. Catteddu, **Cloud Computing: benefits, risks and recommendations for information security.** Springer, 2010.

[42] Gartner, **Assessing the Security Risks of Cloud Computing**, June 2008. Disponível em: <http://www.gartner.com/DisplayDocument?id=685308/>.

[43] Cloud Security Alliance–CSA. **Top Threats to Cloud Computing V1.0**. Disponível em: <http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf/>. Acesso em: 12 abr. 2012.

[44] M. A. P. Laureano, C. A. Maziero, e E. Jamhour, **"Detecção de intrusão em máquinas virtuais"**, 5o Simpósio Segurança Em Informática–ssi São José Dos Campos, p. 1–7, 2003.

[45] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, e E. Vázquez, **"Anomaly-based network intrusion detection: Techniques, systems and challenges"**, Comput. Secur., vol. 28, no 1, p. 18–28, 2009.

[46] J. Allen, A. Christie, W. Fithen, J. McHugh, e J. Pickel, **"State of the practice of intrusion detection technologies"**, DTIC Document, 2000.

[47] T. H. Ptacek e T. N. Newsham, **"Insertion, evasion, and denial of service: Eluding network intrusion detection",** DTIC Document, 1998.

[48] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, e D. Zamboni, **"An architecture for intrusion detection using autonomous agents"**, in Computer Security Applications Conference, 1998. Proceedings. 14th Annual, 1998, p. 13–24.

[49] Santos, O. M. and Campello, R. S. **Estudo e implementação de mecanismos de tolerância a falhas em sistemas de detecção de intrusão**. Santa Maria, UNIFRA, 2001.

[50] R. Bace e P. Mell, **"NIST special publication on intrusion detection systems"**, DTIC Document, 2001.

[51] G. Ramachandran e D. Hart, **"A P2P intrusion detection system based on mobile agents"**, in Proceedings of the 42nd annual Southeast regional conference, 2004, p. 185–190.

[52] J. Li, B. Li, T. Wo, C. Hu, J. Huai, L. Liu, e K. P. Lam, **"CyberGuarder: A virtualization security assurance architecture for green cloud computing"**, Future Gener. Comput. Syst., vol. 28, no 2, p. 379–390, 2012.

[53] R. Ando, K. Byung, e Y. Kadobayashi, **“Log analysis of exploitation in cloud computing environment using automated reasoning”,** in Neural Information Processing. Models and Applications, Springer, 2010, p. 337–343.

[54] H. Fang, Y. Zhao, H. Zang, H. H. Huang, Y. Song, Y. Sun, e Z. Liu, **“VMGuard: an integrity monitoring system for management virtual machines”**, in Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on, 2010, p. 67–74.

[55] S. Bharadwaja, W. Sun, M. Niamat, e F. Shen, **“Collabra: a xen hypervisor based collaborative intrusion detection system”**, in Information Technology: New Generations (ITNG), 2011 Eighth International Conference on, 2011, p. 695–700.

[56] K. Vieira, A. Schulter, C. B. Westphall, e C. M. Westphall, **“Intrusion detection for grid and cloud computing”**, It Prof., vol. 12, no 4, p. 38–43, 2010.

[57] S.-F. Yang, W.-Y. Chen, e Y.-T. Wang, **“ICAS: An inter-VM IDS Log Cloud Analysis System”**, in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, 2011, p. 285–289.

[58] S. N. Dhage, B. B. Meshram, R. Rawat, S. Padawe, M. Paingaokar, e A. Misra, **“Intrusion detection system in cloud computing environment”**, in Proceedings of the International Conference & Workshop on Emerging Trends in Technology, 2011, p. 235–239.

[59] S. Roschke, F. Cheng, e C. Meinel, **“An extensible and virtualization-compatible IDS management architecture”**, in Information Assurance and Security, 2009. IAS’09. Fifth International Conference on, 2009, vol. 2, p. 130–134.

[60] T. Alharkan e P. Martin, **“IDSaaS: Intrusion Detection System as a Service in Public Clouds”**, in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012, p. 686–687.

[61] H. A. Kholidy e F. Baiardi, **“CIDS: A Framework for Intrusion Detection in Cloud Systems”**, in Information Technology: New Generations (ITNG), 2012 Ninth International Conference on, 2012, p. 379–385.

[62] S. E. Coull e B. K. Szymanski, **“Sequence alignment for masquerade detection”**, Comput. Stat. Data Anal., vol. 52, no 8, p. 4116–4131, 2008.

[63] J. Schmuller, **Sams teach yourself UML in 24 hours.** Sams publishing, 2004.

[64] D. Pilone e N. Pitman, **UML 2.0 in a Nutshel**l. O’Reilly Media, Inc., 2009.

[65] Gilleanes T. A. Guedes. **“UML 2: Uma abordagem prática”**. Novatec Editora. São Paulo, 2009.

[66] D. Svantesson e R. Clarke, **“Privacy and consumer risks in cloud computing”**, Comput. Law Secur. Rev., vol. 26, no 4, p. 391–397, 2010.

[67] ZAIDI, A. **Recherche et détection des patterns d'attaques dans les réseaux IP à haut débits**. Tese (Doutorado)— Université d'Évry Val d'Essonne, Évry, 2011. 109 f.

[68] THIAGO, V. S. **Arquitetura Multi-agentes para Detecção Distribuída de Intrusão**. Vinícius da Silva Thiago. Dissertação (mestrado).UFC, 2012. 102p.

[69] L. Siqueira e Z. Abdelouahab, **"A fault tolerance mechanism for network intrusion detection system based on intelligent agents (NIDIA)"**, in Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on, 2006, p. 6 pp.

[70] F. A. Pestana (2005). **Proposta de atualização automática dos sistemas de detecção de intrusão por meio de web services**. In Dissertação de Mestrado. Coordenação de Pós-Graduação em Engenharia de Eletricidade. Universidade Federal do Maranhão, São Luís, Maranhão, Brasil.

[71] R. L. da Rocha Ataide, **"Uma arquitetura para a detecção de intrusos no ambiente wireless usando redes neurais artificiais"**, Dissertação de Mestrado, PPGEE/UFMA, 2007.

[72] KVM – **Kernel Based Virtual Machine**. Red Hat. White Paper. http://www.redhat.com. (2009).

[73] Website oficial libvirt,http://www.libvirt.org/, acessado em 29 de abril de 2012.

[74] Virtual Machine Manager, http://virt-manager.org/, acessado em 11 de maio de 2012.

[75] JPcap. **"Jpcap - a Java library for capturing and sending network packets"**. Disponível em: <http://netresearch.ics.uci.edu/kfujii/jpcap/doc/>. Acessado em 14 de Agosto de 2012.

[76] Libpcap. TCPDUMP/LIBPCAP public repository. Disponível em: http://www.tcpdump.org/. Acessado em 09 de abril de 2012.

[77] Sanfilippo, S. (2001) **"Hping"**, http://www.hping.org, Dez. 2002.

[78] MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, **George. Hackers Expostos: Segredos e soluções para a segurança de redes**. 2. ed. São Paulo: Makron Books, 2000.

[79] K. Kendall, **"A database of Computer Attacks for the Evaluation of Intrusion Detection Systems"**. Massachusetts, 1999. Dissertação (Mestrado em Computação), Massachusetts Institute of Technology, 1999.

[80] SOUSA, Bruno. F. **Agentes Inteligentes para a Detecção e Tratamento de Ataques a Redes de Computadores**. São Luís, 2001. Monografia (Graduação em Ciência da Computação), Universidade Federal do Maranhão – UFMA, 2001.

[81] Sam Johnston, **Introducing the cloud computing stack**. Disponivel em: <http://samj.net/2009/04/introducing-cloudcomputing-stack-2009.html>.

[82] P. RENATO, **Backtrack: Solução open source para pen test**. Disponível em: <http://www.slideshare.net/pseixas/backtrack-soluco-open-source-para-pen-test>. Acesso: 12 mar. 2013.

# APÊNDICES

# APÊNDICE A — Códigos implementados

Com o objetivo de demonstrar o funcionamento do IDS proposto, alguns dos principais trechos de código dos componentes que compõem o sistema de detecção foram selecionados.

## A.1 IDS_Admin

A seguir são mostrados trechos do código implementado para o IDS_Admin.

```
1   public class Admin {
2       .
3       .
4       .
5
6       public static void main(String[] args) {
7           if (args.length < 1) {
8               System.out.println("usage: java Admin [porta]");
9               return;
10          }
11
12          System.out.println("Initializing Admin...\n");
13
14          try {
15              ServerSocket servidor = new ServerSocket(2020);
16              System.out.println("Waiting for connection...");
17
18              while (true) {
19                  Socket sock_node = servidor.accept();
20                  System.out.println("Connected at "
21                          + sock_node.getInetAddress().getHostAddress());
22                  InputStream input = sock_node.getInputStream();
23                  System.out.println("\nWaiting for data...\n");
24                  while (sock_node.isConnected()) {
25                      Scanner s = new Scanner(input);
26                      String alerta;
27
28                      fos_arq_ip = new FileOutputStream(arq_ips);
29                      buf_arq_ip = new BufferedOutputStream(fos_arq_ip);
30                      dados_arq_ip = new DataOutputStream(buf_arq_ip);
31
32                      while (s.hasNextLine()) {
33                          alerta = s.nextLine();
34                          if (alerta.length() > 0) {
35                              System.out.println(alerta);
36                              dados_arq_ip.writeBytes(alerta + "\n");
37                              buf_arq_ip.flush();
38                          .
39                          .
40                          .
```

**Figura A.1:** Trecho de código do IDS_Admin

## A.2 Start_IDS

A seguir são mostrados trechos do código implementado para o Start_IDS.

```
1  public class start_ids {
2      private static BufferedReader buf_ips_ativos;
3
4      public static void main(String[] args) {
5          if (args.length < 2) {
6              System.out
7                      .println("usage: java start_ids [idx placa] [qtd pacotes]");
8              return;
9          }
10         System.out.println("Start_IDS");
11         System.out.println("Waiting for Node to start...\n");
12         try {
13             buf_ips_ativos = new BufferedReader(new FileReader(
14                     "/ids/arq_ips.txt"));
15
16             // habilita porta para conexão com o Node (cliente)
17             ServerSocket servidor = new ServerSocket(2014);
18             // aceita conexão com o cliente
19             Socket cliente = servidor.accept();
20
21             // executa o pool para realizar a verificação de ips ativos
22             Thread trpool = new Thread(new Pool());
23             trpool.start();
24             trpool.join();
25
26             String ip;
27                 if (buf_ips_ativos.ready()) {
28                     if ((ip = buf_ips_ativos.readLine()) != null) {
29                         if (ip != null) {
30                         while (cliente.isConnected()) {
31                             Netsensor net = new Netsensor(
32                                     Integer.parseInt(args[0]),
33                                     Integer.parseInt(args[1]));
34                             Thread tn = new Thread(net);
35                             tn.start();
36                             tn.join();
37                         }
38                     }
39                     Thread trpool2 = new Thread(new Pool());
40                     trpool2.start();
41                     trpool2.join();
42                 .
43                 .
44                 .
```

**Figura A.2:** Trecho de código do Start_IDS

## A.3 Netsensor

A seguir são mostrados trechos do código implementado para o Netsensor.

```
public class Netsensor implements Runnable {
    class Handler_packets implements PacketReceiver {
        String msg = "", tipo = "";
        char sep = ';';
        int contador = 0;
        boolean started = false;

        public void receivePacket(Packet packet) {
            .
            .
            .

            msg = String.format(date + sep);

            if (packet instanceof TCPPacket) {
                TCPPacket tcp = (TCPPacket) packet;
                tipo = "tcp";
                msg += String.format("tcp" + sep + tcp.src_ip.getHostAddress()
                        + sep + tcp.src_port + sep
                        + tcp.dst_ip.getHostAddress() + sep + tcp.dst_port
                        + sep + tcp.syn + sep + tcp.ack + sep + tcp.fin);

            } else if (packet instanceof UDPPacket) {
                UDPPacket udp = (UDPPacket) packet;
                tipo = "udp";
                msg += String.format("udp" + sep + udp.src_ip.getHostAddress()
                        + sep + udp.src_port + sep
                        + udp.dst_ip.getHostAddress() + sep + udp.dst_port
                        + sep + udp.length);

            } else if (packet instanceof ICMPPacket) {
                ICMPPacket icmp = (ICMPPacket) packet;
                tipo = "icmp";
                msg += String.format("icmp" + sep
                        + icmp.src_ip.getHostAddress() + sep
                        + icmp.dst_ip.getHostAddress() + sep + icmp.type + sep
                        + icmp.code);

            } else if (packet instanceof ARPPacket) {
                ARPPacket arp = (ARPPacket) packet;
                tipo = "arp";
                msg += String.format("arp" + sep + arp.header);
            }

            System.out.print("\t" + contador++ + tipo);
            if (contador % 5 == 0) {
                System.out.println("");
            }
            analisar_pacote(msg);
            try {
                PrintStream p = new PrintStream(new BufferedOutputStream(
                        new FileOutputStream("/ids/bd_cap.txt", true)));
                p.println(msg);
                p.close();
            } catch (IOException e1) {
                System.out
                        .println("Erro ao escrever no arquivo de banco de captura.");
                e1.printStackTrace();
            }
        }
```

**Figura A.3:** Trecho de código do Netsensor. Parte 1

```
62          public void main(String[] args) throws java.io.IOException {
63              NetworkInterface[] devices = JpcapCaptor.getDeviceList();
64
65              try {
66                  JpcapCaptor captor = JpcapCaptor.openDevice(devices[index],
67                          65535, true, 20);
68                  captor.setFilter("ip", true);
69                  captor.loopPacket(qtd_packs, this);
70              } catch (IOException ioe) {
71                  System.out.println("Erro em captor.");
72                  System.out.println(ioe.getMessage());
73              }
74          }
75      }
76
77      public void analisar_pacote(String pacote) {
78          String pack[], protocol, alerta = null;
79
80          pack = pacote.split("\\;");
81          protocol = pack[1];
82
83          if (protocol.equals("tcp")) {
84              alerta = ana.analise_pacote_tcp(pack);
85          } else if (protocol.equals("udp")) {
86              alerta = ana.analise_pacote_udp(pack);
87          } else if (protocol.equals("icmp")) {
88              alerta = ana.analise_pacote_icmp(pack);
89          }
90
91          if (alerta != null) {
92              ana.envia_node(alerta);
93              ana.save(alerta);
94          } else {
95              ana.envia_node(protocol);
96          }
97      }
98      .
99      .
100     .
```

**Figura A.4:** Trecho de código do Netsensor. Parte 2

## A.4 IDS_Node

A seguir são mostrados trechos do código implementado para o IDS_Node.

```
1   public class Node {
2       .
3       .
4       .
5
6       public static void main(String args[]) {
7           if (args.length < 1) {
8               System.out.println("usage: java packet_ids/Node [nome deste node]");
9               return;
10          }
11          System.out.println("Initializing Node...\n");
12
13          // servidor para o Analyser
14          try {
15              sock_admin = new Socket("192.168.27.53", 2020);
16              System.out.println("Conectado a Admin.");
17              sock_start = new Socket("localhost", 2014);
18              System.out.println("Conectado a Start_ids.");
19              saida_to_admin = new PrintStream(sock_admin.getOutputStream());
20
21              while (true) {
22                  ServerSocket servidor = new ServerSocket(2013);
23                  Socket sock_ana;
24                  while ((sock_ana = servidor.accept()) != null) {
25                      TrataAnalyser tc = new TrataAnalyser(
26                              sock_ana.getInputStream(), args[0]);
27                      new Thread(tc).start();
28                  }
29                  servidor.close();
30              }
31          } catch (IOException e) {
32              e.printStackTrace();
33          }
34      }
35
36      private static void anomalia(String tipo_pack, String ip_vit) {
37          .
38          .
39          .
40
41          if (tipo_pack.equals("tcp")) {
42              contador_tcp++;
43
44              if (contador_tcp > 800) {
45                  if (!attack_tcp) {
46                      saida_to_admin.println(date + ":"
47                              + " Ataque DoS (TCP) na rede em " + ip_vit + ".");
48                      attack_tcp = true;
49                  }
50              } else if (contador_tcp > 300) {
51                  if (!susp_tcp) {
52                      saida_to_admin.println(date + ":"
53                              + " Trafego suspeito TCP na rede em " + ip_vit
54                              + "." + " Enviar pacotes para o NIDIA.");
55                      susp_tcp = true;
56                  }
57              }
```

**Figura A.5:** Trecho de código do IDS_Node. Parte 1

```
59          } else if (tipo_pack.equals("udp")) {
60              contador_udp++;
61
62              if (contador_udp > 800) {
63                  if (!attack_udp) {
64                      saida_to_admin.println(date + ":"
65                              + " Ataque DoS (UDP) na rede em " + ip_vit + ".");
66                      attack_udp = true;
67                  }
68              } else if (contador_udp > 300) {
69                  if (!susp_udp) {
70                      saida_to_admin.println(date + ":"
71                              + " Trafego suspeito UDP na rede em " + ip_vit
72                              + "." + " Enviar pacotes para o NIDIA.");
73                      susp_udp = true;
74                  }
75              }
76          } else if (tipo_pack.equals("icmp")) {
77              contador_icmp++;
78
79              if (contador_icmp > 800) {
80                  if (!attack_icmp) {
81                      saida_to_admin.println(date + ":"
82                              + " Ataque DoS (ICMP) na rede em " + ip_vit + ".");
83                      saida_to_admin.println("ICMP Flooding!");
84                      attack_icmp = true;
85                  }
86              } else if (contador_icmp > 300) {
87                  if (!susp_icmp) {
88                      saida_to_admin.println(date + ":"
89                              + " Trafego suspeito ICMP na rede em " + ip_vit
90                              + "." + " Enviar pacotes para o NIDIA.");
91                      susp_icmp = true;
```

**Figura A.6:** Trecho de código do IDS_Node. Parte 2

```
98      static class TrataAnalyser implements Runnable {
99          private InputStream cliente;
100         private String ip_ids;
101
102         public TrataAnalyser(InputStream cliente, String ip_cliente) {
103             this.cliente = cliente;
104             this.ip_ids = ip_cliente;
105         }
106
107         public void run() {
108             String mensagem;
109             Scanner s = new Scanner(this.cliente);
110
111             while (s.hasNextLine()) {
112                 mensagem = s.nextLine();
113                 if (mensagem.length() > 0) {
114                     if (mensagem.equals("*")) {
115                         // se o netsensor reinicializou
116                         // zera os contadores
117                         contador_tcp = 0;
118                         contador_udp = 0;
119                         contador_icmp = 0;
120                         susp_tcp = false;
121                         attack_tcp = false;
122                         susp_udp = false;
123                         attack_udp = false;
124                         susp_icmp = false;
125                         attack_icmp = false;
126                     } else if (mensagem.length() == 3 || mensagem.length() == 4) {
127                         // se chegar um pacote normal
128                         anomalia(mensagem, this.ip_ids);
129                     } else {
130                         // se for alerta
131                         saida_to_admin.println(mensagem);
132         .
133         .
134         .
```

**Figura A.7:** Trecho de código do IDS_Node. Parte 3

## A.5 IDS_Pool

A seguir são mostrados trechos do código implementado para o IDS_Pool.

```
1   public class Pool implements Runnable {
2       .
3       .
4       .
5
6       public void run() {
7           try {
8               String[] cut;
9               Process process = run
10                      .exec("arp-scan --interface=virbr5 --localnet");
11              if (process == null) {
12                  System.out.println("Pool dont started.");
13              } else {
14                  System.out.println("Pool alive...");
15              }
16              BufferedReader in = new BufferedReader(new InputStreamReader(
17                      process.getInputStream()));
18              String line;
19              fos_arq_ip = new FileOutputStream(arq_ips);
20              buf_arq_ip = new BufferedOutputStream(fos_arq_ip);
21              dados_arq_ip = new DataOutputStream(buf_arq_ip);
22
23              while ((line = in.readLine()) != null) {
24                  if (line != null) {
25                      cut = line.split("\\s");
26                      if (cut[0].startsWith("10.")) {
27                          dados_arq_ip.writeBytes(cut[0] + "\n");
28                          System.out.println(cut[0]);
29                          buf_arq_ip.flush();
30                      }
31                  }
32              }
33      .
34      .
35      .
```

**Figura A.8:** Trecho de código do IDS_Pool

## A.6 IDS_Analyser

A seguir são mostrados trechos do código implementado para o IDS_Analyser.

```
130     public void envia_node(String mensagem) {
131         String ip_admin = "localhost";
132         int porta = 2013;
133         Socket _node;
134         PrintStream saida_node;
135
136         try {
137             _node = new Socket(ip_admin, porta);
138             saida_node = new PrintStream(_node.getOutputStream());
139             saida_node.println(mensagem);
140         } catch (IOException e) {
141             e.printStackTrace();
142         }
143     }
```

**Figura A.9:** Trecho de código do IDS_Analyser. Parte 1

```
public class Analyser {

    .
    .
    .

    public String analise_pacote_tcp(String[] pack) {
        String src_ip, src_port, dest_ip, dest_port;
        String tcp_syn, tcp_ack, tcp_fin, tcp_data;
        String ip_da_rede = null; // ip atual da rede interna
        boolean eh_ip_rede = false; // if src_ip eh ip ativo da rede interna

        open_files();

        .
        .
        .

        try {
            if (pack.length > 9) {
                System.out.println("Tamanho do TCP packet invalido.");
                return null;
            }
            src_ip = pack[2];
            src_port = pack[3];
            dest_ip = pack[4];
            dest_port = pack[5];
            tcp_syn = pack[6];
            tcp_ack = pack[7];
            tcp_fin = pack[8];
            // tcp_data = pack[8];

            if (buf_ips_ativos.ready()) {
                while ((ip_da_rede = buf_ips_ativos.readLine()) != null) {
                    if (ip_da_rede != null) {
                        if (src_ip.equals(ip_da_rede)) {
                            eh_ip_rede = true;
                            break;
                        }
                    }
                }
            }

            .
            .
            .

        } catch (ArrayIndexOutOfBoundsException | IOException e) {
            e.printStackTrace();
        }

        close_files();
        return null;
    }// end analise_pacote_tcp method

    public String analise_pacote_udp(String[] pack) {
        return null;
    }

    public String analise_pacote_icmp(String[] pack) {
        String src_ip, dest_ip, type, code;
        String ip_broad = null;// ip de broadcast corrente
        boolean eh_ip_broad = false; // if dest_ip eh ip broad
        String ip_da_rede = null; // ip atual da rede interna
        boolean eh_ip_rede = false; // if src_ip eh ip ativo da rede interna
```

**Figura A.10:** Trecho de código do IDS_Analyser. Parte 2

```
67          open_files();
68
69          .
70          .
71          .
72
73          try {
74              if (pack.length > 6) {
75                  System.out.println("Tamanho do ICMP packet inválido.");
76                  return null;
77              }
78              src_ip = pack[2];
79              dest_ip = pack[3];
80              type = pack[4];
81              code = pack[5];
82
83              // verifica se o ip de origem eh ip da rede ativo da rede interna
84              if (buf_ips_ativos.ready()) {
85                  while ((ip_da_rede = buf_ips_ativos.readLine()) != null) {
86                      if (ip_da_rede != null) {
87                          if (src_ip.equals(ip_da_rede)) {
88                              eh_ip_rede = true;
89                              break;
90                          }
91                      }
92                  }
93              }
94
95              // verifica se o ip de destino eh ip broadcast
96              if (buf_broadcast.ready()) {
97                  while ((ip_broad = buf_broadcast.readLine()) != null) {
98                      if (ip_broad != null) {
99                          if (dest_ip.equals(ip_broad) || dest_ip.endsWith("255")) {
100                             eh_ip_broad = true;
101                             break;
102                         }
103                     }
104                 }
105             }
106
107             .
108             .
109             .
110
111         } catch (IOException e) {
112             e.printStackTrace();
113         }
114         close_files();
115         return null;
116     }// end detectar ataque icmp
117
118     public void save(String alerta) {
119         try {
120             PrintStream p = new PrintStream(new BufferedOutputStream(
121                     new FileOutputStream("/ids/bd_alertas.txt", true)));
122             p.println(alerta);
123             p.close();
124         } catch (IOException e1) {
125             System.out.println("Erro na escrita no banco de alertas.");
126             e1.printStackTrace();
127         }
128     }
```

**Figura A.11:** Trecho de código do IDS_Analyser. Parte 3

# APÊNDICE B — Ataques de rede

A segurança física é um dos princípios fundamentais de segurança em ambientes computacionais. Caso um atacante obtenha acesso direto a uma máquina com informações importantes e que deveriam ser sigilosas, é questão de tempo para o comprometimento da segurança. Se a segurança da máquina é garantida, então a rede que a interliga a outras máquinas passa a ser o alvo de possíveis ataques, onde várias tentativas em encontrar vulnerabilidades serão realizadas, muitas vezes despercebidas pelo administrador de rede.

Diferentes protocolos de comunicação servem como alvo em vários tipos de ataques, porém com a consolidação do TCP/IP como padrão de comunicação entre redes, como é o caso da Internet, o torna o protocolo mais utilizado, sendo que o estudo dos incidentes de segurança que utilizam o TCP/IP de extrema importância.

Um pacote TCP/IP é formado por protocolos da camada de rede e de transporte que compõe esta arquitetura. Observando os protocolos de rede IP e o de transporte TCP[1] temos o seguinte esquema:

- Na composição de um **datagrama** IP temos um cabeçalho, que contém informações de controle, e um campo de dados, que encapsula o protocolo de transporte (TCP);
- Analogamente, um pacote TCP também é composto por duas partes que são o cabeçalho, com informaçãoes de controle, e o campo de dados ou payload, que carrega as informações propriamente ditas que devem ser trocadas entre dois elementos da rede.

.

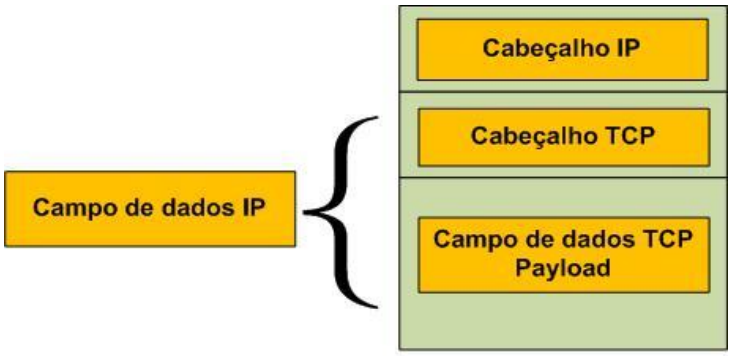



**Figura B.1:** Simplificação de um datagrama IP contendo um pacote TCP no seu campo de dados

[1] O protocolo não orientado à conexão UDP também serve de suporte a uma gama de ataques.

De acordo com a estrutura do pacote TCP/IP mostrado na Figura A.1, podemos classificar os ataques de rede em duas grandes categorias: ataques baseados em cabeçalho e ataques baseados em conteúdo.

## B.1 Ataques baseados em cabeçalho

As seguintes ameaças utilizam os cabeçalhos de rede e de transporte:

- Mapeamento da Rede (*Probe*) - antes de executar qualqer tipo de ataque, *hackers* experientes realizam uma fase de reconhecimento, na tentativa de identificar possíveis vulnerabilidades na rede da vítima. Através desta sondagem o atacante pode identificar quais são as máquinas ativas e seus respectivos serviços disponíveis, podendo assim efetuar uma investida mais qualificada e específica em cima de alvos realmente existentes. Este mapeamento é composto de duas fases: a varredura da faixa de endereços IP para descobrir quais máquinas são respondentes e a varredura de portas para identificar quais serviços estão ativos nestas máquinas [romulo].

- Negação de Serviço (*Denial of Service* – DOS) - segundo [78], “um ataque DOS interrompe ou nega completamente serviço a usuários legítimos, redes, sistemas ou outros recursos”. Como variedades de DOS temos os ataques que abusam de ações legítimas (ex: envio de pacotes ao extremo na tentativa de exaurir ou ocasionar alguma falha no sistema alvo) e ataques que enviam pacotes com cabeçalhos mal formados, causando perturbações no funcionamento da pilha TCP/IP da máquina vítima [79]. Estas duas classes de DOS são explicadas com mais detalhes nas seções seguintes.

### B.1.1 Negação de serviço através de pacotes mal-formados

Existem regras para as informações que trafegam nos cabeçalhos IP e TCP, assim como diferentes implementações da pilha do protocolo TCP/IP. A exploração de vulnerabilidades é consequência de erros na implementação, que por sua vez são explorados por pacotes cujos cabeçalhos não seguem as regras definidas, fugindo do que seria considerado um tráfego normal. Ataques como a negação de serviço (DOS) utilizam desse preceito. Esse tipo de ataque

é ocasionado pelo falta de tratamento a pacotes com cabeçalho inesperado e/ou malformado.

Seguem abaixo alguns ataques de recusa de serviço que possuem esta particularidade:

**Ataque Land** – Esse tipo de ataque ocorre quando no cabeçalho do pacote o par IP/Porta aparece tanto no endereço da máquina origem, quanto da máquina destino. De acordo [80], “algumas versões de sistemas operacionais, como o *Windows* 95 e *Windows* NT *Workstation* com *Service Pack* 3, têm suas máquinas travadas quando sofrem esse tipo de ataque pela porta 139 e as distribuições mais antigas do Linux apresentam queda de desempenho quando várias cópias deste ataque são disparadas contra ele”.

**Ataque Xmas Tree** – o cabeçalho TCP possui os seguintes bits de controle (flags) para o controle do fluxo das conexões:

FIN – finaliza a conexão enviando dados
SYN – Inicia uma conexão
PSH – envia dados para aplicação
RST – reseta a conexão
ACK – flag de confirmação de recebimento
URG – urgent pointer

Algumas combinações feitas a partir de flags específicas podem gerar um tráfego malicioso, quando o invasor forja os pacotes que as possuem. Uma vez que que não estariam presentes em um tráfego considerado normal. Por exemplo, antes de iniciar realmente a troca de informações, duas máquinas devem estabelecer primeiramente uma conexão. Para isso devem passar por um processo de negociação denominado *handshake* (Figura B.2).

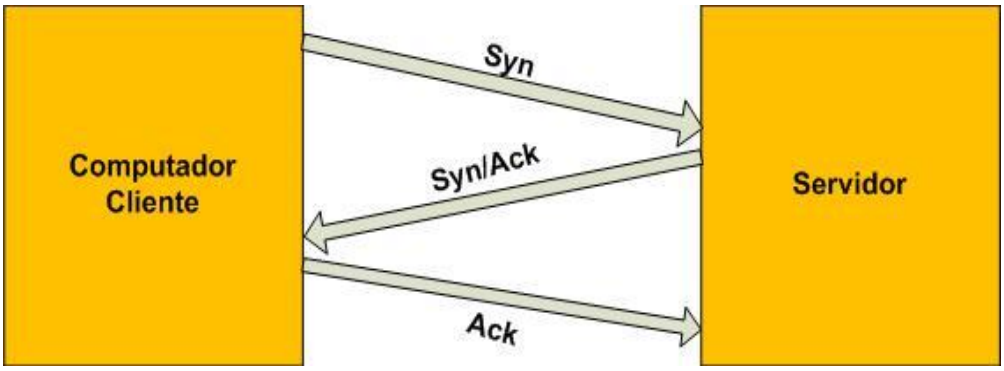



**Figura B.2**: Processo de *handshake*

O primeiro passo consiste no envio, da máquina cliente, do flag SYN ligado para o computador servidor para uma tentativa de conexão. No segundo passo, o servidor responde através de um pacote com os flags SYN e ACK ligados, indicando a sua disponibilidade para conexão. Por fim, o cliente envia um pacote com o flag ACK ligado para estabelecer conexão.

Qualquer combinação dos bits que não esteja dentro das regras definidas do protocolo TCP é tomada como suspeita. Dessa forma, por exemplo, um pacote forjado que tenha ao mesmo tempo os flags SYN e FYN ligados pode causar uma negação de serviço no computador para o qual foi enviado. Esta combinação não deveria normalmente acontecer, porque seria uma tentativa de simultaneamente iniciar e terminar uma conexão.

O ataque **XMAS Tree** utiliza do cabeçalho dos pacotes fora de padrão com os bits de flags ligados e compostos por combinações não usais, para gerar instabilidade nas máquinas.

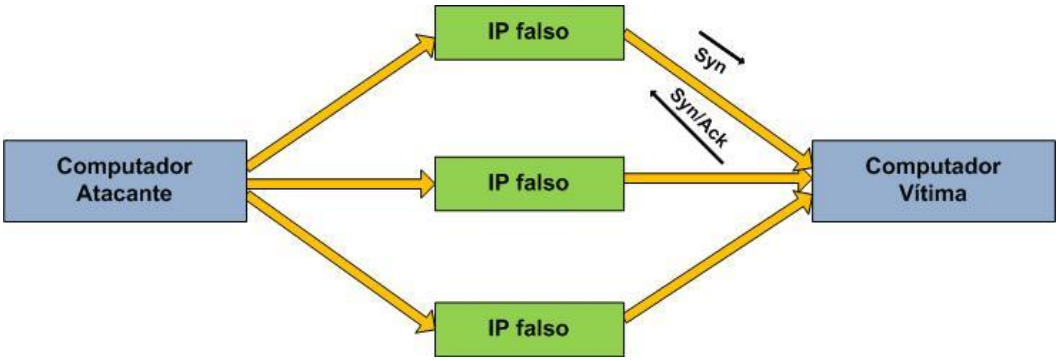
### B.1.2 Negação de serviço através do abuso de ações legítimas

Os ataques até agora vistos, utilizaram de pacotes fora de padrão, há outros ataques de cabeçalho que utilizam um grande número de pacotes considerados normais para gerar um ataque de DOS na máquina alvo. Isso consistem em submeter a máquina alvo a uma grande quantidade de pacotes em um curto espaço de tempo, fazendo com que a mesma fique tão ocupada reservando recursos para estas conexões que acabe recusando requisições de usuários legítimos.

Um ataque que usa dessa premissa é o **SynFlood**. Este utiliza uma técnica denominada *IP Spoofing*, onde o atacante altera o cabeçalho do pacote

modificando o IP de origem para um diferente do verdadeiro, caracterizando uma falsidade ideológica em relação do IP do host do atacante.

No ataque **SynFlood** o atacante envia inúmeros pacotes com o flag SYN ativado. Tais pacotes têm o seu endereço de origem alterado para números IPs não respondentes (*ip spoofing*). Desta forma, o processo de *handshake* não é completado e o computador da vítima manterá recursos alocados para estas conexões até que todos os recursos da máquina alvo sejam consumidos e a mesma recuse requisições de usuários reais (Figura B.3).




Pacotes com endereço IP origem alterados

**Figura B.3:** DOS SynFlood - computador atacante envia requisições de início de conexão através de pacotes com endereços IPs origem falsos