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

Eduardo Henrique de Carvalho Moura

Modelo de Segurança Autonômica para Computação em Nuvem com Uso de Honeypot

São Luís
2013

Eduardo Henrique de Carvalho Moura

Modelo de Segurança Autonômica para Computação em Nuvem com Uso de Honeypot

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

Orientador:
Prof. Ph.D. Zair Abdelouahab

São Luís
2013

Moura, Eduardo Henrique de Carvalho

Modelo de Segurança Autonômica para Computação em Nuvem com uso Honeypot/ Eduardo Henrique de Carvalho Moura.- São Luis, 2014.

92 f.

Impresso por computador (fotocópia).

Orientador: Zair Abdelouahab

Dissertação (Mestrado em Engenharia de Eletricidade) – Universidade Federal do Maranhão, 2014.

1. Computação em Nuvem 2. Segurança 3. Computação Autonômica 4.Metodologia de Decepção

CDU 004.77

MODELO DE SEGURANÇA AUTONÔMICA PARA COMPUTAÇÃO EM NUVEM COM USO DE HONEYPOT

Eduardo Henrique de Carvalho Moura

Dissertação aprovada em 26 de novembro de 2013.

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

Profa. Christiane Ferreira Lemos, Dra.
(Membro da Banca Examinadora)

Prof. Aristófanes Corrêa Silva, Dr.
(Membro da Banca Examinadora)

Aos meus pais Henrique Cézar e Delza Moura, minha querida esposa Olívia Maria pelo apoio incondicional.

Resumo

A Computação em Nuvem é um novo paradigma da computação que visa oferecer serviço sob demanda. Suas características como escalabilidade e disponibilidade de recursos infinitos vêm atraindo muitos usuários e empresas. Junto como eles vem também muitos usuários mal intencionados que querem se aproveitar dessa possibilidade de compartilhamento de recurso. Também migração de redes e servidores para nuvem significa que técnicas de invasão estão agora destinados a servidores baseados em nuvem . Ataques podem ser originados ate mesmo dentro do ambiente, quando uma de máquina virtual que esta sendo executada em uma de suas Vlans é utilizada para sondar, capturar dados ou inserir ataques a servidores que estão instanciados na nuvem. Tudo isso aliado a uma difícil administração devido à complexidade da infraestrutura do ambiente deixa a segurança sendo um ponto critico. A proposta desse trabalho é utilizar um framework autonômico juntamente com uma metodologia de decepção para propor um modelo segurança autonômica para nuvens computacionais que auxiliem na segurança de servidores e instâncias *works* contra ataques oriundos de outras instâncias.

Palavras Chaves: Computação em Nuvem, Segurança, Computação Autonômica, Metodologia de Decepção.

ABSTRACT

Cloud computing is a new computing paradigm which aims to provide on-demand service. Characteristics such as scalability and availability of infinite resources have attracted many users and companies. As they come along too many malicious users who want to take advantage of this possibility of resource sharing. Also migration networks and servers for cloud means hacking techniques are now destined to cloud-based servers. Attacks can originate until even within the environment, when a virtual machine that is being performed on one of his Vlans is used to probe, capture data or insert server attacks that are instantiated in the cloud. All this combined with a difficult to administer due to the complexity of the infrastructure leaves the safety of the environment to be a critical point. The purpose of this study is to use an autonomic framework with a methodology for disappointment to propose a security model for autonomic computing clouds that assist in the security of servers and instances works against attacks from other instances.

Keywords: Cloud Computing, Security, Autonomic Computing, Methodology of Deception.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado à oportunidade de cursar o mestrado assim como força para conclui-lo.

Ao Prof. Zair Abdelouhab, pelo tempo e paciência a mim dedicados sendo de fundamental importância para a conclusão desta dissertação.

A todo o Programa de Pós Graduação em Engenharia Elétrica.

A minha esposa Olivia Maria, pela compreensão e apoio.

Aos meus pais, que foram meu porto seguro nos momentos mais difíceis da vida.

Aos meus amigos de inicio de caminhada no mestrado, Mauro Melo e Josenilson Dias.

Aos meus colegas de mestrado, Renato, Gleidson, Dhuleane, Luís, Cláudio, Jesseildo, Rafael, Dejailson, Berto, Jessica, Ariel, Liana e Vladmir, pelo companheirismo e auxílio técnico indispensáveis.

Aos meus irmãos Emerson Bruno e Jessika Moura pela força e confiança em mim depositado.

O Willian Ribeiro que foi meu braço direito, dando grande auxilio para finalização de minhas pesquisas.

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

“No meio da dificuldade encontra-se a oportunidade”.

Albert Einstein

Lista de Figuras

Lista de Tabelas e Quadros

Lista de Siglas

T.I. - *Tecnologia da Informação.*

CSA - *Cloud Securyti Alliance.*

VM -*Virtual Machine*.

IDS - *intrusion detection system.*

CERT - *Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança.*

SMTP - *Simple Mail Transfer Protocol.*

TCP - *Transmission Control Protocol.*

UDP - *User Datagram Protocol.*

FTP - *File Transfer Protocol.*

SQL - **Structured Query Language.**

SSH - *Secure Shell.*

IaaS - *Infrastructure as a Service.*

PaaS - *Platform as a Service.*

SaaS - *Software as a Service.*

NIST- *National Institute of Standards and Technology.*

QoS - *Quality of service.*

SLA - *Services Level Agreement.*

IDaaS - *Identity as a Service.*

AWS - *Amazom Web Service.*

CA - *Computação Autonômica.*

ACL - *Access Control List*

LCS - *Longest Common Sub-string.*

HaaS - *Honeypot as a Service.*

VMM - *Virtual Machine Manager.*

FRE - *Filter and Redirection Engine.*

Vlan - Virtual *Local area network.*

EMI- *Eucalyptus Machine Images.*

ID- Identity.

XSS- Cross Site Scripting.

NIDS- *Network-based IDS*

Sumário

INTRODUÇÃO

A computação em nuvem ou do inglês “CloudComputing” é a tendência da computação. Com a computação em nuvem não será mais necessário a instalação vários softwares em seu dispositivo para realizar qualquer tipo de tarefa, desde a edição de imagens e vídeos até programas de escritórios, uma vez que tudo estará acessível por meio da internet como um serviço. Então, caso o usuário queira qualquer um desses serviços, basta cadastrar-se em um provedor do serviço que desejar e acessar sua nuvem, ao termino da sua atividade, pode salvar os dados gerados na própria nuvem e acessa-los de qualquer lugar, bastando apenas uma conexão com a internet e um hardware bem resumido.

Com a necessidade de construção de infraestruturas de Tecnologia da informação (T.I.) cada vez mais complexas, em que usuário é responsável pela instalação, configuração, atualização de software e a substituição do hardware rapidamente obsoleto, a computação em nuvem, com a terceirização de plataformas computacionais, é uma solução para diversos problemas enfrentados no modelo tradicional. Na computação em nuvem, os recursos de T.I. são fornecidos como um serviço, permitindo que os usuários o acessem sem a necessidade de conhecimento sobre a tecnologia utilizada. Assim, os usuários e as empresas passaram a acessar os serviços sob demanda e independente de localização, aumentando a quantidade de serviços disponíveis [1].

Grandes empresas como a Amazon, Google, Microsoft e outras, investem alto e estão a passos largos por acreditar nessa tendência. Devido a este alto investimento, o desenvolvimento de sistemas e serviços de computação em nuvem tem aumentado, dando mais opções de soluções disponíveis para os usuários e

empresas, atraindo-os cada vez mais a usar infraestruturas baseadas em Nuvem Computacional [2].

O que ainda restringe a adoção da computação em nuvem por grandes empresas é a desconfiança com a segurança, integridade, confidencialidade e disponibilidade dos dados e aplicações da nuvem. Para que a solução de computação em nuvem seja segura, várias decisões devem ser tomadas: qual tipo de nuvem deve ser implementada, como os dados serão armazenados e como administrar o acesso aos dados [4].

Para uma garantir segurança na nuvem, além de uma boa definição de papéis na nuvem e criação de politicas de grupo de acesso bem definidas outras estratégias para garantir a segurança da nuvem computacional vem sendo adotadas, tais como: uso de IDS (Sistemas de detecção de intrusão) [13], firewall e antivírus.

Mesmo assim, o problema de segurança é tão preocupante que a Cloud Security Alliance[1] (CSA) publicou um relatório mostrando as nove principais ameaças a computação em nuvem para 2013. Em primeiro lugar na lista ficou as violações de dados. A CSA cita um trabalho de pesquisa que descreve como uma máquina virtual poderia usar informações trafegadas dentro da nuvem para extrair chaves criptográficas privada sem uso por outras VMs (Virtual Machines) no mesmo servidor [18]. O que mostra a necessidade de estruturas que possam auxiliar na defesa da nuvem.

[1] A CSA foi fundada em dezembro de 2008, após uma ideia surgida durante o ISSA CISO Forum (Las Vegas, Novembro de 2008). A CSA é uma organização sem fins lucrativos, que visa promover as boas práticas de segurança em Computação em Nuvem.

Uma dessas estruturas seria a utilização de Honeypots juntamente com sistemas que dão características autonômicas para proteção da nuvem, como auxilio ao sistema de segurança. Tal estrutura já é utilizada em redes reais conforme [6, 7, 8] e recentemente utilizado para nuvens [5, 9] . A grande diferença da utilização dessas técnicas em um ambiente de nuvem seria o caráter distribuído e formas de bloqueio desse usuário mal intencionado às VMs que ele tenta acessar.

1.1 Definição do problema

A computação em nuvem vem se tornando cada vez mais um modo das grandes empresas compartilharem recursos valiosos e de forma rentável. Só em 2011 foram investidos cerca de 40 bilhões de dólares, sendo previsto um aumento de mais de 200 bilhões de dólares até 2020 [1]. Todo esse investimento torna a computação em nuvem alvo de várias ameaças de segurança, o que faz a segurança ser o seu principal problema.

O CERT[2] divulgou em seu site o número dos dados referentes a ataques reportados por colaboradores que apenas de janeiro a março de 2013 foram relatados 173.693 ataques a servidores.

[2] O 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.

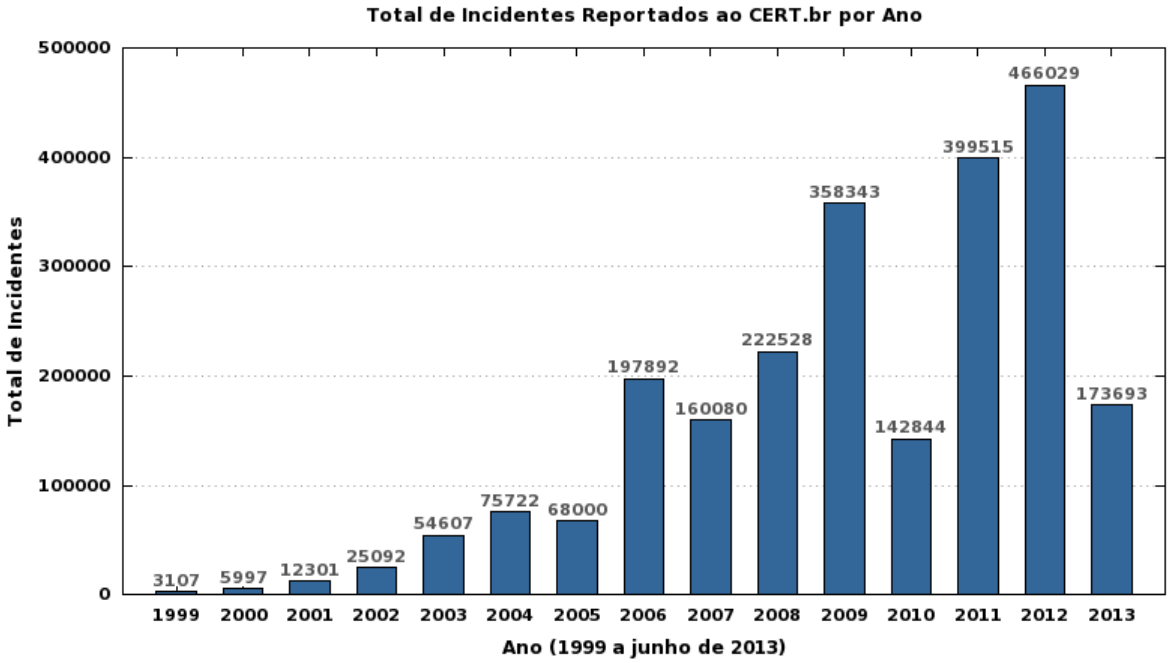

Figura 1.1: Estatísticas segundo o site CERT.br [48].

Segundo o CERT [48], as notificações sobre ataques a servidores web, varreduras e propagação de códigos maliciosos são respectivamente a segunda e a terceira posições no ranking de incidências de ataques a servidores, perdendo apenas para tentativas de fraudes, onde são criadas paraginas semelhantes as reais para capiturar as informações dos usuários. Mesmo com os ataques contra servidores web cresceram 32% em relação ao trimestre anterior, mas foram 43% menores em relação ao mesmo período de 2012.

Já as varreduras, que é quando pessoas mal intencionadas analizam as portas dos servidores afim de detectar serviços não utilizados porém com suas portas ativas, tornando-os muito sucetiveis a ataques, teve em 2013 seu número de notificações idêntico ao do segundo trimestre de 2012. As notificações de varreduras SMTP (Simple Mail Transfer Protocol) na porta 25 do protocolo TCP

(Transmission Control Protocol) continuam em destaque, atingindo 34,5% do total, no trimestre anterior elas atingiram 46,8% do total [48].

A computação em nuvem vem sofrendo dos mesmos problemas das redes de computadores tradicionais e seus servidores. Por isso, a administração de servidores de computação em nuvem e instâncias vem se tornando uma atividade muito complexa e árdua. Instâncias da nuvem podem sofrer ataques de usuários que utilizam suas instâncias para tentar capturar dados ou até mesmo deflagrar ataques, além de ataques externos oriundos da internet.

Como motivação principal, tem-se a constante evolução nas estratégias de ataques às nuvens computacionais. Deste modo, uma forma de entender tais ataques, seria a implantação de armadilhas (*Honeypots*) que atrairiam os ataques fazendo-lhes gerar *logs* da interação ocorrida e junto com estratégias autonômicas melhorariam a segurança no ambiente. Segundo [18], o *Honeypot* é uma tecnologia com enorme potencialidade na comunidade de segurança e pode ser usada de inúmeras formas dentro de um ambiente de computação em nuvem.

1.2 Solução proposta

A solução proposta é a união do uso de um honeypot com um framework autonômico para prover mais segurança às instâncias dentro da nuvem. Esta solução proporcionaria um ambiente no qual alguns tipos te ataques poderiam ser descobertos e sanados de maneira autonômica diminuindo assim a necessidade de constante intervenção humana no gerenciamento das instâncias.

Com isso ataques oriundos de outras instâncias da nuvem com intenções maliciosas poderia ter repostas mais rápidas, garantindo a segurança dos dados das instâncias, assim como da própria nuvem.

A figura 1.2 ilustra o modelo conceitual da topologia proposta, onde teremos a estrutura interna da nuvem com uma controladora da nuvem, com suas Vlans e instâncias (máquinas virtuais), e ligado a ela a controlodora dos honeyports e onde vai ficar o framework AutonomicSec-Cloud.

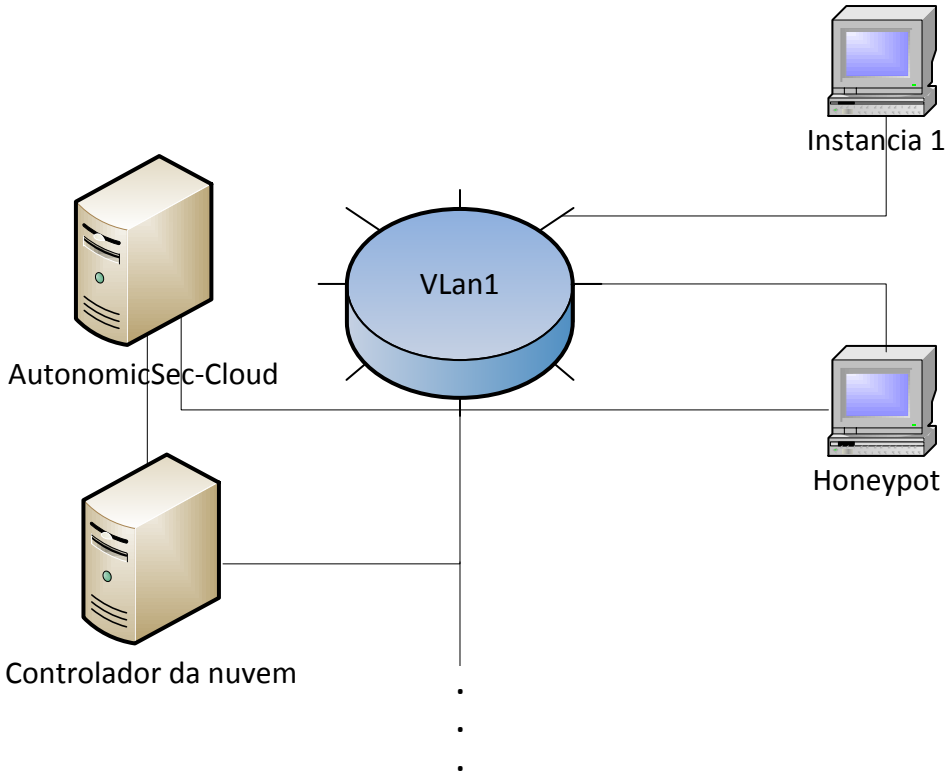

Figura 1.2: Modelo Conceitual da Topologia.

1.3 Metodologia utilizada

A metodologia utilizada para o desenvolvimento deste trabalho consistiu das seguintes etapas:

- Pesquisa bibliográfica sobre Computação Autonômica, utilização de metodologia de decepção e honeypots, computação em nuvem, segurança em sistemas de computação em nuvem, computação Autonômica aplicada a computação em nuvem, virtualização;
- Montagem de uma Nuvem Computacional para teste da topologia proposta. Composta por um microcomputador utilizando sistema

operacional Linux no qual foi instalado o midleware Eucalyptus que utiliza ferramentas da AMAZON EC2[3] que fornece um ambiente de computação em nuvem em todos os níveis de serviço, permitindo assim teste em um ambiente simulado de infraestrutura de nuvem real;

- Adaptação e extensão do framework AutonomicSec (desenvolvido para auxiliar na segurança de servidores em rede reais) a um ambiente de computação em nuvem e virtualização;
- Instalação e configuração de uma instancia Honeypot;
- Testes com o honeypot e com o framework.

1.4 Objetivos gerais e específicos

O objetivo deste trabalho é gerar um modelo autonômico para computação em nuvem utilizando logs gerados por honeypot, que possa ser disponibilizado como um serviço, atendendo às características desejadas em [19] para um sistema autonômico. Assim, o sistema terá a capacidade de autoconfiguração (self-configuration), auto-otimização (self-optimization), autocura (self-healing) e de autoproteção (self-protection).

No sentido de alcançar o objetivo geral, tem-se os seguintes objetivos específicos:

1. Desenvolver um modelo de sistema capaz de responder a ataques oriundos de instâncias da própria nuvem.
2. Desenvolver um modelo de *honeypot* para computação em nuvem.
3. Desenvolver um modelo de sistema autonômico para computação em nuvem.
4. Implementar Framework que possibilite a criação de ciclos autonômicos para computação em nuvem dando ênfase na segurança das instâncias dos clientes da nuvem e dando uma abordagem de IaaS.
5. Apresentar e demonstrar a utilização de computação em nuvem: técnicas e segurança.
6. Contribuir para aumentar o nível de segurança de redes.

[3] O Amazon Elastic Compute Cloud (Amazon EC2) é um serviço da Web que fornece uma capacidade de computação redimensionável na nuvem. Ele foi projetado para facilitar a computação de escala na Web para os desenvolvedores.

1.5 Estrutura da dissertação

Este trabalho está organizado da seguinte forma:

No Capítulo 2, é apresentada a Fundamentação teórica sobre os principais conceitos em computação em nuvem, segurança em computação em nuvem, computação autonômica e metodologia de decepção, além dos trabalhos relacionados que serviram de base para o trabalho.

No Capítulo 3 será apresentado o modelo de segurança autonômica para computação em nuvem baseado em *honeypot*.

No Capítulo 4 será apresentado a implementação do protótipo de teste, os testes e os resultados.

No Capítulo 5 será apresentada a conclusão deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capitulo serão apresentados os conceitos importantes para realização deste trabalho. Seguindo a ordem de apresentação primeiramente o conteúdo referente a computação em nuvem, seguido da segurança na computação em nuvem, metodologia de decepção e, por fim, o conteúdo referente a computação autonômica

2.1 Computação em nuvem

O termo computação em nuvem é designado para representar uma tecnologia que consiste em utilizar, em qualquer lugar e, independente dos sistemas operacionais nas máquinas dos usuários, variadas aplicações por meio da internet, com a mesma facilidade dos *softwares desktop* [20]. Tem como objetivo proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso [1].

A metáfora com o termo nuvem vem da representação comum em diagramas de rede, em que a figura de uma nuvem é usada para representar a internet, abstraindo, assim, o transporte de dados de *backbones* de transporte para uma localização *endpoint* (localização esta que fica na periferia da nuvem onde o usuário tem acesso) do outro lado da nuvem.

Em1961, o Professor John Mc Carthy sugeriu uma tecnologia de compartilhamento de tempo do computador, podendo levar a um futuro em que o poder de computação e até mesmo aplicações específicas poderiam ser vendidos através de um modelo de negócio do tipo utilitário [22]. Tal ideia foi deixada de lado devido a falta de tecnologia que a apoiasse na época sendo e com o passar do tempo o termo *Cloud Computing* retorna ao senário de T.I. como uma tendência da computação futura.

Esta tendência veio da necessidade de construção de infraestruturas de T.I. cada vez mais complexas e maiores. Sendo o usuário encarregado muitas vezes da atividade de instalação, configuração e atualização de *software*. Outro ponto que fortalece esta tendência e a constante evolução do *hardware* que se torna obsoleto e/ou desatualizado em pouco tempo, fazendo com que sua reposição e expansão demandem um custo elevado para as empresas. A simplificação da utilização dos

serviços por parte do usuário é outra grande força que impulsiona a computação em nuvem, os usuários necessitam ter apenas uma máquina com sistema operacional, um navegador e acesso a internet para acessar os recursos da nuvem, sem a necessidade de um alto poder computacional e liberando o *hardware* para executar tarefas que condizem com sua capacidade de processamento. Com isso o modelo de computação em nuvem trouxe como objetivo o fornecimento de serviços de fácil acesso, baixo custo e com garantias de disponibilidade e escalabilidade [1].

A nuvem é o conjunto de *hardware* e *software* dos grandes centros de dados. Quando essa infraestrutura é disponibilizada para o publico em geral na forma "*pay-as-you-go*", ou seja, pagando pelo que usar, dizemos que esta nuvem é uma nuvem pública. O serviço oferecido é o que chamamos de *Utility Computing* ou computação utilitária. Porém quando uma empresa tem sua própria nuvem, na qual apenas usuários da sua rede podem utilizar os serviços oferecidos por ela, é chamada de nuvem privada.

A nuvem trouxe três aspectos novos à computação do ponto de vista do custo para provisionamento de *hardware*. O primeiro é que com a nuvem os recursos computacionais disponíveis são infinitos, não tendo necessidade de planejamento por parte do usuário. Esse prévio provisionamento de *hardware*, isto fica a cargo do provedor da nuvem. Segundo a eliminação de um compromisso antecipado dos usuários da nuvem com o provedor, já que ele paga apenas pelo que ele usar e sempre que a demanda aumentar ele pode pagar pelo que aumentou. Em terceiro a métrica para o uso é normalmente calculada sobre horas ou dias de uso o que facilita a liberação do recurso assim que ele não seja mais necessário.

Com a tendência da *Green Computing* outro ponto importante é abordado pela computação em nuvem, que é a redução do consumo de energia elétrica, a largura de banda da rede, operações, *software* e *hardware* dando um caráter não apenas financeiro a redução, mas também social e ecológico [21].

Devido a tantos beneficio a computação em nuvem vem se tornado cada vez mais uma realidade dentro das grandes empresas e também entre os usuários domésticos, aumentando assim a quantidade de serviços disponibilizados com pagamento baseado no uso.

2.1.1 Modelo da computação em nuvem

A computação em nuvem tem seus serviços divididos em três classes as quais os recursos têm níveis de abstração e modelo de serviço do provedor diferente. O nível de abstração referisse à camada do sistema real que é abstraída na nuvem. As três classes de serviço são nomeadas como Infraestrutura como Serviço (IaaS), camada inferior; Plataforma como Serviço (PaaS), camada intermediária; e Software como Serviço (SaaS), camada superior [23].

Os modelos de serviço do provedor pedem ser representada em camadas, tendo como base do modelo a infraestrutura física, composta por todo o hardware (servidores, equipamentos de rede) e sistema operacional. Sobre a camada física fica a camada de virtualização, com os gerenciadores de virtualização e máquinas virtuais e sobre ela os Serviços oferecidos pela nuvem.

Na Figura 2.1 é mostrado modelo em camadas da arquitetura da computação em nuvem.

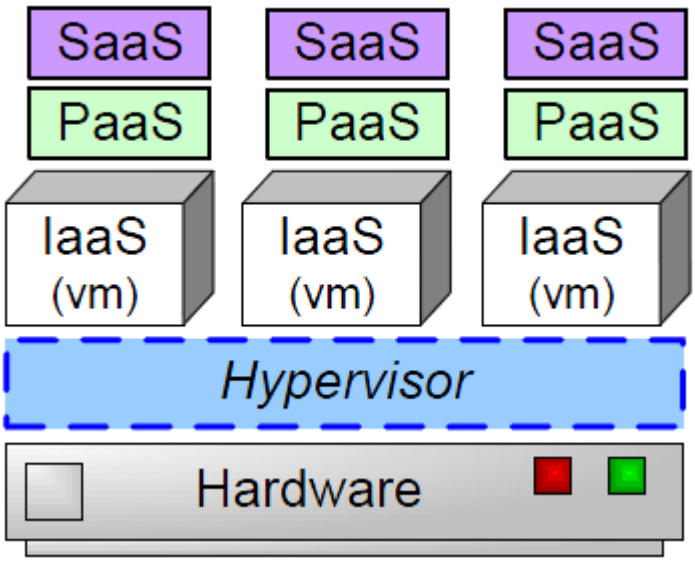

Figura 2.1: Modelo de Arquitetura da computação em nuvem [26].

Como mostra Sousa [1] o modelo de computação em nuvem definido pelo NIST[4] é composto por cinco características essenciais, três modelos de serviço e

[4]O NIST é uma agência governamental não-regulatória da administração de tecnologia do Departamento de Comércio dos Estados Unidos. A missão do instituto é promover a inovação e a

quatro modelos de implantação, como pode ser visto em [24] e como ilustra a Figura 2.2.

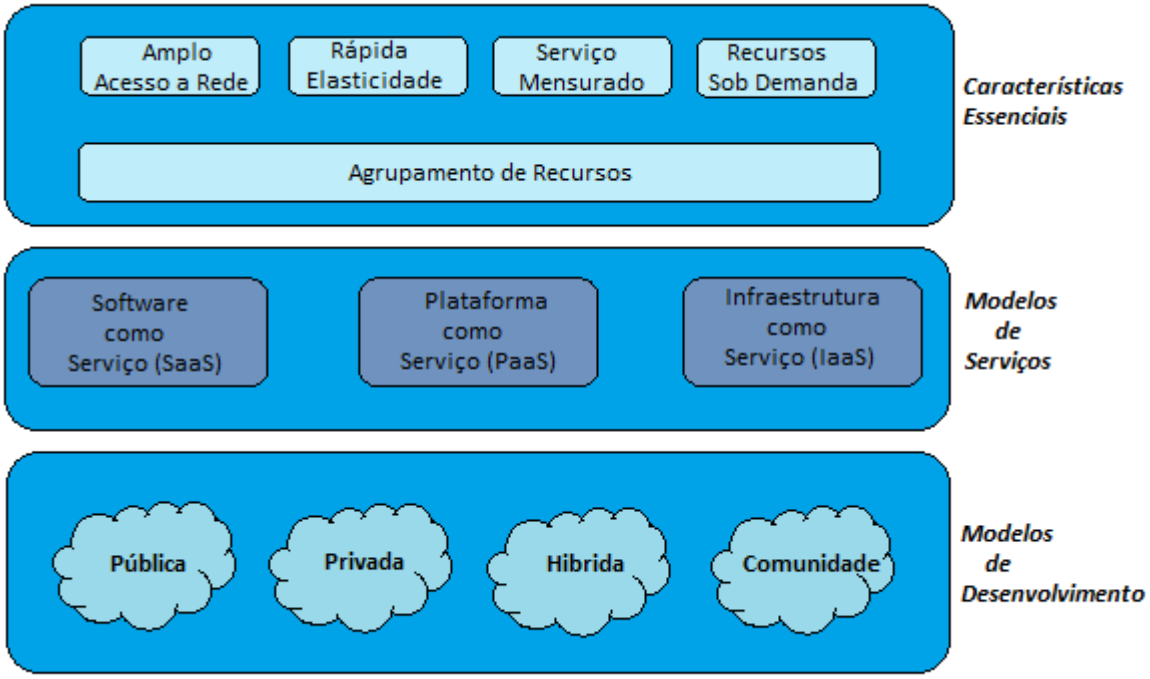

Figura 2.2: Modelo da Computação em Nuvem definido pelo NIST [23].

2.1.2 Características essenciais

A computação em nuvem deve trazer cinco características essenciais que atribui a ela, vantagens necessárias para ser uma opção competitiva e a distingue de outros paradigmas como mostra [1], [23] e [25]. Estas características são Self-service sob demanda (*On-demand self-service*), Amplo acesso (*Broad network access*), agrupamento de recursos (*Resource pooling*), Elasticidade rápida (*Rapid elasticity*) e Serviço medido (*Measured Service*).

No self-service sob demanda a infraestrutura da computação em nuvem deve ser capaz de fornecer recurso como tempo de processamento ou armazenamento em um servidor, de acordo com a necessidade do cliente e a demanda de utilização, sem que seja necessária interação humana entre o cliente e provedor do serviço. Todo esse ajuste de *hardware* e *software* deve ser feito de

---

competitividade industrial dos Estados Unidos, promovendo a metrologia, os padrões e a tecnologia de forma que ampliem a segurança econômica e melhorem a qualidade de vida.

maneira transparente para o usuário, afim de não interferir na utilização dos serviços contratados por ele.

Como amplo acesso 0s recursos da computação em nuvem tem uma das principais características o amplo acesso, onde o usuário deve ter acesso aos recursos oferecidos pela nuvem independente da plataforma utilizada ou sistema operacional, tendo como único fator limitante de acesso à rede ou a internet dependendo do modelo de nuvem adotado.

Segundo [1], a interface de acesso à nuvem não obriga os usuários a mudar suas condições e ambientes de trabalho, como por exemplo, linguagens de programação e sistema operacional, sendo que o acesso à nuvem feito através software clientes instalados localmente devem ser leves, como um navegador de Internet.

O Agrupamento de recursos é outra característica importante para os serviços *multi-tenant* ou multi-inquilino, o provedor de computação em nuvem necessita de um grupo organizado de recursos físicos e virtuais. Tais recursos devem ser atribuídos e ajustados dinamicamente de acordo com a demanda e a permissão dada ao usuário ou grupo de usuários. As localizações desses recursos devem ser desconhecidas para os usuários, assim o único controle sobre a localização do recurso era saber que ele estaria localizado no *backbone* do provedor contratado [1].

Elasticidade rápida é essencial para o provedor de computação em nuvem, ele tem que ser capaz de dispomibilizar qualquer quantidade do serviço oferecido, dependendo apenas da demanda do cliente, dando a ilusão de recursos infinitos. Esta expansão só é possível devido à tecnologia de virtualização aplicada a computação em nuvem [1].

Os recursos adicionais oferecidos ao cliente são rapidamente removidos após a diminuição da demanda, podendo ou ser parte de um controle autonômico do uso da nuvem [3].

O serviço baseado em consumo é outra característica da computação que atrai muita gente para esse novo paradigma. Como os serviços da nuvem são tarifados normalmente por horas, os serviços implementados para tarifação devem garantir um eficiente comércio de serviços, com a tarifação adequada, contabilidade, faturamento, monitoramento e otimização do uso [3].

A transparência no monitoramento e controle no uso dos recursos por parte do usuário deve ser uma busca constante do provedor de serviço. Isso pode ser garantido através de politicas de *Qualityof Service* (QoS), utilizando um abordagem baseada em acordo de nível de serviço (*Services Level Agreement*-SLA). Informações sobre os níveis de disponibilidade, funcionalidade, desempenho ou outros atributos do serviço como o faturamento e até mesmo penalidades em caso de violação destes níveis podem ser conseguidas com o SLA.

2.1.3 Modelos de prestação de serviço

O modelo de prestação de serviço em computação em nuvem é divididos normalmente em 3 classes dependendo do nível de abstração e do serviço a ser disponibilizado para o cliente [23]. Assim temos os modelos de *Software* como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestruturas como serviço (IaaS), esses são os modelos básicos de prestação de serviço, porém existe na literatura outros modelos que englobam também um outro serviço que seria a Identidade como um Serviço(IDaaS) [26].

A IDaaS é reconhecido pelo *Cloud Security Alliance*(CSA) como um serviço que fornece gerenciamento de identidade e do ciclo de vida dos usuários, funções de controle de acesso, etc. E pode ser usado juntamente com todos os outros modelos de serviço (SaaS, PaaS e IaaS) [26].

A Figura 2.3 mostra os modelos de serviços da computação em nuvem. Cada camada abstrai os serviços da camada logo abaixo e o IDaaS trabalha em conjunto com todas as camadas.

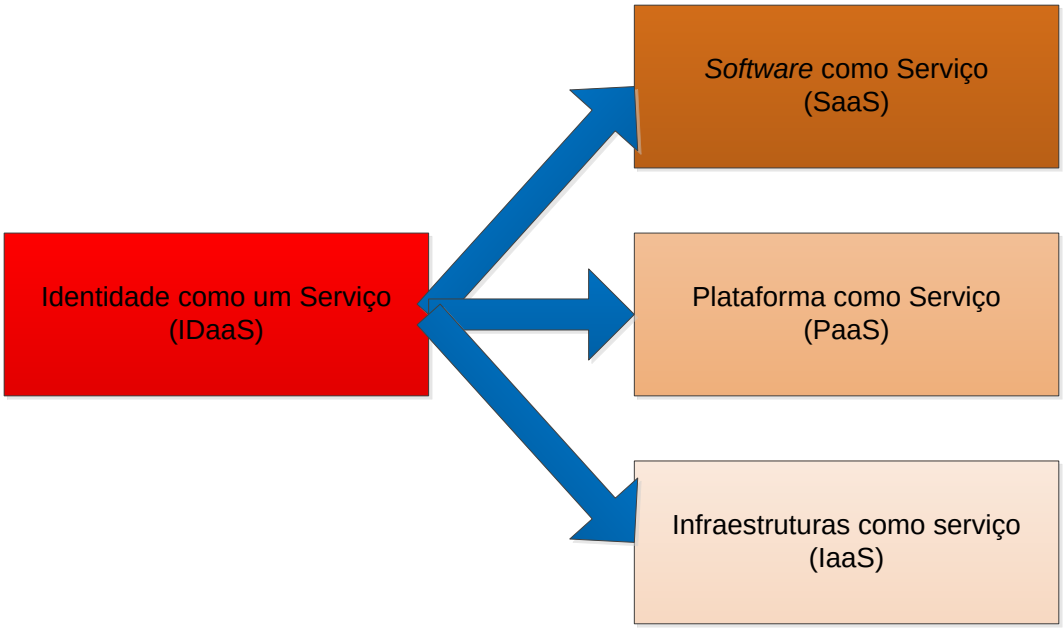

Figura 2.3: Modelos de Serviços da Computação em Nuvem.

2.1.3.1 Infraestrutura como Serviço (IaaS)

O modelo de prestação de serviço com menor nível de abstração é o IaaS, fornece ao cliente recursos computacionais como processamento, armazenamento, rede entre outros sendo responsável pela conectividade entre a infraestrutura física e lógica. O cliente pode utilizar diferentes arquiteturas de sistemas para implantar e executar uma variedade programas, sistemas operacionais e aplicativos, tendo controle limitado para gerenciar a infraestrutura física disponibilizada pela nuvem [26].

Devido ao uso de técnicas de virtualização dos recursos computacionais, o IaaS permite que a infraestrutura seja escalada dinamicamente, aumentando ou diminuindo os recursos de acordo com a demanda e necessidade das aplicações do cliente [1].

Há diversos fornecedores deste tipo serviço dentre os mais utilizados estão o Amazon EC2 (*ElasticCloudComputing*) e o Eucalyptus[5] (*Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*) [1].

---

[5] Eucalyptus é um software de código aberto para a construção da Amazon Web Services (AWS) compatíveis com ambientes de computação em nuvem privada e híbrida comercializados pela empresa Eucalyptus Systems [1].

Na Figura 2.4 é mostrada uma representação do modelo de infraestrutura como serviço.

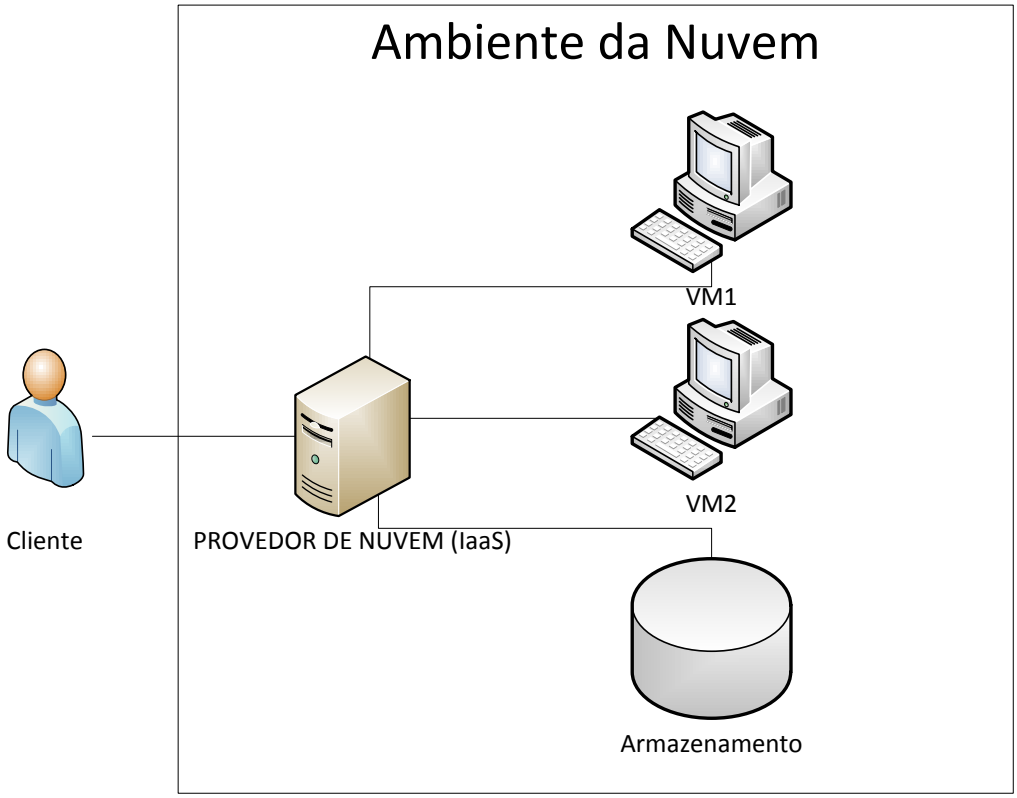

Figura 2.4: IaaS- Infraestrutura como serviço.

2.1.3.2 Plataforma como Serviço (PaaS)

O PaaS oferece uma camada intermediária com um ambiente no qual o desenvolvedor pode criar e implementar aplicações, sem a preocupação referente a processamento e quantidade de memória usada pela tarefa [23]. Sendo assim não faz parte do controle do usuário à infraestrutura de rede, servidores, sistemas operacionais ou armazenamento, porém cabe a ele a implantação e configuração das aplicações hospedadas [1].

Cabe então ao PaaS oferecer um sistema operacional, com linguagem de programação e ambiente de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de *software*, uma vez que contém ferramentas de desenvolvimento e colaboração entre desenvolvedores [1]. Temos o Google appEngine e Microsoft Azure como exemplos de PaaS no mercado [27].

As principais características de um PaaS incluem serviços para desenvolver, testar, implantar, hospedar e gerenciar aplicativos que suportem o ciclo de vida desenvolvimento de aplicativos [28]. Na Figura 2.5 mostra um modelo de PaaS.

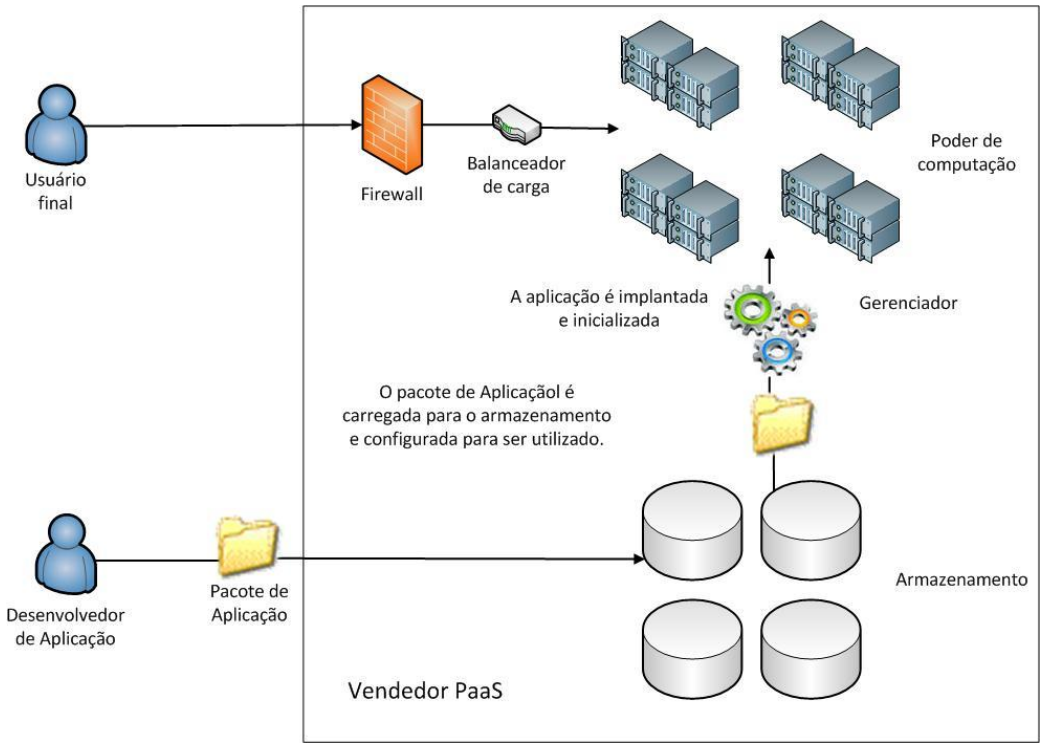

Figura 2.5: Modelo de Plataforma como Serviço (PaaS).

2.1.3.3 Software como Serviço (SaaS)

O modelo SaaS é um modelo de distribuição de *software* onde o aplicativo é hospedado por um fornecedor ou provedor de serviço e disponibilizado ao cliente através da rede, geralmente a internet [28]. Seguindo as características inerentes da computação em nuvem sua tarifação normalmente é medida pelo uso, também conhecida como *pay-as-you-go* [26]. A maior disponibilidade e melhorias nos serviços de banda larga contribuem para uma maior aceitação e uso desse tipo de serviço ao redor do mundo [28].

Para o uso o cliente apenas precisará acessar o serviço na nuvem através de um *browser* ou de um aplicativo específico, com máquinas com poucos recursos, já que a nuvem provera toda infraestrutura necessária para a execução da

aplicação. Com o serviço disponível na internet o cliente poderá ter acessa-lo de qualquer lugar, sem a necessidade de instalação ou configuração de software, permitindo a mobilidade, otimizando assim seu uso.

Inúmeros provedores já disponibilizam o *software* como serviço, dentre eles a pioneira foi a Salesforce.com e seus produtos na área de Gestão de Relacionamento com o Cliente. A Google com o Google Apps, que consiste num conjunto de aplicações como o correio eletrônico (Gmail), o programa de mensagens instantâneas (Gtalk) e o editor de documentos (Gdocs), é a mais popular e outros como Panda *Cloud* Antivirus, Photoshop *online* e Microsoft *Office Online* também são exemplos deste modelo de serviço. Na Figura 2.6 mostra um modelo de SaaS.

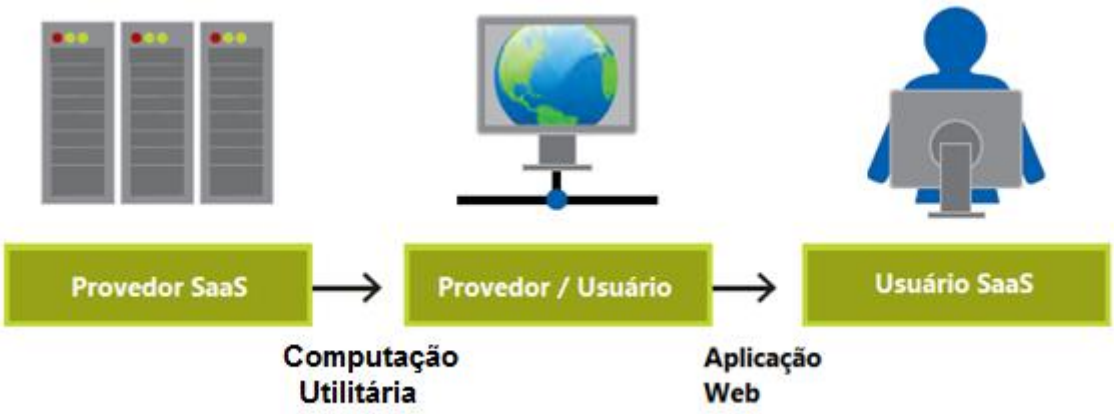

Figura 2.6: Software como serviço (SaaS) [29].

2.1.4 Modelos de Implantação da computação em nuvem

De acordo com modo de acesso e a disponibilidade dos serviços que o provedor de computação em nuvem oferece ao usuário, existem diferentes modelos de implantação. Podendo de acordo com seu processo de negócio, tipo de informação e do nível de visão, restringir ou liberar o acesso. Os modelos de implantação da computação em nuvem podem ser divididos em nuvem:

- Nuvens públicas são disponibilizadas na forma *pay-as-you-go*(pagando pelo que usa) ou gratuita para o público em geral, o serviço que está sendo vendido é *utility computing* [29]. No modelo de implantação publico

as restrições de acesso quanto ao gerenciamento de redes e o uso de técnicas para autenticação e autorização não podem ser aplicadas [1];

- Nuvens privadas é o termo para se referir aos centros de dados internos de uma empresa ou outra organização que são não disponibilizadas ao público em geral, sendo aplicadas sobre elas técnicas de autenticação e autorização, a fim de restringir o acesso [29];
- Nuvem comunitária tem como característica o compartilhamento da infraestrutura por várias organizações, normalmente essas organizações compartilham preocupações como missão, requisitos de segurança, política e considerações de conformidade [3];
- Nuvem híbrida é composta por dois ou mais modelos de implantação, onde cada nuvem é de uma entidade distinta, porém estão unidas pelo uso de estruturas de padronização ou proprietárias que garantem a portabilidade dos dados e aplicações [3]. Assim teremos nuvem publicas e privadas e/ou comunitária.

A Figura 2.7 mostra um modelo de Nuvem Pública, Nuvem Privada e Nuvem Comunitária.

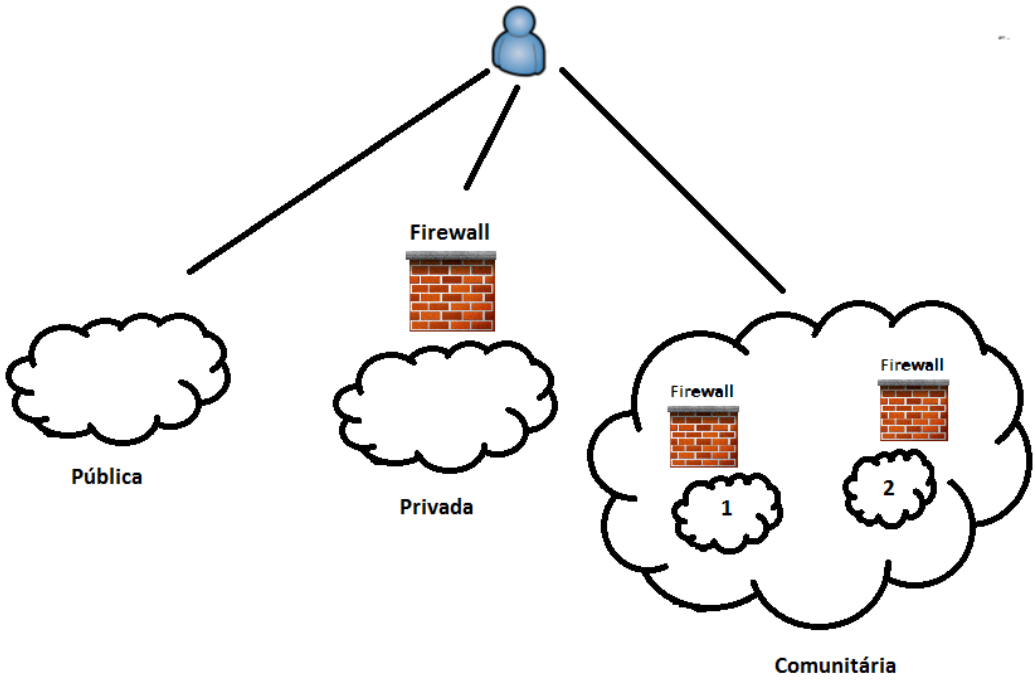

Figura 2.7: Nuvem Pública, Nuvem Privada e Nuvem Comunitária.

2.1.5 Papéis na Computação em nuvem

A definição dos papéis na computação em nuvem vem da necessidade de definir responsabilidades, políticas de acesso e perfil para os diferentes tipos de usuários. Os usuários podem ser classificados de acordo com os papais desempenhados.

O provedor que é responsável por disponibilizar, gerenciar e monitora toda a estrutura da nuvem. O desenvolvedor tem os recursos fornecidos e disponibiliza os serviços para os usuários finais. Por fim o usuário final consome os serviços disponibilizados pelo desenvolvedor. A Figura 2.8 mostra os diferentes tipos de papeis desempenhados pelos usuários da computação em nuvem.

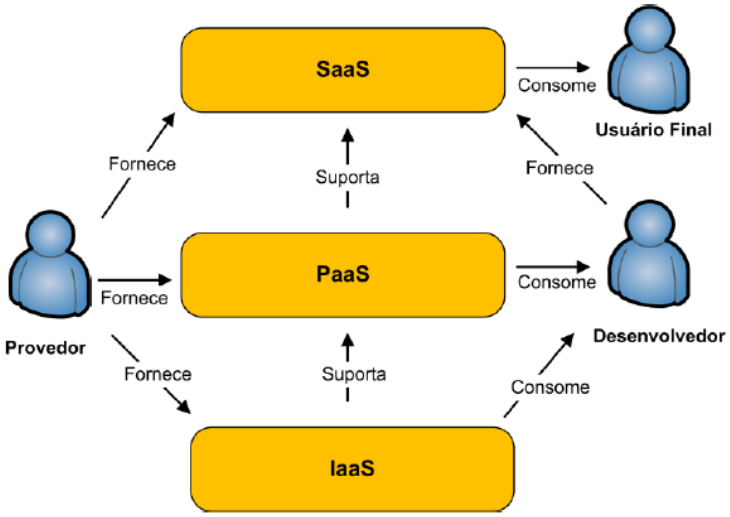

Figura 2.8: Papéis na computação em nuvem.

2.1.6 Desafios da computação em nuvem

Os desafios da computação em nuvem são entraves que devem ser superados para que esse novo paradigma se expanda ainda mais. Os principais desafios estão relacionados à segurança e padronização de estrutura, porém outros como escalabilidade, confiabilidade e disponibilidade também devem ser superados para que a computação em nuvem tenha uma maior aceitação por parte dos usuários e organizações mundo a fora.

A segurança é o maior ponto de desconfiança por parte dos não adeptos a computação em nuvem. As informações que antes tinham seu armazenamento em um disco local agora passaram a ser armazenadas em local não específico onde não se pode precisar nem o local e nem os tipos de dados armazenados juntos.

Preocupação com a privacidade e integridade desse dado passou a ser item muito importante, uma vez que o acesso à nuvem é feito pela internet expondo-a a ataques [3]. Outra preocupação com o armazenamento dos dados é que um provedor de computação em nuvem pode alocar seus centros de dados em países diferentes, com legislações e regulamentos diferentes deixando uma duvida quanto à proteção desses dados [30]. A Figura 2.9 mostra os principais elementos que devem ser protegidos na nuvem.

Figura 2.9: Elementos principais para proteger na nuvem.

A padronização é um ponto importante a ser observado antes que dados confidenciais e regulamentados sejam migrados para o ambiente de nuvem. Padrões de segurança e compatibilidade que abrangem autenticação sólida, autorização delegada, gerenciamento de chaves para dados criptografados, proteção contra perda de dados e emissão de relatórios normativos [31]. Seguindo essas especificações como modelo os provedores teriam uma maior segurança e chegariam a uma federação dos controles de autorização, porém não há uma agencia reguladora e chegar a esses padrões ainda pode levar vários anos. Tal fator impede uma interoperabilidade entre nuvens não permitindo que usuários executem seus programas e seus dados em nuvens diferentes.

A escalabilidade vem da necessidade das aplicações para a nuvem serem elásticas, ou seja, escalável. Com essa característica os recursos podem ser

alterados conforme sua utilização exigindo assim flexibilidade das aplicações e de seus dados [3].

Outro desafio da computação em nuvem é a confiabilidade, ou seja, à frequência com que o sistema falha e se elas geram ou não perda de dados. Por isso as aplicações desenvolvidas para computação em nuvem devem possuir uma arquitetura que garanta a integridade dos dados mesmo que haja falhas dos servidores ou instancias (máquinas virtuais). Este problema pode ser amenizado com o uso de políticas de gerenciamento de backups [3].

Por ultimo a disponibilidade, uma vez que a aplicação em nuvem pode ser um gargalo de conexão, gerando assim grande preocupação. Outra preocupação a respeito da disponibilidade é a necessidade de conexão com a internet, uma alternativa para isso seria ter mais de um provedor de nuvem, porém isso acarretaria em ter mais de uma nuvem, o que pode aumentar muito os custos além de exigir interoperabilidade entre elas [3].

2.1.7 Segurança na computação em nuvem

A segurança na computação em nuvem é um aspecto importante porque extrapola a segurança da infraestrutura tradicional migrando essa tarefa para a nuvem e tirando do controle da organização. Deve ser levado em consideração o modelo de implantação escolhido, já que dependendo do modelo, inúmeros usuários (corporações, redes, provedores de serviços etc.) podem estar envolvidos na mesma infraestrutura e isso pode acarretar problemas em gerenciar dados corporativos confidenciais.

A segurança computacional passou por várias fases no decorrer dos anos, inicialmente visava prevenir violações de proteção, depois o objetivo era detectar e eliminar as violações que não podiam ser prevenidas. Em seguida o foco passou a ser a tolerância a ataques para manter o serviço sempre disponível. Agora a segurança é um serviço comercializado onde o usuário deve confiar em quem o vende [26].

Mesmo com a computação em nuvem, se inserindo na proposta de um principio organizador de grande poder computacional como serviço, ela ainda carece de controles efetivos de segurança para conviver nesse cenário de virtualização de rede e compartilhamento de recursos de forma autonômica [32]. A segurança na

nuvem é um subdomínio emergente da segurança da informação. Refere-se a um amplo conjunto de políticas, tecnologias e controles implantados para proteger os dados, aplicativos e infraestrutura associada à Computação em Nuvem [9].

Na norma ISO 7498-2, produzido pela Organização Internacional de Normalização (ISO), Segurança da Informação deverá abranger um número de temas sugeridos [4]. Segurança em Computação em Nuvem deve também seguir estas recomendações, a fim de tornar-se uma solução de tecnologia eficaz e segura, estes itens são:

- Identificação e autenticação: dependendo do tipo de nuvem adotada e do modelo de troca de dados, os usuários em primeiro lugar devem ser especificados e estabelecidos às prioridades de acesso e permissões. Isto é feito com o proposito de verificar e validar os usuários da nuvem através de nome de usuário e senhas [4].
- Autorização: exigência importante de segurança da informação em computação em nuvem para garantir a integridade referencial seja mantida. Decorre sobre exercer controle e privilégios sobre os fluxos de processo dentro de computação em nuvem [4].
- Confidencialidade: é uma obrigação quando se emprega uma nuvem pública, devido à natureza das nuvens públicas acessíveis por varias organizações e usuários diferentes. Afirmando confidencialidade nos perfis dos usuários e protegendo seus dados, isso deve impedir que os dados sejam acessados por outros usuários. São usados protocolos de segurança da informação em várias camadas diferentes de aplicações na nuvem [4].
- Integridade: é uma implicação principalmente do acesso aos dados na computação em nuvem. Portanto, as propriedades ACID (atomicidade, consistência, isolamento e durabilidade) de dados da nuvem devem sem dúvida ser robustamente impostas em todos os modelos de computação em nuvem [4].
- Não-repúdio: em computação em nuvem pode ser obtido através da aplicação dos tradicionais protocolos de segurança de e-commerce e provisionamento *token* para transmissão de dados dentro de aplicações em nuvem, tais como assinaturas digitais, *timestamps* e serviços de

confirmação de recebimentos (de recepção digital de mensagens de dados confirmando enviadas / recebidas)[4].

- Disponibilidade: é um dos requisitos de segurança da informação mais crítico na computação em nuvem, porque é um fator de decisão importante ao decidir entre os fornecedores de nuvem privados, pública ou híbrida [4].

2.1.7.1 Segurança da infra-estrutura

Com o uso da infraestrutura como um serviço, o aprovisionamento de diversos servidores virtuais em uma mesma plataforma IaaS, cria um risco de que servidores virtuais menos seguros possam ser criados isso acarretaria em uma fonte de ataque dentro da própria infraestrutura. Como forma de garantir a segurança padrões das industrias da computação em nuvem para efeito de infraestrutura devem ser seguidos.

Segundo [33], para proteger os servidores virtuais de uma nuvem é necessária a adoção de alguns procedimentos de segurança operacionais bem fortes, aliados à automação de procedimentos. Para isto devi-se seguir as seguintes recomendações:

- Usar configurações seguras para as imagens (sistema operacional instalado em uma instância de máquinas virtual) padrão. Construindo imagens de VM personalizados, que têm apenas os recursos e serviços necessários para apoiar a pilha de aplicativos. Limitando as capacidades da pilha de aplicativos subjacente não só limita a superfície de ataque do hospedeiro, mas também reduz o número de *patches* necessários para manter essa pilha de aplicativos seguros;
- Controlar as imagens de VM disponíveis no provedor observando as versões e o sistema operacional. Sendo que a melhor alternativa é a utilização de suas próprias imagens de acordo com os padrões de segurança estipulados e testados por você;
- Proteger a integridade da imagem com o uso de regras de acesso bem rígidas;
- Salvaguardar as chaves privadas necessárias para acessar hosts na nuvem;

- Isolar as chaves de decodificação da nuvem, onde os dados são hospedados, a menos que seja necessário para descodificação e apenas no período de atividade de decodificação real, salvo quando for necessária para decodificação de dados de uma aplicação que requer processamento contínuo;
- Não incluir credenciais de autenticação em suas imagens virtuais com exceção de uma chave para decifrar a chave do sistema de arquivos;
- Não permitir a autenticação baseada em senha para acesso *shell;*
- Exigir senhas de super-usuário ou acesso baseado em função;
- Executar um firewall de host e abrir apenas as portas mínimas necessárias para suportar os serviços em uma instância;
- Executar apenas os serviços necessários e desativar os que não são utilizados (por exemplo, desligar FTP, serviços de impressão, serviços de arquivos de rede e serviços de banco de dados se eles não são obrigatórios);
- Usar estratégias para detecção de acesso não autorizado as *Vlan;*
- Habilitar a auditoria do sistema e registro de eventos;
- Instâncias comprometidas ou com um comportamento suspeito devem ser encerradas;
- Instituir método para recuperar imagens danificadas;
- Revisão periódica nos *logs* de atividades suspeitas.

Porém nem todos esses cuidados podem ser encontrados em uma nuvem pública, ficando assim para o cliente sanar todas as lacunas deixadas pelo provedor. Talvez no futuro, com a padronização dos serviços disponibilizados pelos provedores de computação em nuvem, tais especificações sejam sanadas e oferecidas juntamente como serviços sob demanda.

2.2 Computação autonômica

Com o aumento constante da complexidade e da exigência de rapidez nas respostas de gerenciamento dos sistemas atuais, surgiu à necessidade que tal gerenciamento não ficasse atrelado apenas as ações humanas. Os diferentes tipos

de dados, aplicações, ambientes de programação e infraestrutura de informação estão rapidamente se tornando frágil, incontrolável e inseguro devido a forma tradicional de gerenciamento estático. Levando os pesquisadores a considerar um paradigma alternativo de programação e técnicas de gestão que são baseados em estratégias utilizadas pelos sistemas biológicos para lidar com a complexidade, dinamismo, heterogeneidade e incerteza [11].

Esse paradigma é inspirado no sistema nevoso autônomo humano que lida com relações de complexidade e incertezas. Os sistemas de computação e aplicações autonômicas são capazes de gerir-se com mínima intervenção humana. A partir da obsarvação do sistema humano chegou-se os passos que são a base para o comportamento autonômico. Então todo sistema autonômico tem os seguites passos: sentir, analizar, conhecer e executar [11].

As propriedades de um sistema de computação autonômica são definidas em oito principios que são [11] [12]:

- Auto-Conhecimento (*Self-Awareness*): um sistema autonômico conhece a si mesmo e está ciente de seu estado e seus comportamentos [11];
- Auto-Proteção (*Self-Protecting*): um sistema autonômico deve ser capaz de detectar e proteger-se contra ataques aos seus recursos internos e externos, além da manutenção da segurança global do sistema e integridade [12];
- Auto-Otimização (*Self-Optimizing*): um sistema autonômico deve ser capaz de detectar a degradação do desempenho do sistema e imediatamente executar funções de auto-optimização para melhorar seu desepenho [11];
- Auto-Cura (*Self-Healing*): um sistema autonômico deve estar cientes dos potenciais problemas e deve ter a capacidade de se reconfigurar para continuar a funcionar sem problemas [11];
- Auto-Configuração (*Self-Configuring*): um sistema autonômico deve ter a capacidade de ajustar dinamicamente os seus recursos com base em seu estado e lenvando em concideração o estado de seu ambiente de execução [11];
- Ciencia de Contexto (*Contextually Aware*): um sistema autonômico deve estar ciente de seu ambiente de execução e ser capaz de reagir a mudanças nele [11];

- Aberto (Open): um sistema autonômico deve ser portável entre mulpitos hardware e arquiteturas de software e, conseqüentemente, deve ser construída sobre protocolos e interfaces padrão e aberto [11];
- Antecipatório: um sistema autonômico deve ser capaz de antecipar, na medida em que for possível, as suas necessidades e comportamentos do seu contexto, e ser capaz de gerir a si mesmo de forma proativa [11];

A Figura 2.10 ilustra as propriedades da computação autonômica.

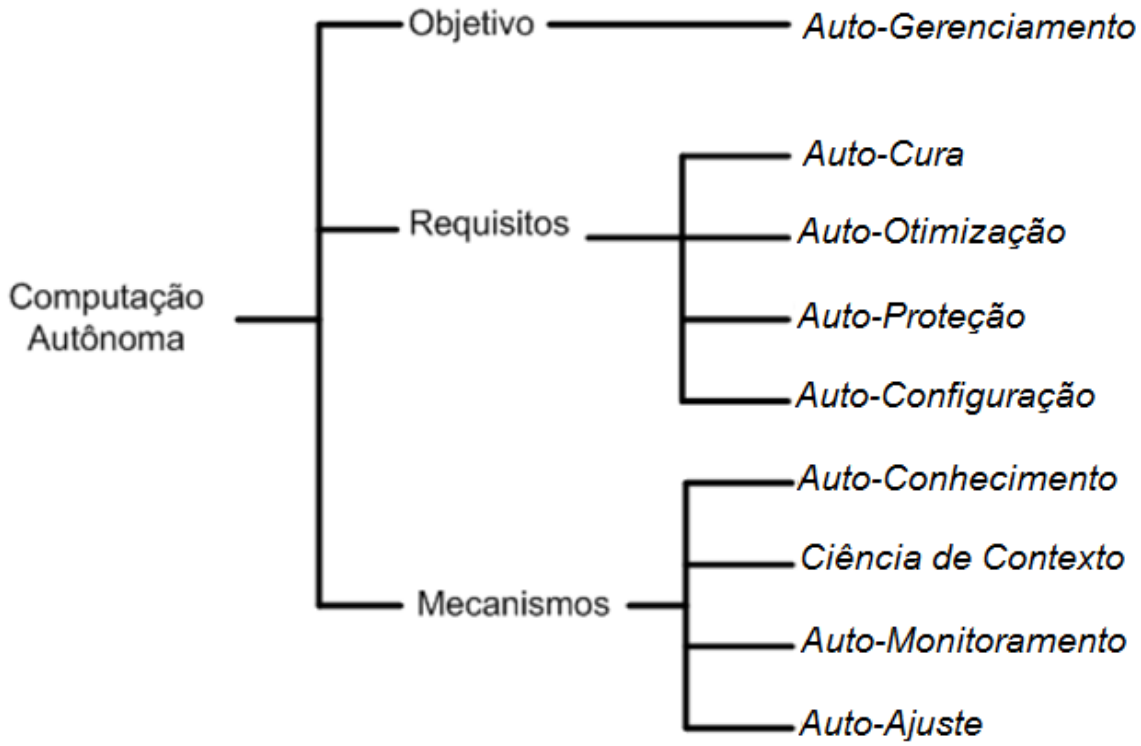

Figura 2.10: Propriedades Gerais para Computação Autonômica [13].

Em [13] são mostradas as três principais áreas onde tem maior concentração de trabalhos relacionados com a computação autonômica (C.A.) e são elas: gerenciamento de energia em grandes centros de computação, gerenciamento de ambientes em grades computacionais e computação ubíqua. Porém nos últimos anos devido ao grande crescimento da computação em nuvem, ela também passou a ser uma área onde a C.A. tem grande utilização como mostra [14], [15], [16] e [17].

As características da C.A. normalmente são postas de forma a integrar mais de uma propriedade, ou seja, uma aplicação tem como propriedades autonômicas a autoconfiguração, auto-otimização, autoproteção e auto cura. Então a

aplicação e a infra estrutura fornecem informações que permitem assimilação do contexto e auto conhecimento, depois é disparado um evento gatilho para inicia a analise, dedução e planejamento usado o auto conhecimento do sistema, em seguida são traçados os planos de execução que podem ser de auto cura, auto configuração, auto proteção, etc [11]. Estas etapas são feitas a fim de alcançar uma capacidade adaptativa do sistema, inseridos *loops* de controle, coleta de informação e tomada de decisões alcançando a adaptação conforme o necessário.

2.2.1 Requisito para construção de sistemas autônomos

Nesta sessão serão abordados os requisitos básicos para a construção de sistemas autônomos. Para isto são seguidas três perspectivas: arquitetura de *software* para construção de sistemas autônomos, modelos de arquiteturas para representação do conhecimento e infraestrutura de apoio.

Arquitetura de software para construção de sistemas autônomos

Devido a grande complexidade da construção de sistemas autonômicos algumas arquiteturas foram propostas, tais arquiteturas que na verdade são formas de automatizar o ciclo de gerenciamento do sistema. Nos ciclos de gerenciamento propostos algumas atividades podem ser destacadas, tais como:

- Monitoramento: tem por função a coleta, agregação, correlação e filtragem dos dados sobre os recursos gerenciados [13].
- Análise e decisão: nesta etapa são examinados os dados e são feitos as inferências a respeito deles, caso necessário são definidas novas estratégias e políticas para que o sistema volte ao equilíbrio [12].
- Controle e execução: as novas estratégias e políticas definidas na fase anterior são aplicadas com o objetivo de restaurar o equilíbrio do sistema [13].

A figura 2.11 mostra o modelo de ciclo de gerenciamento autonômico MAPE-K (Monitor, Analyze, Plan, Execute , Knowledge) desenvolvido pela IBM no qual mostra as atividades do ciclo de gerenciamento baseado no modelo MAPE-K.

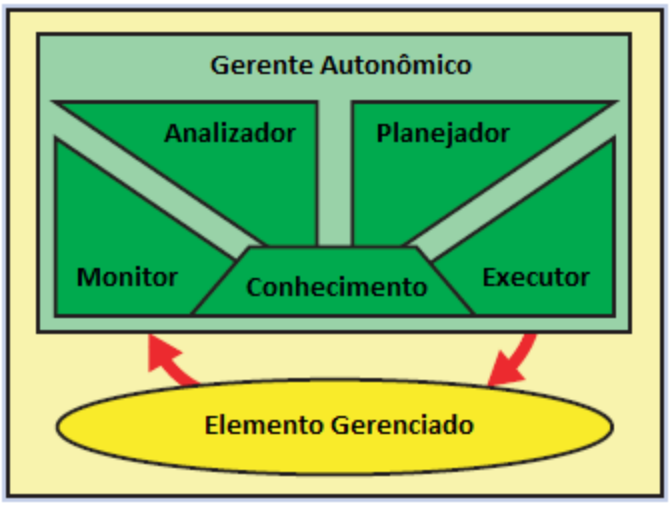

Figura 2.11: Ciclo de gerenciamento autonômico MAPE-K [12].

O ciclos de gerenciamento podem ser implementados seguindo dois tipos de arquiteturas: as baseadas em elementos autônomos e as baseadas em infraestrutura.

As arquiteturas baseadas em elementos autônomos para sistemas autogerenciáveis são formados pela colaboração de elementos autônomos. O elemento autônomo consiste em dois módulos, uma unidade onde acontece a execução dos serviços providos e uma unidade de controle que monitora seu estado de contexto, analisando a condição corrente e promovendo a adaptação necessária [13], e outras partes como o elemento gerenciado, o ambiente e o controle.

O elemento gerenciado é a unidade funcional, a qual é monitorada, pois pode ser afetada em virtude de falhas, escassez de recursos, ataques, problemas de desempenho, etc. O ambiente são fatores que influenciam o elemento gerenciado, podendo ser interno quando se tratar de mudanças no próprio elemento gerenciado ou externo quando tratar de fatores que fazem parte do ambiente de execução. O controle é responsável por gerenciar o elemento, ele recebe definições dos requisitos, inspeciona e caracteriza o estado do elemento, inspeciona o estado geral do sistema, determina o estado do ambiente e com posse desses dados ele controla as operações do elemento gerenciado para adaptar seu comportamento através dos sensores e atuadores [34].

O modelo MAPE (Monitora, Analisa, Planeja e Executa) é divide em dois dentro de um elemento autonômico como mostra a Figura 2.12, sendo um local para

tratar de estados conhecidos e um global que trata de estados não conhecidos, os quais para gerar uma solução ele terá que empregar técnicas de aprendizado de máquina, inteligência artificial ou mesmo intervenção humana para que obtenha uma resposta com uma instrução para que atue junto ao elemento gerenciado para que o mesmo volte ao estado de equilíbrio.

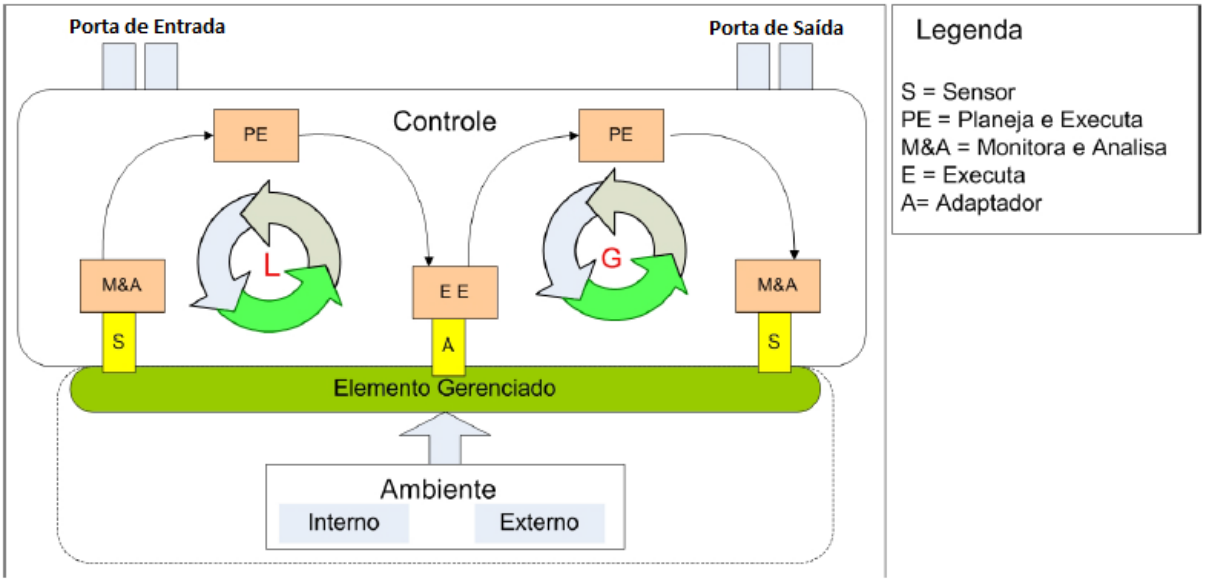

Figura 2.12: Estrutura de um elemento autonômico [13].

Arquitetura baseada em infraestrutura

O outro modelo de arquitetura para construção de sistemas autonômicos é o baseado em infraestrutura, no qual os elementos do sistema não são autonômicos. Eles têm suas propriedades autonômicas providas pela infraestrutura através de modelos que descrevem e analisam os componentes do sistema. Os modelos arquiteturais tem sua representação em grafos que descrevem os componentes do sistema e o seus relacionamentos [13]. A Figura 2.13 mostra a arquitetura de abordagens baseadas em infraestrutura.

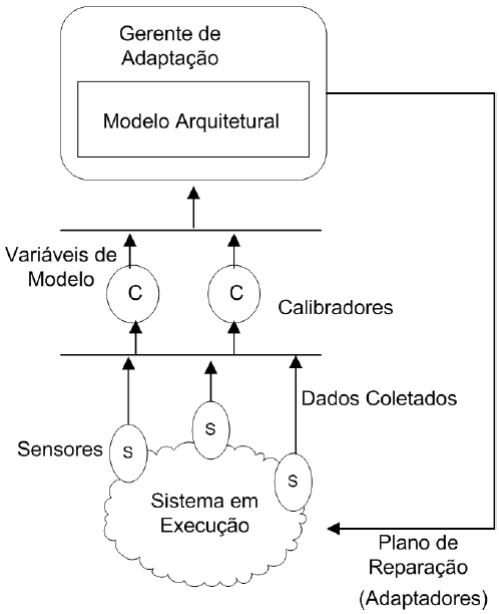

Figura 2.12: Arquitetura genérica de uma abordagens baseadas em infraestrutura [13].

2.3 Metodologia de decepção

A metodologia de decepção é uma metodologia que visa atrair possíveis ataques destinados a um sistema real para um sistema que simula os serviços, desviando-os, assim, do sistema real e possibilitando a geração de *logs* que podem ser estudados, podendo gerar novas ou estratégias de defesas mais eficientes. Um representante do uso dessa metodologia são os *honeypots*.

Os *honeypots* (tradução para pote de mel) são sistemas de segurança defensivos como os IDS (Sistemas de detecção de intrusão) e *Firewalls* e ele não defende a rede ativamente, mas simula serviços disponíveis na rede para atrair ataques de intrusos [5]. É um sistema capaz de simular serviços como FTP (Protocolo de Transporte de Arquivo), SQL (Linguagem Estruturada de Consulta), Web, etc. Em suas interações com os intrusos ou atacantes são gerados *logs* que podem ser utilizados para gerar ACLs (Listas de Controle de Acesso) ou mesmo assinaturas desses ataques.

Os *honeypots* podem ser fisicamente instalados ou virtualizados [35]. São classificados quanto ao grau de interação com o atacante podendo ser: de baixa, média e alta interatividade [5][35]. O *honeypot* de baixa interatividade proporciona uma comunicação limitada entre o *honeypot* e o atacante porém são muito usados para pesquisa. Eles oferecem a possibilidade de emular serviços de rede em portas

pré-configuradas, como FTP, SQL, Web, SSH, etc, no entanto, esses serviços são restritos a apenas responder às perguntas básicas (como o comando *ping* ou uma tentativa de conexão) que deve ser.

Já o de média interatividade recolhe mais informações sobre os ataques do que uma baixa interação, dando a possibilidade do atacante interagir um pouco mais com o *honeypot*. Essa interação média leva o atacante a interagir um pouco mais com *honeypot* que será capaz de responder a comandos específicos, usando mensagens pré-configuradas [35] [37]. O de alta interatividade simulando um serviço por completo, coletando o máximo de informação do ataque, porém deve-se ter muito cuidado na utilização de um *honeypot* de alta interatividade, uma vez que, ele é a copia de sistema real e pode ser um utilizado para gerar ataques, se dominado pelo atacante, ou mesmo podem receber requisições verdadeiras enviando informações que podem esta desatualizada.

A tabela 1 mostra as vantagens e desvantagens entre um honeypot de alta-interatividade e de baixa-interatividade.

Alta-Interatividade	**Baixa-Interatividade**
Serviços, Sistema Operacional e Aplicações reais	Serviços emulados de pilha TCP/IP, Vulnerabilidades, e assim por diante
Mais alto risco	Baixo risco
Difícil de implantar e manter	Fácil de implementar e manter
Capturar extensa quantidade de informações	Capturar informações apenas sobre os ataques

Tabela 1: Vantagens e desvantagens entre um honeypot de alta-interatividade e de baixa-interatividade [37].

Outra possível distinção na área *de honeypots* é entre *honeypots* físicos e virtuais. *Honeypot* físico significa que o *honeypot* está sendo executado em uma máquina física, isso implica muitas vezes alta-interação, permitindo, assim, que o sistema seja completamente comprometido. Tem um valor elevado para instalação e manutenção. Em uma grande faixa de endereço, é impraticável ou impossível de implantar um *honeypot* físico para cada grupo de endereços IP. Nesse caso, é preciso implantar *honeypot* Virtual [37].

Temos como as principais funcionalidades de um *honeypot* : coletar informações sobre os ataques, atrair o atacante tirando-o da sistema real, interagir com o atacante afim de coletar a maior quantidade possível de informações sobre o ataque [36] .

Algumas vantagens foram atribuídas ao uso dos *honeypot* por [38]:

- Pequeno volume de dados a ser analisado, uma vez que só gera dados dos atacantes que integem com ele;
- Ausência de falsos positivos;
- Exigência de mínima de *hardware* nos casos de honeypot virtuais;
- Captura de tráfego criptografado;
- Descoberta de novas tecnologias utilizadas para ataques;

Algumas desvantagens no uso de *honeypot* [10]:

- Gera uma visão limitada de tráfego, porque o atacante não é obrigado a interagir com ele;
- Pode servir de “trampolim” para sua rede caso seja comprometido, prejudicando assim a integridade das demais máquinas da rede;
- Caso ele não fique bem posicionado na rede, pode não sofrer nenhuma interação e não gerar *logs*, gerando gastos desnecessários.

2.4 Trabalhos relacionados

The Generation of Attack Signatures Based on Virtual Honeypots

No artigo de Xinyu Tang [6] ele apresenta a implementação de um sistema que gera de forma autônoma assinaturas de ataques com base em *honeypot* virtuais. Ele utiliza técnicas de virtualização para elaboração de um *honeypot* com algumas venerabilidades atraindo assim o atacante. Outra estratégias associadas como a implementação de *firewall* e listas de controle de acesso ajudam a manter a segura e da uma resposta a esses ataques.

Utiliza o Honey-d [37], que é um *honeypot open-source* poderoso criado pelo engenheiro da Google Neils Provos, que cria sistemas operacionais virtuais de com acesso a rede e tem como atributos:

- A capacidade de monitorar milhões de endereços IP e trabalhar ativamente como um endereço IP (criando *Vlans*) e todas as funções funcionam simultaneamente;
- Tem um mecanismo de um mecanismo de impressão digital correspondente para as máquinas virtuais e ferramenta digital que pode enganar atacante;
- Honeyd interagir com *hackers*, carregando os scripts correspondentes *plug-in* e esses *scripts* podem imitar serviços correspondentes para aprofundar a interação com os atacantes;
- É executado como um processo em segundo plano, e o *honeypot* gerado por ele é simulado pelo processo de fundo;
- Produzir muitos *honeypots* virtuais na rede, e esses *honeypots* podem ser colocados dentro de uma mesma rede virtual assim restringindo o acesso dele a rede principal.

Juntamente com o Honeyd alguns *plug-ins* para a geração de assinaturas podem ser instalados, eles fazem a função de enviar informações sobre o ataque através de filtros de *logs*.

O Sistema tenta se conectar a solicitação do link suspeito no endereço IP específico no *front-end* usando um *honeypot*, e pode armazenar uma quantidade limitada de estado da sessão TCP ou UDP. A sessão com mais dados transferidos é a mais valiosa para a extração de características. Assim o Sistema de geração de assinatura inclui as seguintes partes: captura de dados do ataque, processamento de dados e geração de assinatura.

As regras geradas pelo *Snort* são baseadas em arquivos de texto. Os arquivos são diferentes para grupo, como por exemplo, uma regra para conexões FTP vai ter seu nome “ftp.rules”. Ao iniciar o *Snort* ele carrega os arquivos que contém as configuras para geração de assinaturas caso haja algum. A figura 2.13 mostra modelo da arquitetura do sistema.

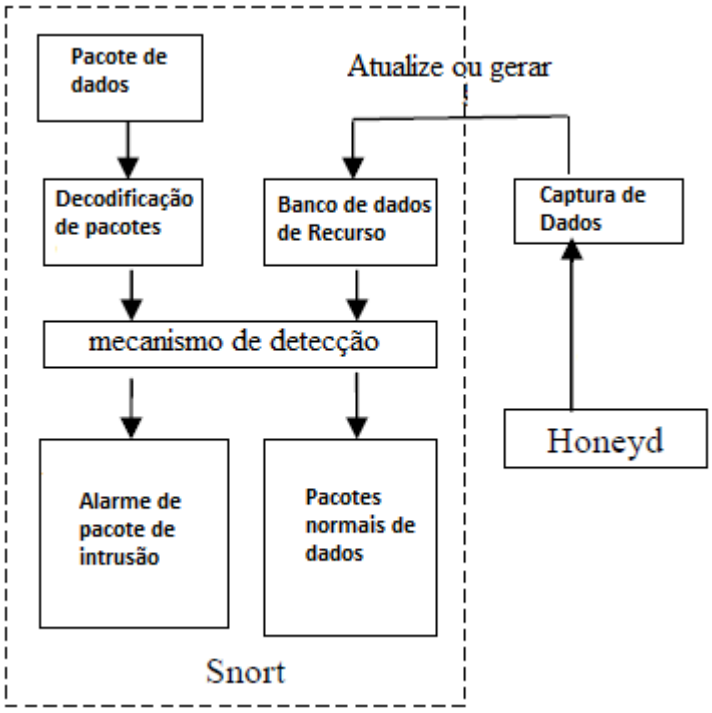

Figura 2.13: Modelo de aplicação do sistema [6].

O *framework* integra um sistema de *honeypot* virtuais, com um sistema de geração de assinaturas (SGS) para coletar e gerar assinaturas de ataques enviados pelo *host* e comparando as regras *snort*. A Figura 2.14 mostra o diagrama do *framework*.

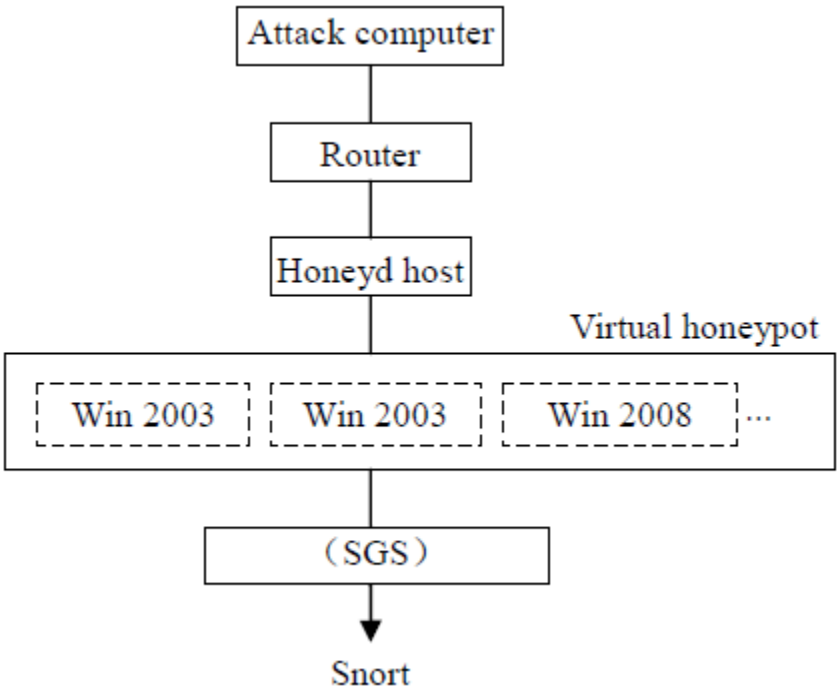

Figura 2.14: Sistema do *Framework* [6].

A Honeynet-Based Firewall Scheme With Initiative Security Strategies

Em [7] ele propõe uma estratégia de defesa em conjunto, onde é agregado ao *firewall* um *honeypot*. Esse *honeypot* coletará informações sobre ataques não conhecidos ainda e as passar para o *firewall* que por sua bloqueara os pacotes de origem maliciosos. Tal solução daria mais dinamismo às atualizações das bibliotecas utilizadas por *firewall* e por IDS (Sistema de detecção de intrusão) [40], [41], [42].

A arquitetura de defesa em conjunto proposta em [7] é composta de "*Firewall* A", *Honeynet*, módulo de apredisagem de regas (*Rule-learning* module), módulo analizador de dados (*Data-analyzing module*) e "*Firewall* B". Como mostra a Figura 2.14.

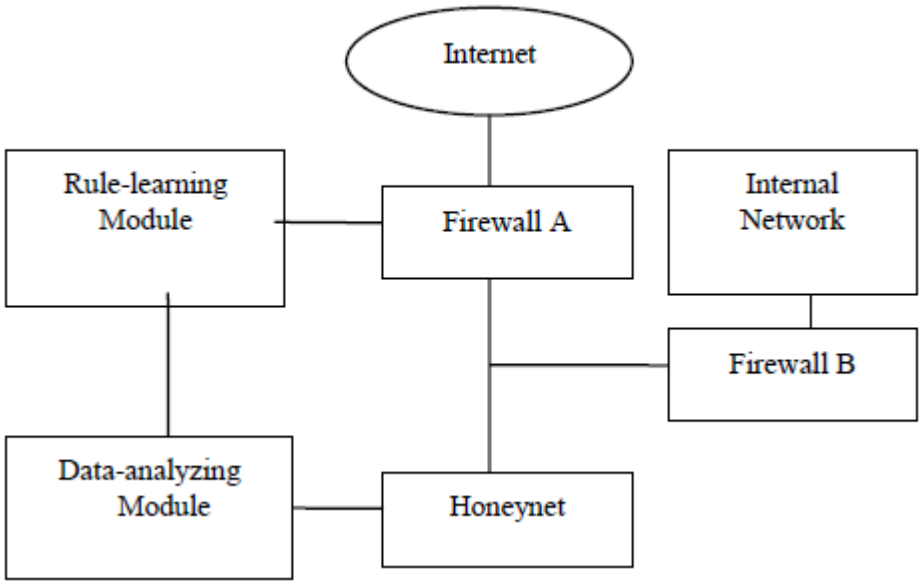

Figura 2.15: Arquitetura do *firewall* baseado em *honeypot* [7]

O "*Firewall* A" é implantado na região mais periférica. Sua principal função é proteger a rede interna. Ele monitora a entra de dados, do primeiro nível de dados capturados por *logs*, e controla os dados que saem. A *Honeynet* e rede interna são implantados atrás do "*Firewall* A" e o *honeypot* é isolado da rede interna por "*Firewall* B". O *honeypot* atrai os ataques na rede interna e captura todos os dados que entram e saem. A *Honeynet* registra todos os comportamentos e as operações dos invasores, e os envia para o módulo de análise de dados (*Data-analyzing module*), com os *logs* de *firewall*. Por mineração de dados, o módulo de análise de dados (*Data-analyzing module*) extrai as informações relevantes a partir dos dados originais e os transmite para o módulo de aprendizagem de regras (*Rule-learning module*). Por fim, o módulo de aprendizagem de regras gera regras de defesa relevantes para o *Firewall* A. A Figura 2.16 mostra como se dá a criação de novas regras.

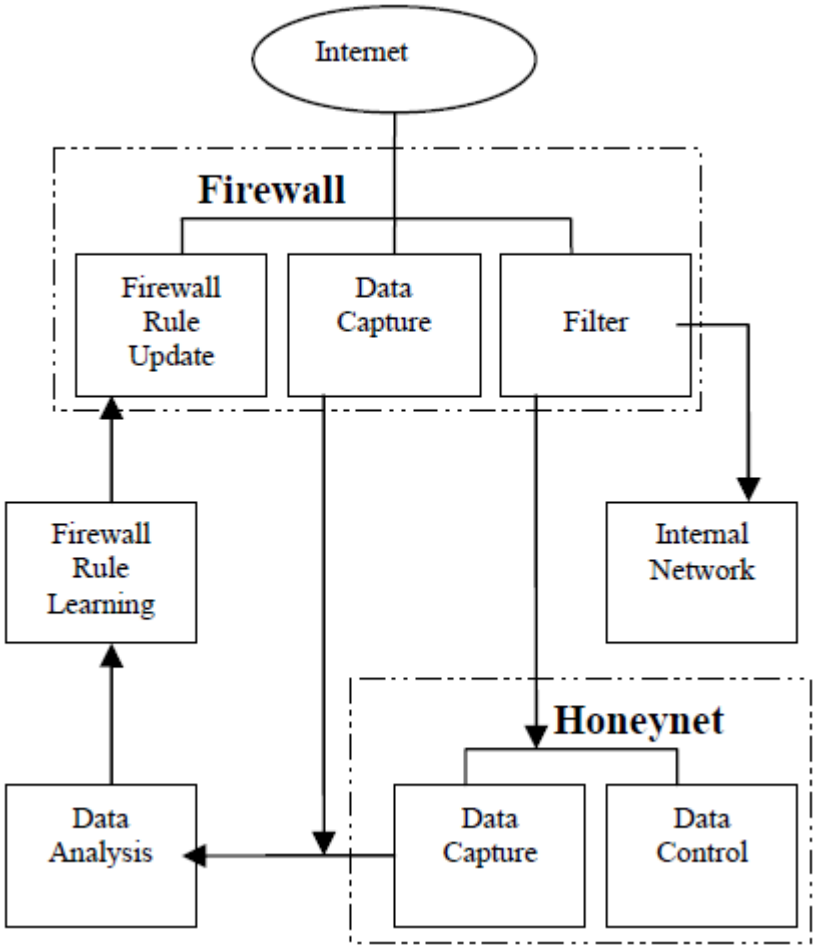

Figura 2.16: Princípio do *firewall* baseado em *honeypot* [7].

O módulo de análise de dados adota a tecnologia de mineração de dados. Primeiramente, os *logs* do *firewall*, depois dados capturados por um *sniffer* [43] e os captados pelos *honeypots* são transmitidas ao módulo de análise de dados. Os dados são armazenados na base de dados de acordo com suas características de conexão com a rede e conteúdo. Os dados captados pelo *honeypot* são relativamente menores, porém com mais fidelidade.

O trabalho de [7] mostra vantagem em relação a um *firewall* tradicional, uma vez que, eliminou a passividade da segurança por *firewall,* inserindo um recurso de segurança com iniciativa, o *firewall* pode atualizar dinamicamente as suas regras para bloquear novos ataques, o que aumenta significativamente a segurança da rede interna.

A Study of Intrusion Signature Based on Honeypot

Em [8] Jun-Feng Tian et. al. apresenta um modelo de SISH (*Study of Intrusion Signature based on Honeypot*), usando *Honeypot* para coletar pacotes de intrusão de rede com ferramentas de captura no modo *multiplayer* e atividades em hospedeiro infectado. Ele analisou os pacotes, para reconstruir processo de ataque com o método de árvore de ataque e classificou as informações de acordo com o objeto de ataque. Depois criava assinatura com o LCS (*Longest Common Sub-string*) na mesma classe para fortalecer IDS e *firewall*.

Para garantir a passagem pela *honeynet* (seguimento da rede onde se encontra os *honeypots*), um roteador é colocado entre a Internet e a *honeynet*. Além disso, o roteador é utilizado como dispositivo de controle de dados secundário, aumentando a capacidade de controle de *firewall*, garantindo que o *honeypot* não seja usado para atacar outros *hosts*. A Figura 2.17 mostra a arquitetura do modelo SISH.

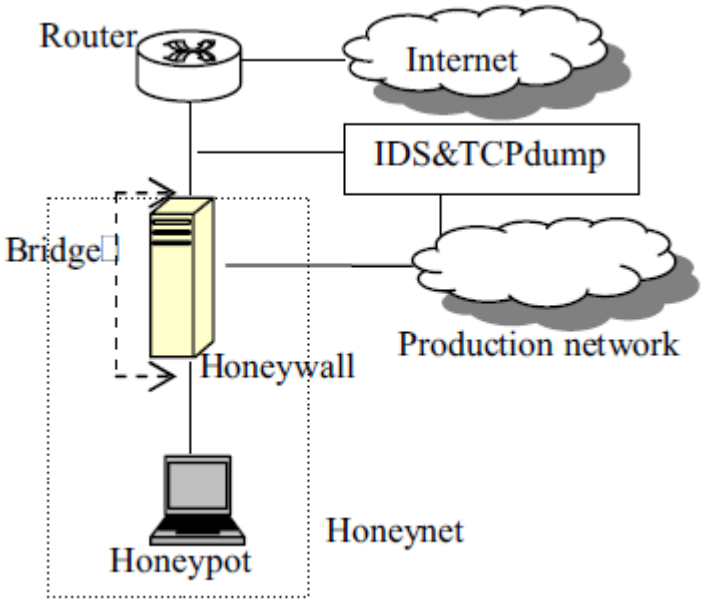

Figura 2.17: Arquitetura do modelo SISH [8].

A arquitetura tem logo a frente da *honeynet* toda a estrutura de *firewall* e ferramentas para bloquei de tráfego de rede não autorizado como IDS. Dificultando assim que o *honeypot* seja facilmente controlado e comece a atacar os outros *hosts* da rede. É utilizado o Honeywall [45] para mediar essa conexão, capturando e

analisando todo trafego que sai do *honeypot.* O Honeywall cria uma ponte transparente entre o router e o *honeypot* tornando-o invisível para o atacante.

Assim o Honeywall tem trêis conexões de rede sendo que uma é ligada ao roteador, a segunda no *honeypot* e a terceira na rede de produção. O modelo interage com os intrusos e intercepta dados dos ataques sofridos pelo *honeypot*. Adota um *honeypot* alta interação, para recolher informações muito mais valor que o de baixa interação [38]. A Figura 2.18

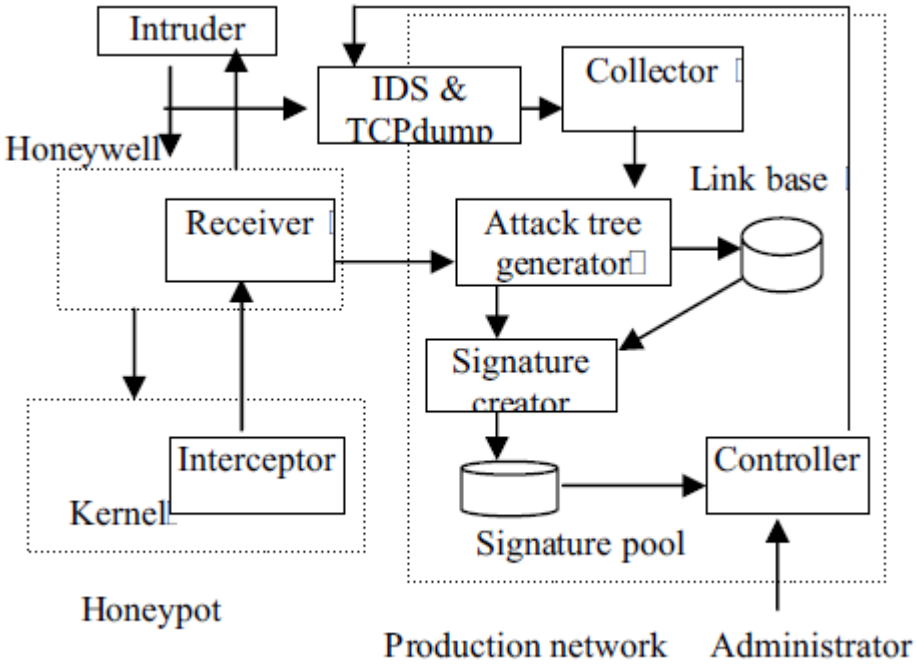

Figura 2.18: Estrutura Funcional do modelo SISH [8].

Automatic XSS detection and Snort signatures/ACLs generation by the means of a cloud-based honeypot system

Jacob [5] propõe em seu trabalho a detecção automática de ataques XSS e a geração de assinaturas do *Snort* para bloquear ataques similares no futuro. Para isso usa um *honeypot* de baixa interatividade instalado em uma nuvem computacional, que simula um servidor *web* que fica exposto a ataques XSS (*cross site scripting*).

Seu *honeypot* é configurando em uma em uma VM de uma nuvem computacional de modo que simule um servidor *web* lá hospedado. Para isso ele

propõe uma arquitetura que proporciona a máxima eficiência na filtragem do tráfego da rede. Divide essa arquitetura em duas sessões. A primeira sessão é composta de duas camadas de segurança que analisam o pacote IP e o conteúdo antes de entrar na infraestrutura de nuvem. A segunda sessão é composta de um sensor de segurança (*honeypot*) colocado no interior da infraestrutura da nuvem, que é capaz de detectar ameaças desconhecidas. No *front-end* da rede, um roteador, onde são carregadas as ACLs, é colocado e seguido por NIDS (*Network-based IDS*) *Snort*. A Figura 2.19 mostra um diagrama do fluxo dos caminhos escolhas para pacotes de entrada.

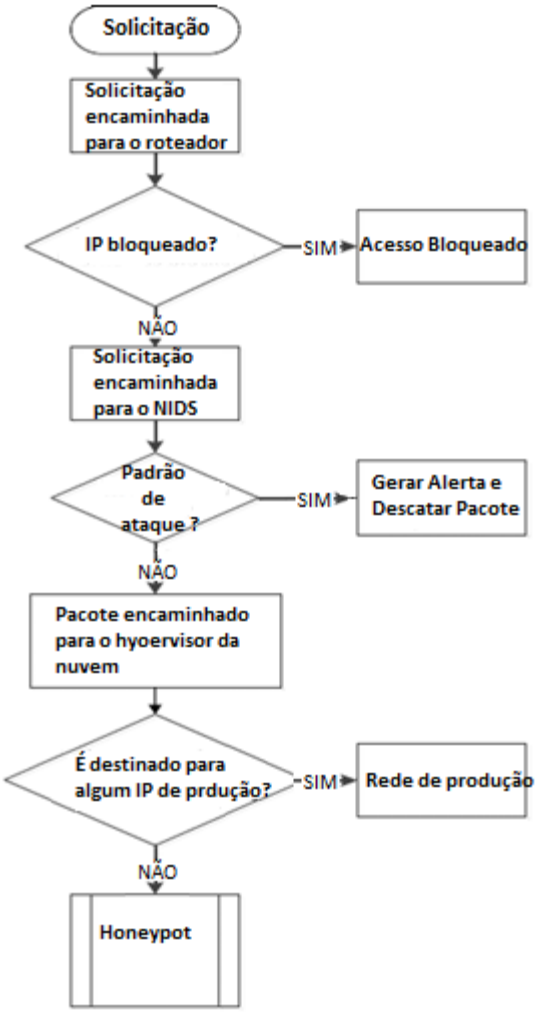

Figura 2.19: Diagrama do fluxo dos caminhos escolhas para pacotes de entrada [10].

O *honeypot* é uma instância configurada com o Honeyd que fica em outro grupo de acesse diferente das instâncias de produção assim reduz o risco de o

ataque ter algum efeito sobre elas ou o atacante ter acesso aos dados das instâncias de Produção. A Figura 2.20 mostra a arquitetura da ideal para a rede.

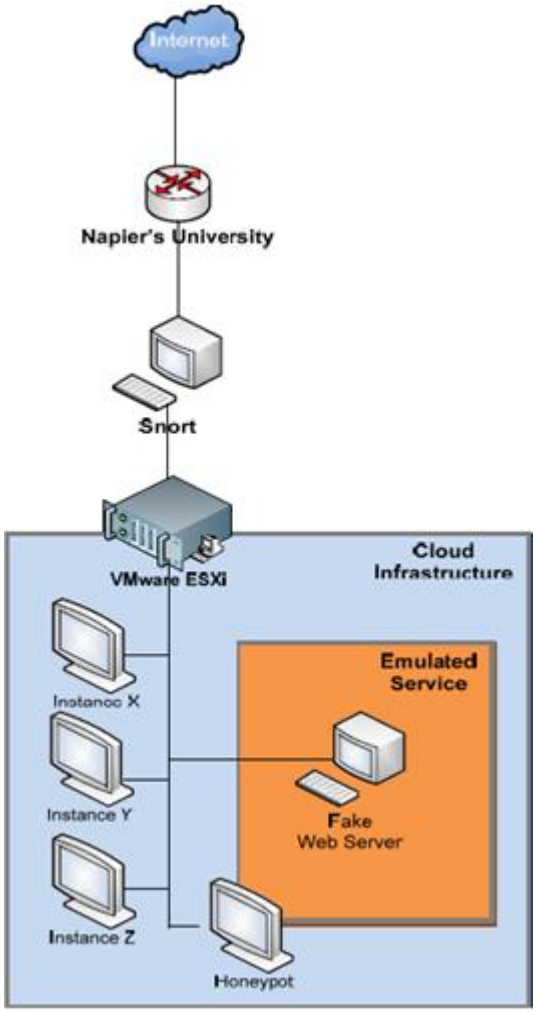

Figura 2.20: Arquitetura da rede ideal em [10].

Honeypot as a Service in Cloud

M. Balamurugan e B. Sri Chitra Poornima [9] demonstra a implementação de um *Honeypot* em um ambiente de nuvem para fornecer segurança e lucro ao negócio. Para uma operação eficiente de tal *honeypot* em nuvem, foi implementado independente do s*oftware* usado para configurar o ambiente de nuvem criando um serviço que nomeou de HaaS (*Honeypot a as Service*). Assim, uma nuvem plataforma *honeypot* independente é a melhor maneira de melhorar a segurança na

nuvem segundo eles, uma vez que, os “nós” controles do *honeypot* ficam separados do controlador da nuvem.

Segundo [9] a implementação de *honeypot* na nuvem é inteiramente diferente uma vez que o meio ambiente é diferente. O h*oneypot* pode criar uma população de instâncias virtuais na rede da nuvem usando endereços os IP não atribuídos. Ele pode ser usado para simular qualquer outro serviço disponibilizado com TCP ou UDP. Cada instância pode ser configurada com um conjunto de serviços emulados e sistema operacional.

A implementação tem os seguintes componentes:

- Controlador da nuvem (*Cloud Controller*) que é a unidade de controlo centralizado para um ambiente de nuvem. Ele é usado para gerenciar e expor os recursos virtualizados (máquinas, rede e de armazenamento) por meio de APIs voltadas para o usuário;
- Controlador de *cluster* (*Cluster Controller*) está presentes em um ambiente de nuvem para controlar os vários *clusters,* que agem como um *Virtual Machine Manager* (VMM) controlando a execução de várias máquinas virtuais em execução nos nós e sua interação com os usuários. Juntamente fica o controlador de armazenamento que é usado para fornecer blocos de armazenamento em nível de dados. Sob cada controlador de cluster dirige-se um grupo de controladores de nós que são usados para gerenciar a inicialização, execução e extinção dos serviços prestados pelas instâncias de máquina virtual;
- Controlador de *honeypot* (*Honey Controller*) é um controlador de nó que é separado dos outros que prestam serviços reais, fornecendo um controlador de cluster separado para as instâncias *honeypot*. Esta separação é altamente essencial porque o comprometimento das instâncias *honeypot* não deve afetar os outros controladores e instâncias em execução na nuvem. Este controlador de cluster em que os *honeypots* estão em execução é chamado de controlador de honeypot. Pode haver muitos casos simulados correspondentes as instâncias em execução na nuvem para fornecer um honeypot de alta interação;
- Motor de filtro e redirecionamento (*Filter and Redirection Engine*-FRE) tem como objetivo separar o tráfego de usuários mal-intencionados de usuários

legítimos e redirecioná-los para o controlador de *honeypot*. O tráfego é analisado com base em algumas assinaturas sinalizadas e as portas falsas em execução no *honeypot*;

- Sistema de armazenamento de logs armazena todos os registros e as informações sobre os agressores é que são registrados no FRE antes de o tráfego ser redirecionado para controlador de *honeypot*. Esses registros são armazenados em um sistema de armazenamento de *log* centralizado no controlador de nuvem e normalmente contêm informações sobre o endereço IP do atacante, tipo, data, hora e frequência de ataque.

A Figura 2.21 mostra um modelo de *honeypot* implementado em um ambiente de nuvem.

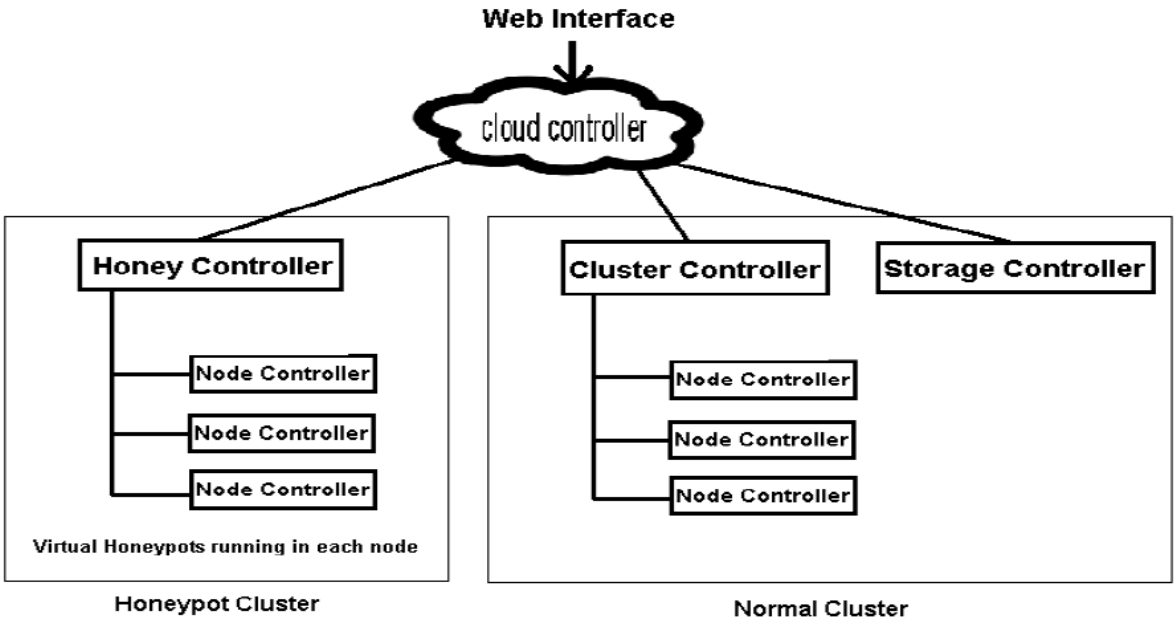

Figura 2.21: modelo de *honeypot* implementado em um ambiente de nuvem [9].

AutonomicSec: Um Mecanismo Autonômico para Segurança de Redes baseado em Decepção

Teles em [10] propões um mecanismo autonômico para proteção de redes de computadores que pode ser dividido em três partes:

- A primeira é framework autonômico, seguindo o modelo MAPE-K [];
- A segunda um ciclo autonômico para gerar regras de firewall baseadas em *logs* de *honeypots*;

- A terceira um ciclo autonômico para restaurar as *honeypots* virtuais consideradas comprometidos.

A Figura 2.22 mostra o fluxo de comunicação dentro do *framework* AutonomicSec [10].

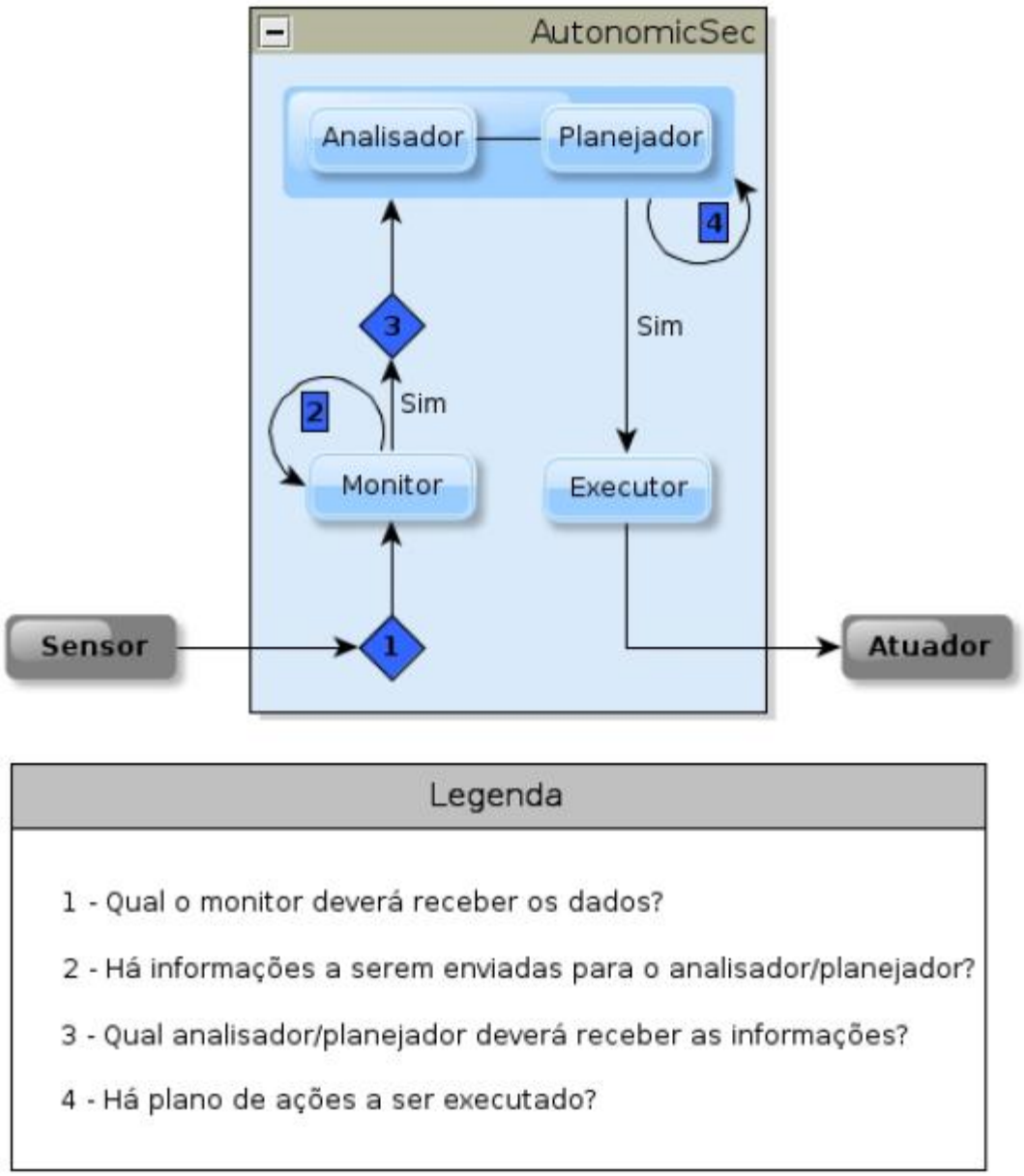

Figura 2.22: Comunicação entre componentes do *Framework* [10].

O *framework* da base para a construção do sistema de autonômico para proteção de rede utilizando *firewall* com regras baseadas em interações sofridas por *honeypot*. Tira as informações de interação de *hosts* maliciosos que interagiram com o *honeypot* e carrega ACLs para barrar a interação com os servidores reais.

A tabela 1 mostra um comparativo com os trabalhos propostos entre os modelos apresentados pelos trabalhos.

Trabalho	Característica Autonômica	Tipo de *Honeypot*	Aplicação na em Nuvem
[6]	Sim	Baixa interação	Não
[7]	Sim	Baixa interação	Não
[8]	Sim	Baixa interação	Não
[5]	Sim	Baixa interação	Sim
[9]	Sim	Alta interatividade	Sim
[10]	Sim	Baixa interação	Não

Quadro 1: Comparativos dos Trabalhos Relacionados

2.5 Considerações Finais

Neste capítulo, foram apresentadas as seções sobre Computação em Nuvem, Metodologia de Decepção, Computação Autonômica e trabalhos relacionados.

Sobre computação em nuvem foram apresentados os principais conceitos, características, modelos de implantação, modelos de prestação de serviço, papeis na solução de computação em nuvem, desafios encontrados e segurança voltado a computação em nuvem. A seção sobre Computação Autonômica forma mostrados os conceitos, características e arquitetura bem como sua utilização. Em Metodologia de Decepção foi mostrado seu uso, os tipos de *honeypot* como cada tipo de ser usado.

Por fim os trabalhos relacionados, onde foram mostrados os trabalhos que contribuirão diretamente para o desenrolar da dissertação. Mostrando metodologias de para desenvolvimento de sistemas autônomos, computação em nuvem autônomas e a utilização de metodologia de decepção aliada a sistemas autonômicos.

3 MODELO PROPOSTO

Este capítulo apresenta a proposta de um mecanismo autonômico para proteção de “servidores instâncias” e *Vlan’s* em ambiente de Computação em Nuvem. Uma visão geral do mecanismo e do seu funcionamento será apresentada, bem como *framework* autonômico, um ciclo autonômico que tem a funcionalidade de gerar regras de acesso as *Vlan’s* e a nuvem baseadas em logs de *honeypots* e ciclo autonômico responsável por manipular dinamicamente *honeypots* virtuais que são considerados comprometidos.

3.1 Arquitetura proposta

A arquitetura proposta para esta dissertação tem por objetivo fornercer características autonômicas a uma nuvem computacional, a fim de assegurar as máquinas virtuais (instâncias) para manter a integridade da nuvem e diminuindo a interação humana e seu tempo de manutenção, já que o ambiente de computação em nuvem tem como uma de suas características a cobrança sob demanda, então quando mais rápido for à resposta a uma invasão menos tempo o serviço ficaria fora do ar ou sofrendo ataques. O mecanismo auxiliará na defesa contra ataques causados por usuários mal intencionados da própria nuvem, que usam suas instâncias para sondar outras instâncias, podendo capturar dados ou lançar ataque para comprometer o seu funcionamento.

O modelo de mecanismos proposto age na segurança em nível de IaaS (Infraestrutura como Serviço), para isso foi utilizado o *framework* autonômico AutominicSec [10] e um *honeypot*, que gera os *logs* utilizados pelo *framework* para gerar novas políticas de grupos e acesso a nuvem e as *Vlans* da nuvem. Ele funciona recebendo dado dos sensores previamente cadastrados juntamente a seu monitor específico, esse dado passa para o analisador/planejador e esse para o seu executor que aplicará o conjunto de ações definidas no analisador/planejador. Caso não haja ocorrência nos sensores o monitor fica em modo de espera.

O modelo de prestação de serviço escolhido foi o IaaS, onde, a nuvem disponibiliza uma infraestrutura, em forma de instância, para instalação de qualquer plataforma. O *honeypote* é uma instância pré-configurada de sistemas para servidor ou *descktop*, que ficam disponíveis a todos os usuários da nuvem.

Cada grupo de usuário tem suas instâncias localizados em *Vlan's* (redes virtuais) da nuvem. O acesso as *Vlan's* e as instâncias é regido por políticas de grupos de usuários configuradas no controlador da nuvem as quais, apenas os usuários administradores do grupo e o administrador da nuvem tem permissão de acesso e edição.

A figura 3.1 ilustra o modelo de ambiente interno proposto para nuvem. O ambiente é composto por uma ou mais *Vlan's* com suas instâncias reais e as instâncias *honeypots*, sendo que a opção de ativar ou não uma ou mais instâncias *honeypot* fica a critério do administrador do grupo. Desta forma o *honeypot* é caracterizado como um serviço da nuvem, podendo ser utilizado como forma de gerar segurança aos usuários e lucros ao provedor da nuvem, que pode cobrar pelo serviço.

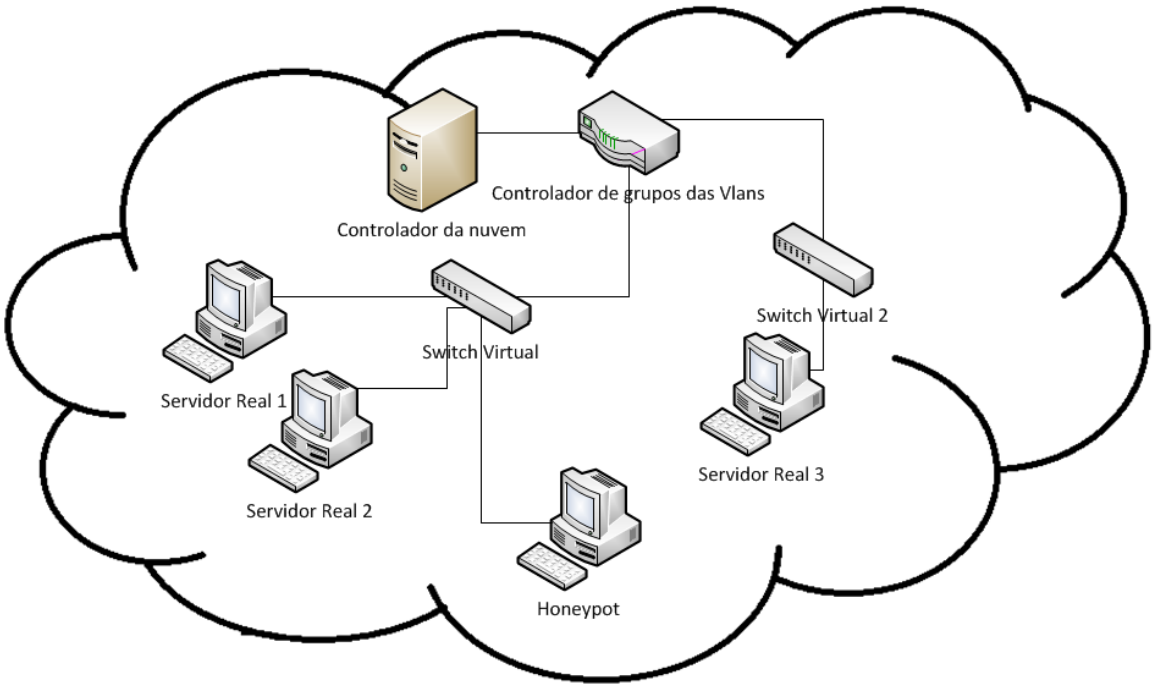

Figura 3.1: Ambiente da nuvem proposto para sistema.

Na arquitetura proposta o AutonomicSec-Cloud fica localizado no controlador da nuvem, como mostra a figura 3.2, para que possa monitorar todos os *honeypot* da nuvem mesmo dentro de diferente *Vlans.* O número de instâncias honeypot fica a critério do administrador da Vlan. A Figura 3.2 mostra o modelo de

arquitetura interna da nuvem com o AutonomicSec-Cloud instalado junto ao controladorda nuvem.

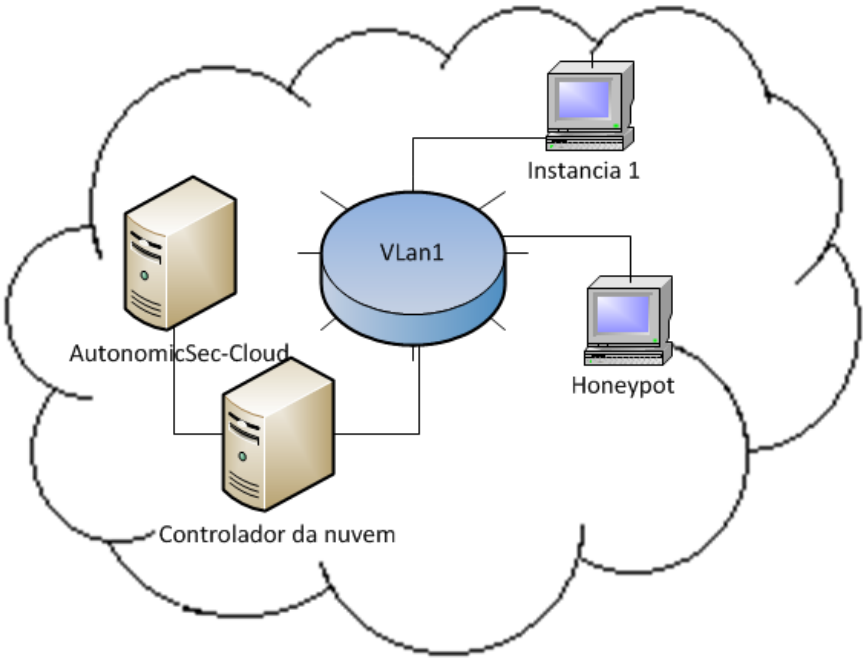

Figura 3.2: Modelo de arquitetura interna da nuvem com o AutonomicSec-Cloud.

3.2 Caracterização do Modelo Proposto

A utilização do *framework* se dá em uma nuvem Privada, devido a necessidade de acesso a controladora da nuvem, para que as ações referentes a bloquei de usuário, finalização e incialização de instância possam ser feitos de forma autonômica. Nas nuvens públicas disponíveis no mercado apenas só é dado permissão de edição de algumas políticas de grupos, finalização e inicialização de instâncias de forma manual.

Assim com o trabalho do Teles [10], no qual este foi inspirado, existem quatro ações que mostram a obtenção de resultado satisfatório que são ela:

- A extensão do *framework* AutonomicSec [10] para ser usado em ambiente de computação em nuvem;
- Criação de um clico autonômico para segurança de instâncias em um ambiente de computação em nuvem, com a utilização de listas de permissão e controle de acesso a partir logs gerados por *honeypot*;
- Criação de um clico autonômico para restauração de instância *honeypot* comprometidas.

- Disponibilização do serviço de *honeypot* na nuvem gerando maior confiança e segurança para as instâncias (servidores e estações de trabalho) que estão disponíveis na nuvem.

Em relação a melhoria na segurança e obtenção de características autonômicas, o *framework* conseguiu todas as características desejáveis [46]:

- Autoproteção porque como o primeiro clico autonômico ele consegue se defender contra alguns ataques de forma autônoma, gerando assim uma segurança maior no uso da computação em nuvem;
- Autocura: ele é capaz de restaurar suas instâncias para um estado inicial por intermédio de snapshot (método que salva uma imagem do estado incial da instância);
- Autoconfiguração: é introduzido novas configurações na nuvem para tirar as instâncias que interagiram com o honeypot de funcionamento;
- Autoaprendizado: a cada nova interação são criadas novas regras de novas politicas de grupo.

O *honeypot* foi configurado em uma instância Linux, na qual foi instalado o *framework* HoneyD que para efeito desse trabalho estava configurado para interagir nas portas 80 e 21, respectivamente um servidor *web* e um servidor FTP, essas portas foram escolhidas para os teste mas qualquer outra porta pode ser simulada.

Qualquer interação com o *honeypot* é considerada maliciosa, por isso até mesmo um *ping* nessa instância é monitorado, porém foram feitos filtros um para eliminar falsos positivos lendo o IP de destino, caso esse IP seja de Broadcast ele descarta essa informação e não marca como interação. A nuvem Eucalytus usada para montagem do protótipo de teste, as instâncias de tempos em tempos disparam sequências de chamadas Broadcast para verificar se as instâncias dentro da mesma *Vlan* ainda estão ativas, isso acarretaria em inúmeros falsos positivos e se essa operação realizada em um intervalo de tempo pequeno do bloqueio de todas as instâncias da nuvem.

3.3 Framework

O nosso *framework* foi feito sobre o modelo do *framework* de [10], onde ele segue o modelo de MAPE-K [47], para autonomia em sistemas de segurança de rede. A arquitetura do framework e a mesma adotada em [10]. A Figura 3.3 ilustra o modelo de arquitetura do AutonomicSec[10], onde cada cor indica um sensor especifico para determinanda ação, ligado a um monitor especifico, a um analisador especifico, a um planejador especifico, a um executor comum e a atuadores especificos.

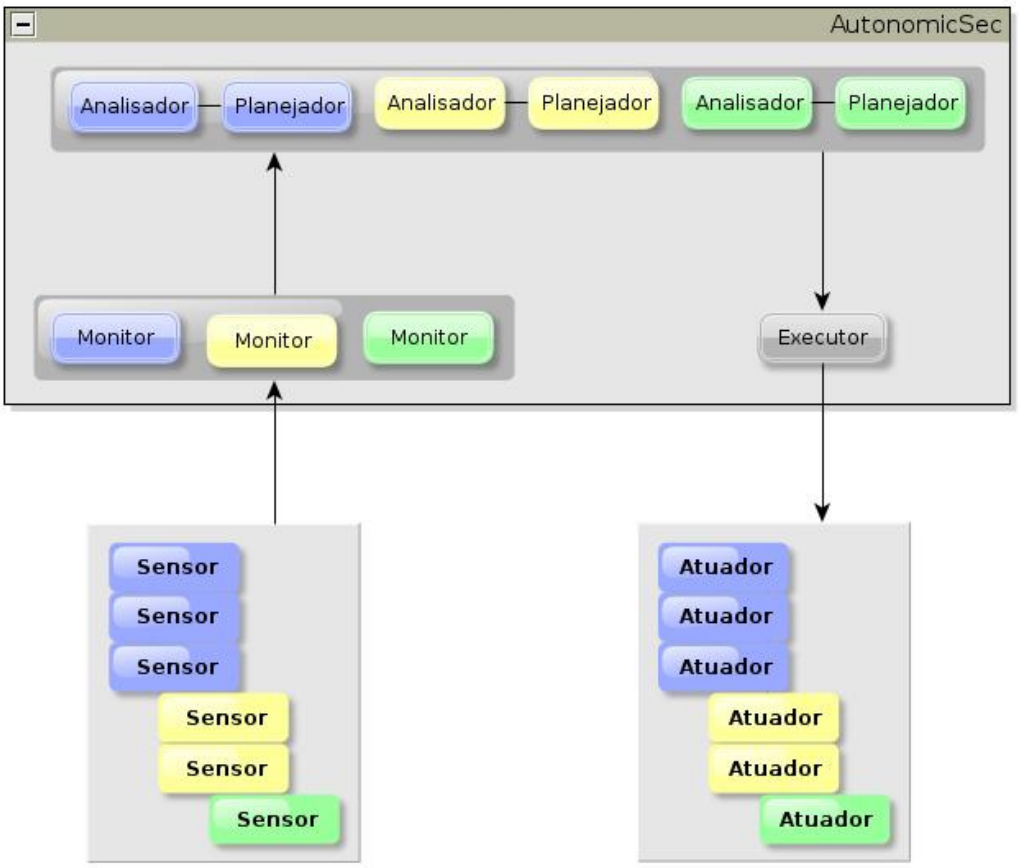

Figura 3.3: Arquitetura do Framework AutonomicSec [10].

Utilizando este *framework* foram configurados dois ciclos autonômicos, o primeiro responsável por verificar se houve interação como o *honeypot*, monitorar esta interação e após a coleta das informações necessárias sobre a interação finalizar o *honeypot*, o segundo responsável por inserir novas regras no controlador da núvem baseadas no *log* de interações do *honeypot*.

No primeiro ciclo autonômico o sensor do *honeypot* fica verificando em intervalos de tempo o arquivo de log do *honeypot,* para saber se ele teve seu tamanho alterado, o que indicaria uma interação. Caso tenha alterado de tamanho, ele executa a classe de manipulação de arquivo. Na classe de manipulação de arquivo o log é minerado, filtrando e deixando só o que é relevante. É verificado se realmente houve um interação, os IPs dos suspeitos são classificados e qualificados de acordo com o número de vezes que ele interagiu e depois é feita uma checagem para saber se as informações foram salvas no arquivo de suspeitos. Após o sensor ter feito a mineração, ele retira do *log* as interações que não são de IPs considerados suspeitos e envia para o monitor apenas o *log* tratado.

A figura 3.4 ilustra o diagrama de fluxo do pacote de log do honeypot.

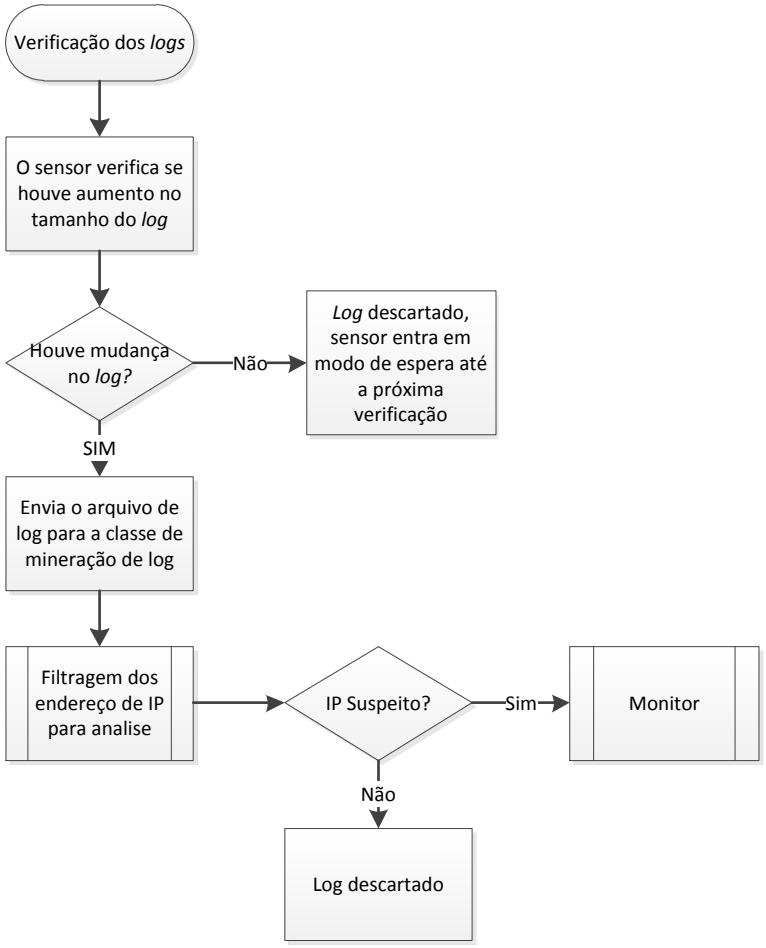

Figura 3.4: Diagrama de fluxo do pacote de log do honeypot.

O monitor do *honeypot* faz uma varredura no log já minerado vindo do sensor, para procurar os IPs e que realizaram interação com os protocolos ou

serviços monitorados . No esquema de teste montado, foi utilizado um honeypot de baixa interatividade, mais utilizado em meio acadêmico [5], mas qualquer outro tipo de *honeypot* (alta e média interação) podem ser usados. Após ter feita filtragem salvando apenas os IPs que fizeram interação com o protocolo de seu interesse envia esse arquivo para o analisador. A figura 3.5 mostra o algoritmo do monitor do *honeypot* responsável pela filtragem.

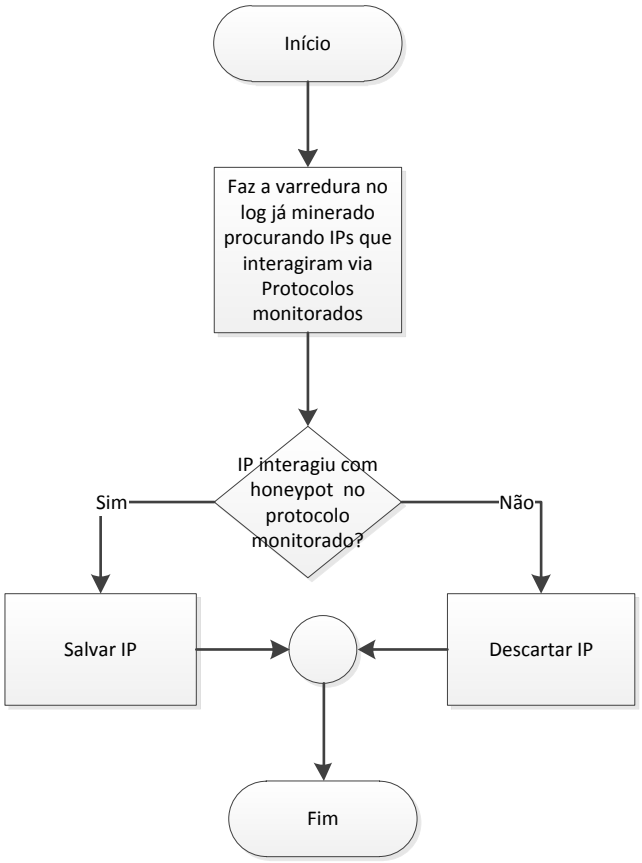

Figura 3.3: Algoritmo do monitor do honeypot.

O analisador do *honeypot* usa a classe instância para gerar um arquivo de configuração com informações das instâncias (EMI, que é o nome da imagem dentro da nuvem, e o ID, que é o número de identificação da imagem na nuvem) com os IPs da lista de suspeitos geradas pelo monitor. Ele faz a verificação se essas instâncias fazem parte de lista de liberados (*White List*). O analisador usa estas informações para gerar regras para bloquear as instâncias consideradas supeitas enviando-as para o atuador. A figura 3.4 ilustra algoritmo do analisador do honeypot responsável por gerar regras para bloqueio das instâncias.

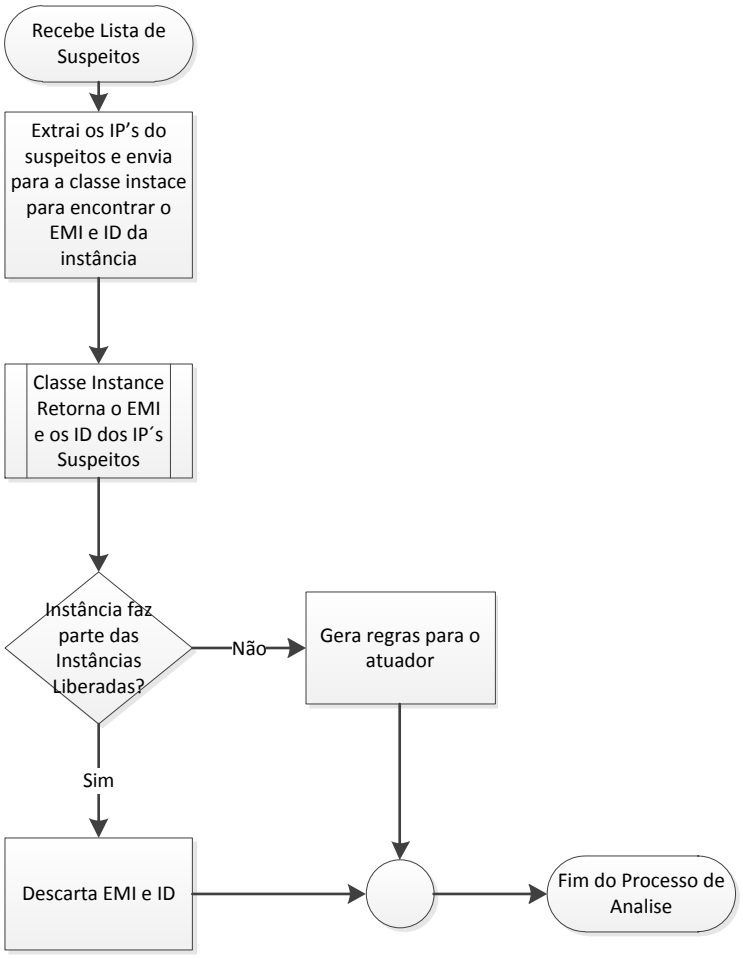

Figura 3.4: Algoritmo do analisador do honeypot.

O atuador do *honeypot,* por sua vez, aplica as regras geradas, criando um arquivo de comandos com as credenciais necessárias. Também é criado um preparador que é um arquivo que contém as permissões de execução para os arquivos de comando do atuador. A Figura 3.5 ilustra o algoritmo do método "execute" do atuador que tem por finalidade executar os arquivos de comandos, criados no atuador, na controladora da nuvem. Esse arquivo contém comandos para finalizar a instância (máquina virtual) do *honeypot.*

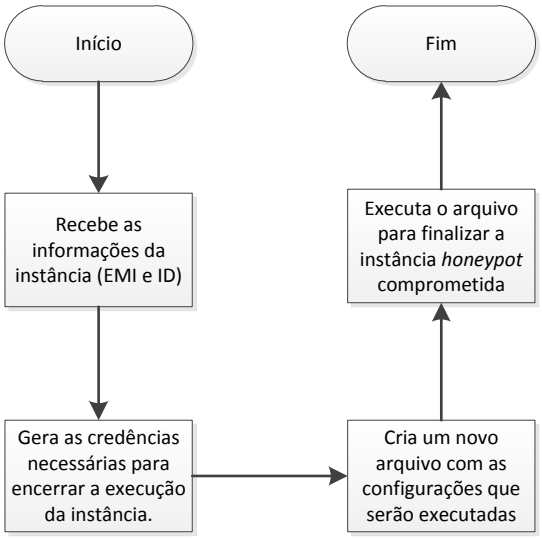

Figura 3.5: Método execute do atuador do *honeypot*.

O segundo ciclo autonômico não se preocupam com as interações do *honeypot*, mas sim com a geração e a aplicação das regras na nuvem e o retorno do *honeypot* ao estado inicial. Para isso monitora o log de regras, visto que, se o log de regras sofreu alteração significa que uma ação de coesão a uma instância foi efetuada e o estado inicial do *honeypot* deve ser retornado.

O sensor do segundo ciclo autonômico é feito a verificação do arquivo de regras geradas pelo analizador do honeypot no primeiro ciclo. Caso haja alteração, ele lê o arquivo de regras em busca da lista de liberados (*White List*) para pegar o EMI e o ID da controladora da nuvem e da instância *honeypot*. Na Figura 3.6 pode ser observado que o sensor do *framework* monitora o arquivo de configuração criado no analisador do *honeypot*.

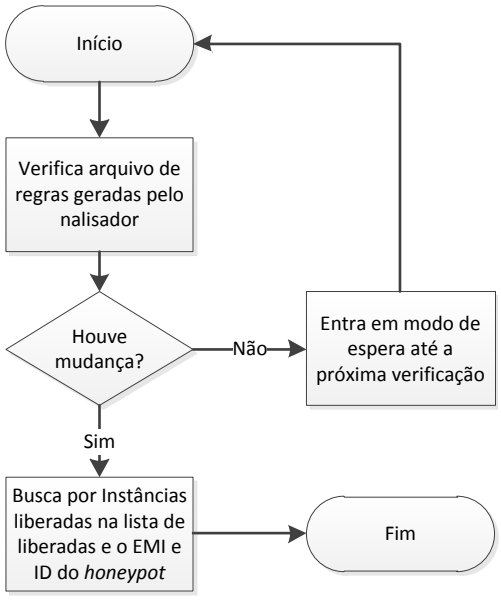

Figura 3.5: Algoritmo sensor do framework.

O monitor do segundo ciclo é tem a função de receber os dados do sensor e verifica o EMI e o ID do *honeypot*. O EMI para poder instancia-lo e ID para encera-lo, para isto seram utilizados nas novas regras criadas no analisador, possibilitando que vários *honeypot* sejam inseridos e apenas o que comprometido seja recuperado. A figura 3.6 ilustra o algoritmo monitor do segundo ciclo autonômico visando a manutenção da estrutura do sistema.

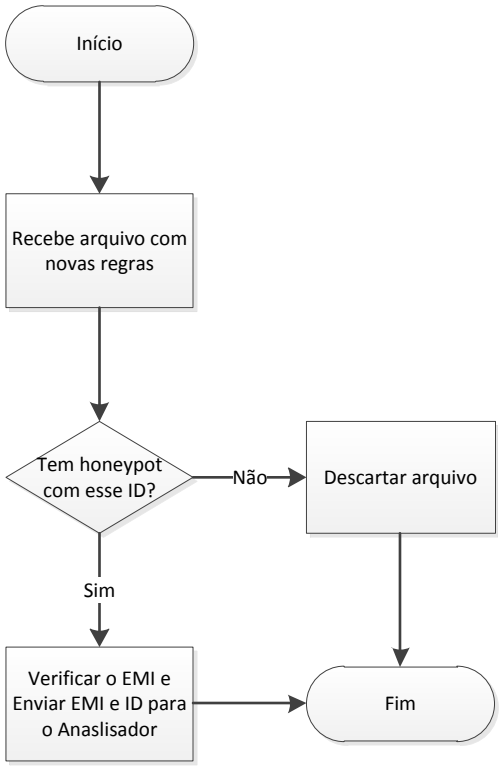

Figura 3.6: Algorítmo do monitor do segundo ciclo autonômico.

O analisador do *framework* retira o ID das instâncias envolvidas nas interação, a partir do arquivo enviado pelo monitor, e gera regras para finalizar a instância e EMI para iniciar uma nova instância. Na Figura 3.7 o algoritmo do analisador do *framework*.

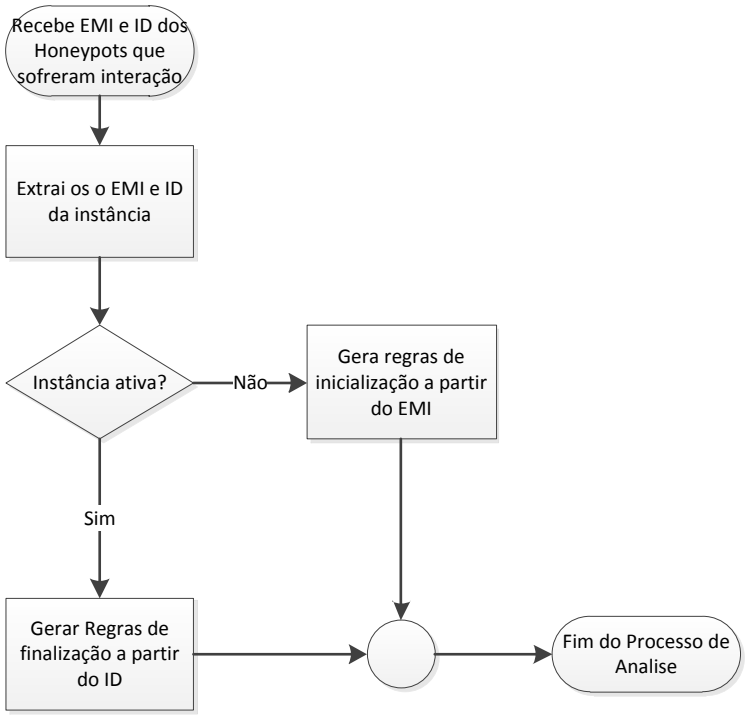

Figura 3.7: Algoritmo analisador do framework.

O atuador do *framework* é quem vai verificar as instâncias para gerar uma nova configuração, gerar os atuadores para finalizar a instancia *honeypot* comprometida e o atuador para instanciar uma nova a partir das instruções passadas pelo analisador.

3.4 Considerações Finais

Neste capitulo foi proposto um modelo de segurança autonômica para nuvem computacional baseado na utilização de *honeypot*. Esse modelo foi desenvolvido pensando em prover a segura em nível de IaaS.

A fim de conseguir as característica autonômicas desejadas a nuvem o *framework* AutonomicSec foi estendido e modificado para se integrar a nuvem uma vez que ele foi desenvolvido para redes reais. As modificações foram apresentadas e explicadas para esclarecer o seu funcionamento.

4. CENÁRIO E TESTES DO MODELO

Neste capito será apresentado implementação de um protótipo para testar o modelo de segurança autonômica proposto, faz parte deste protótipo uma nuvem computacional privada e a instalação do framework modificado.

Primeiramente começaremos falado do senário do teste, depois da nuvem implantada e o resultado dos testes.

4.1 Cenário

O experimento se deu no Laboratório de Sistema em Arquiteturas Computacionais no campus da Universidade Federal do Maranhão. Onde foi usado dois Computadores sendo um onde foi montado a nuvem EUCALYPTUS, que é uma nuvem privada, e o outro apenas para acessar as instâncias da nuvem. A Quadro 1 mostra as configurações dos computadores.

Máquina	Ip	Hardware	Configuração
Server_Cloud	192.168.27.81	Core i7, 8GB RAM, HD de 1 TB	Eucalyptus FastStart 3.3.1
Labsac	192.168.27.78	QuadriCore 2GB RAM HD de 500GB	Ubuntu 12.04

Quadro 2: Configuração do Hardware Utilizado.

Para o senário foi escolhida usamos um microcomputador, Server_Cloud, onde foi instalada a nuvem EUCALYPTUS [49] em com a instalação FastStart 3.3.1 do tipo Cloud_in_Box, que tem suas limitações quanto a quantidade de usuários, de máquinas virtuais disponíveis, porém é pré-configurada e de fácil instalação. O ambiente proposto pede uma controladora de nós separada para os *honeypot,* porém isso não é necessário no nosso ambiente de teste. A Figura 4.1 mostra o modelo do nosso ambiente montado para de teste.

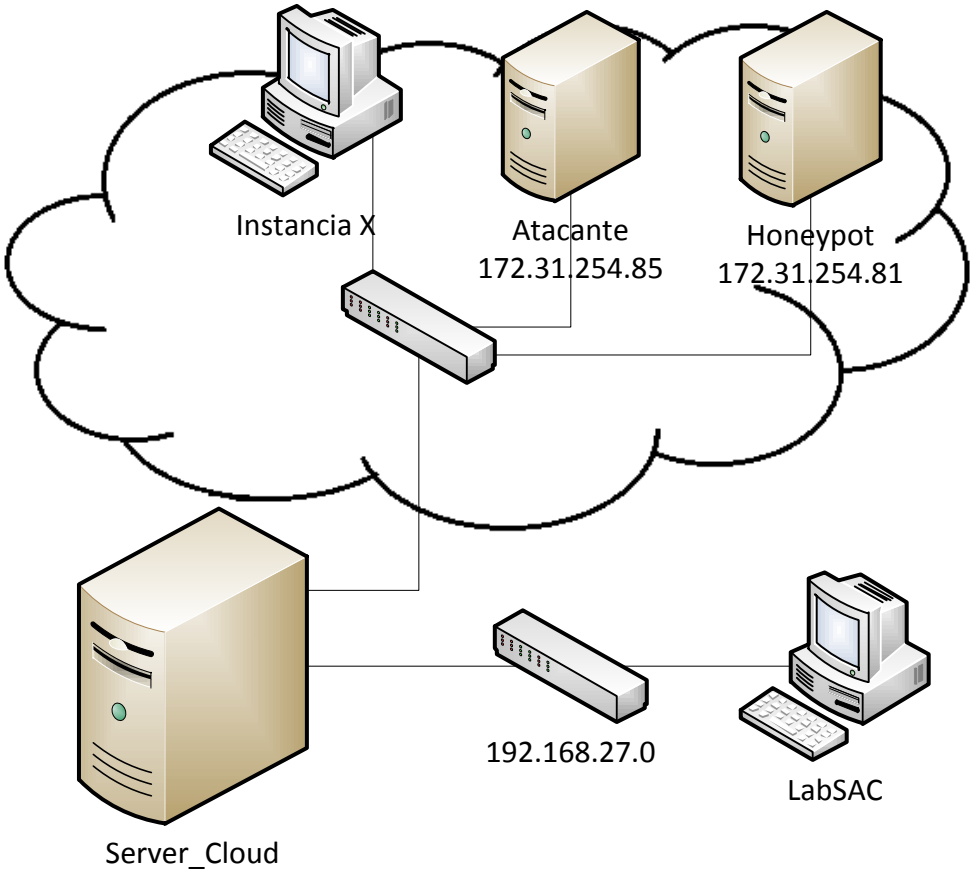

Figura 4.1: Ambiente montado para de teste.

4.2 Nuvem EUCALYPTUS

Para o teste foi escolhido o EUCALYPTUS como nuvem computacional poque tem estrutura open-source que usa infra-estrutura computacional e de armazenamento que comumente é usada por grupos de pesquisa acadêmica para fornecer uma plataforma, modular e aberto a instrumentação experimental e estudo [49]. O sistema permite que o usuário inicie, controle o acesso e terminar as máquinas virtuais inteiras, usando uma emulação de SOAP do Amazon EC2 e interfaces de "*Query*". Sua estrutura é dividida em quatro componetes básicos que são:

- Controlador de nó (*Node Controller*) que controla a execução da instâncias de VMs no host onde são executados, permitindo a inspeção e a finalização das mesmas.

- Controlador de Cluster (*Cluster Controller*) que controla as informações sobre as VMs (horário de execução) em controladores de nó específico, além de administrar as Vlans das instâncias.
- Controlador de Armazenamento (Walrus) é um serviço de armazenamento de dados que no EUCALYPTUS e implemento pela interface S3 da Amazon, fornecendo um mecanismo para armazenar e acessar imagens de máquinas virtuais e dados do usuário.
- Controlador da Nuvem (*Cloud Controller*) é o responsável por gerenciar a entrada dos usuários e administradores da nuvem. Faz todo gerenciamento de recurso através das solicitações feito as outras controladoras.

A Figura 4.2 mostra uma representação hierárquica da estrutura do EUCALYPTUS

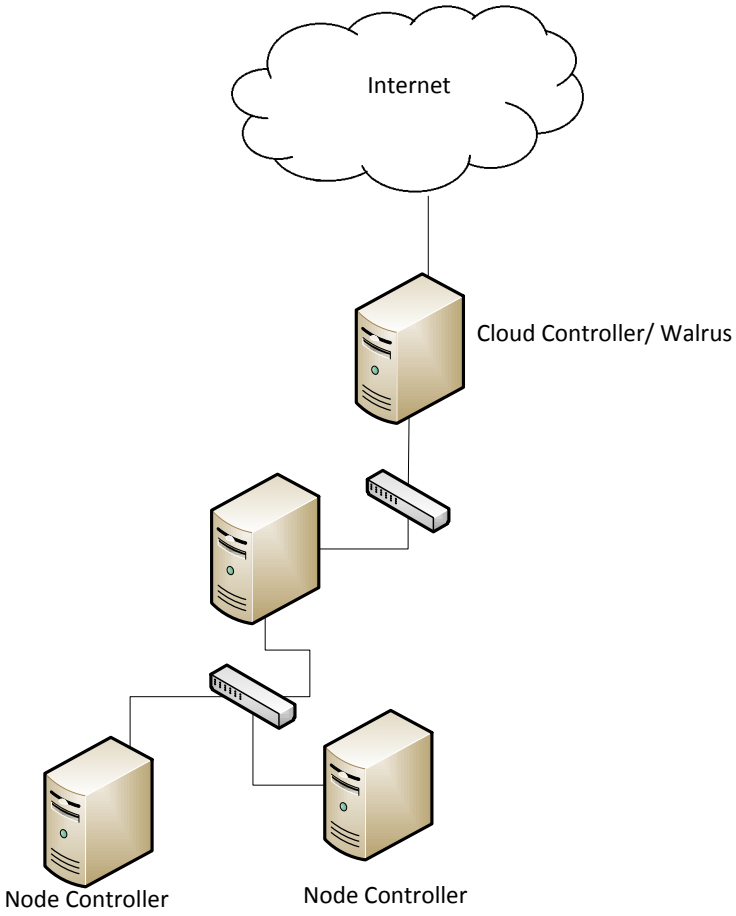

Figura 4.2: Estrutura hierárquica do EUCALIPTUS.

Devido a sua plataforma modular pode ter diversas arquitetura de implantação, desde uma onde cada controladora fica em um host diferente até uma versão onde todas ficam dentro da mesma máquina. O que vai determinar disponibilidade de recurso e a demanda de uso do cliente. No caso da nossa nuvem de teste foram instalados todos os componentes em uma única máquina (“Server_Cloud”).

A imagem 4.3 mostra a interface *DashBoard* que é utilizada para administração da nuvem via *browser.*

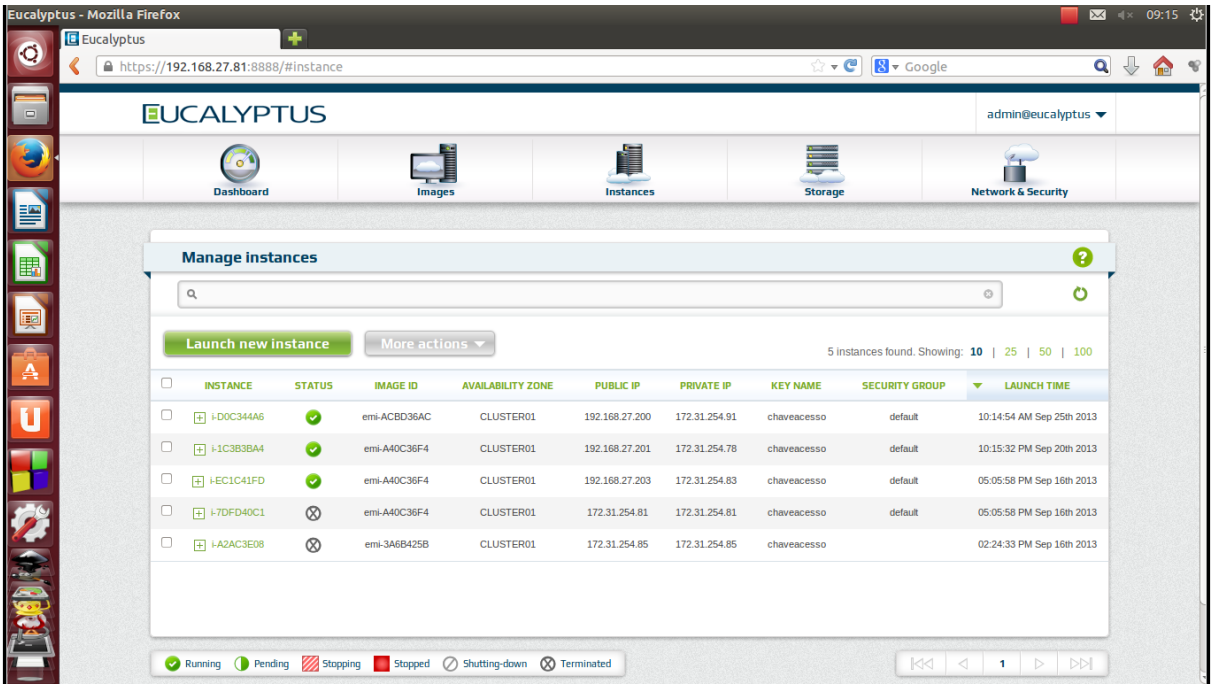

Figura 4.3: administração da nuvem via browser.

4.3 Teste dos ciclos autonômicos

Os testes tiveram como objetivo mostrar a eficácia dos ciclos autonômicos, configurados no *framework*, na defesa da nuvem contra ataques entre instância da própria nuvem. Os ciclos autonômicos configurados foram dois, baseados no trabalho de [10], como mostrado no capítulo anterior, o primeiro visando a defesa contra o ataque e possíveis novos ataques, e o segundo ciclo autonômico é responsável por reestabelecer a estabilidade e assegurar que o sistema não fique comprometido.

4.3.1 Ataque na instância *honeypot* (Primeiro Ciclo Autonômico)

O primeiro teste representa o primeiro ciclo autonômico onde o *honeypot* sofre de interação de outra instância da rede e a partir dai esta interação deve ser percebida pelo sensor do *honeypot*, que indicará para o monitor do *honeypot* que houve uma interação e passará para o monitor do *honeypot* o *log* da interação. A Figura 4.4 ilustra interação entre a instância atacante e o *honeypot.*

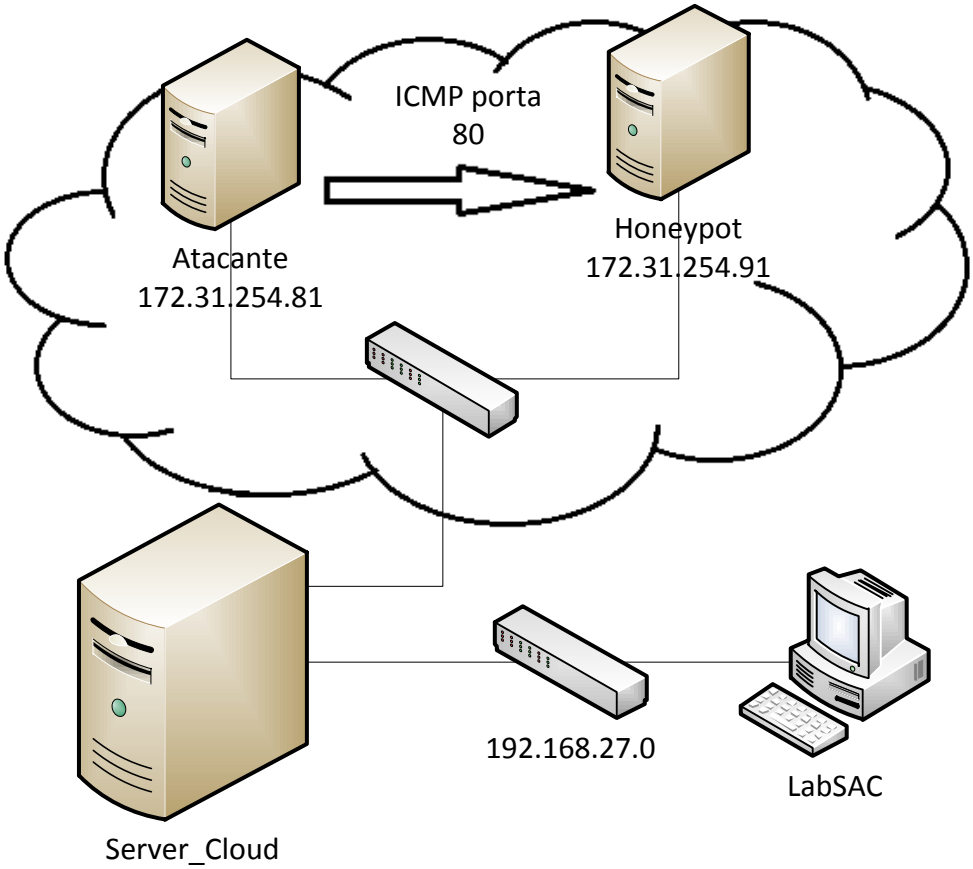

Figura 4.4: interação entre a instância atacante e o *honeypot.*

Quando a interação ocorre, dá inicio a fase de monitoramento. É dado um tempo pelo sensor, esperando que haja mais interação. Ao fim desse tempo, o sensor do *honeypot* representado no *framework* pela classe *SensorHP* lê o *log* no arquivo *VM.txt* gerada pelo *Honeyd* instalado na instância *honeypot*. Este arquivo contém todas as interações que o *honeypot* tem com a rede. É aplicado um filtro para que ele descarte informações como mensagem de *broadcast,* não pondo-as como uma tentativa de interação, uma vez que para a manutenção das instâncias e das VLans corriqueiramente uma instância dispara uma sequência de mensagens

de broadcast. A Figura 4.5 ilustra um arquivo *Vm.txt,* que contém o *log* com a interação(varredura na porta 80) entre as instâncias do *honeypot* com *IP* 172.31.254.91 e a instância atacante com *IP* 172.31.254.81.

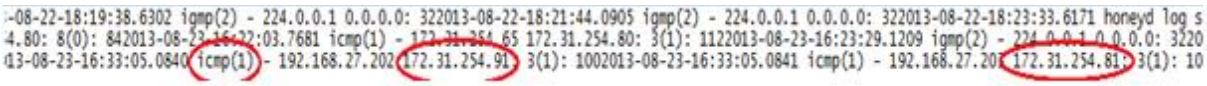

Figura 4.5: Arquivo *Vm.txt.*

A classe *SensorHp* então lê o arquivo *Vm.txt* e cria um arquivo *suspeitos.txt*. Logo em seguida, chama os métodos da classe *ManipulateFile* ()para fazer a checagem do *log,* verificar se esse *IP* não está na lista de liberados (*whiteList*), além de minerar o *log,* tirando assim dele os dados que não são de interesse. No nosso teste o arquivo *Vm.txt* é o da Figura 4.5 após a mineração ele envia o *log* já minerado para a Monitor do *honeypot* que é representado pela classe *MonitorHP* como já explicado na seção 3.3. Ele envia o *IP* desse suspeito para o analisador representado pela classe "*AnalyzerHP*" que cria um objeto da classe "*Instancia"* que criará o arquivo "*configuracao.txt"* como na figura 4.6. A classe "*Instancia*" cria esse arquivo a partir do arquivo "*atuardoInstancia*", que é um arquivo criado pela "Instancia", figura 4.7 mostra o conteúdo do arquivo "*atuardoInstancia*" o qual contém um comando para a nuvem descrever todas as instancias, porém no arquivo "*instances"* só ficaram as que estão em execução (*running*) como mostra a Figura 4.8.

Figura 4.6: Arquivo configuracao.txt.

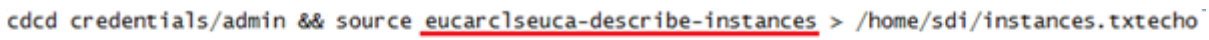

Figura 4.7: Arquivo atuadorInstancia.

Figura 4.8: Arquivo *instances.txt*.

Após todos os passo anteriores são criadas as regras que serão executadas pelo atuador representado pela classe *ActionHP*. As regras são arquivos contendo as informações necessárias para finalizar a instancia que está oferecendo risco a nuvem, ou seja, a que interagiu como o *honeypot*. A figura 4.9 mostra o arquivo de regras do *ActionHP* onde ele utiliza o *ID* (marcado de vermelho) da instância para finaliza-la.

```
#!/bin/bashcdcd credentials/adminsource eucarceuca-terminate-instances i-D00B46DB
```

Figura 4.9: Arquivo *"atuadorhp.txt"*.

As Figuras 4.9 mostra o começo do teste com o acesso a nuvem sendo feito no computador "*LabSAC*" utilizando a instancia com IP 172.31.254.81 antes da interação com a instância o *honeypot* com IP 192.31.254.91. A Figura 4.10 e 4.11 mostram a finalização da instância com o IP 172.31.254.81(que é a instância que interage com o *honeypot*). Na 4.10 a instância ainda está em execução e gerando chamadas através do protocolo icmp, como mostra o a janela de terminal do lado direito da figura.

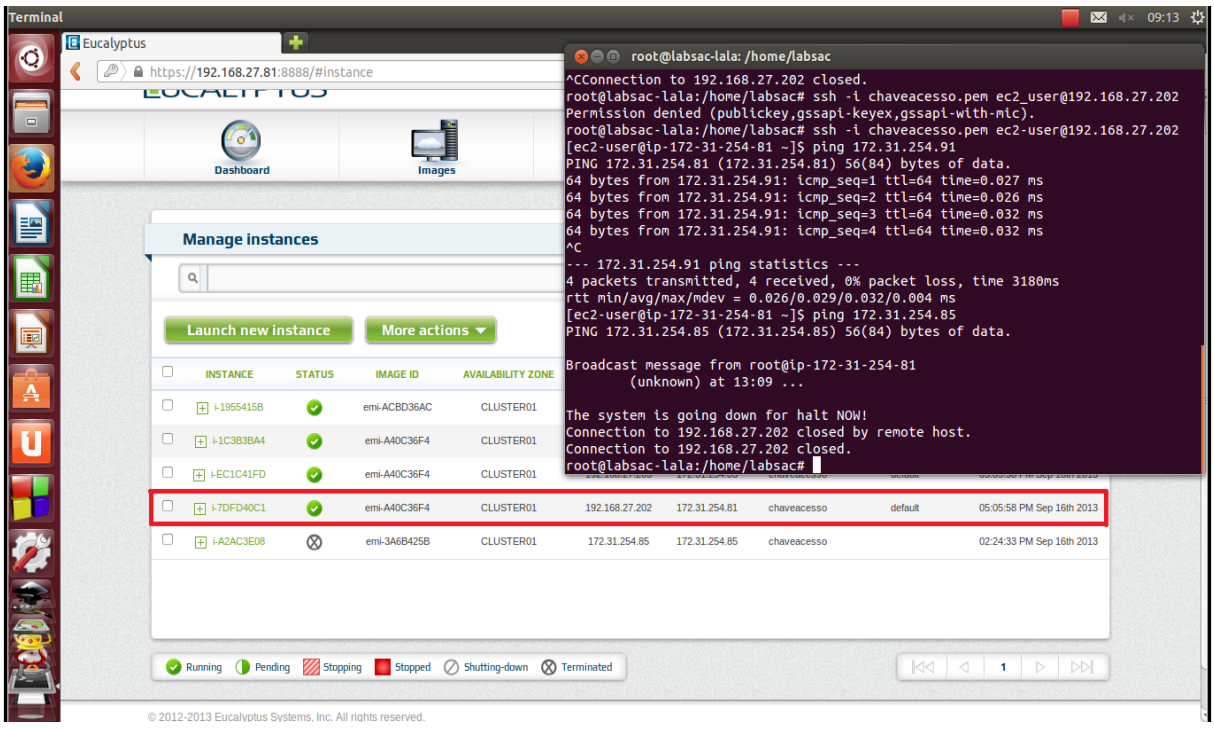

Figura 4.10: Instância interagindo com o *honeypot*.

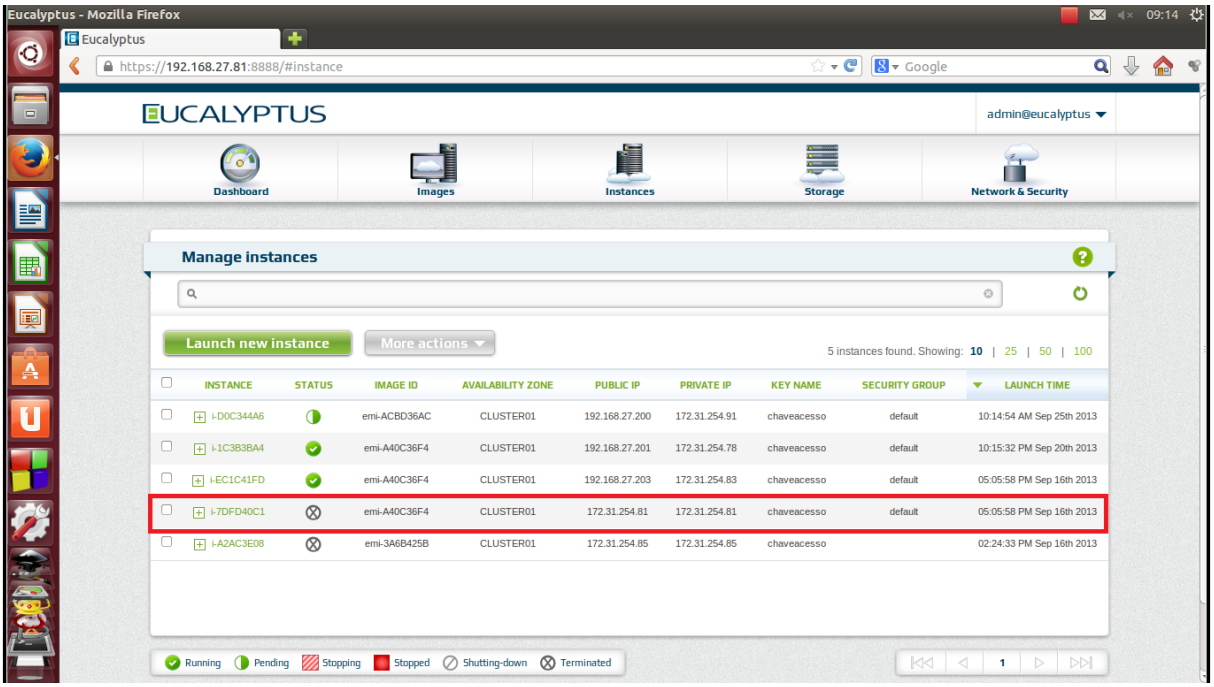

Figura 4.11: Intância finalizada.

E assim finaliza com êxito o primeiro ciclo autonômico que é responsável por manter a IaaS segura com base em interações sofridas com o *honeypot.* A instância que interagiu com o *honeypot* é finalizada e entra em uma lista de suspeitos, caso ultrapasse o número de ocorrências limites, que vai depender da política de segurança adota, esta instância é considerada comprometida e fica bloqueada, sem poder ser executada ou acessada pelos usuários da nuvem.

4.3.2 Autocura da instância *honeypot* (Segundo Ciclo Autonômico)

O segundo ciclo autonômico tem como função o retorno das características iniciais do *honeypot,* uma vez que todo *honeypot,* após sofrer alguma interação, é considerado comprometido [10], [8] e [5]. Para isso, foi utilizada a função de *Snapshot* da nuvem que não permite que sejam salvas alterações na configuração de uma imagem nela contida. Assim, após a execução, tudo que foi realizado com a instância é apagado e uma instância idêntica à inicial é garantida para a próxima vez que o usuário for usar.

Como gatilho para a recupeção do sistema, foi utilizado o método de monitorar o arquivo de suspeitos e o de regras. Estes arquivos são arterados no primeiro ciclo autonômico, sinalizado assim que o *honeypot* sofreu alguma interação então deve ser recupedar para que o sistema retorne sempre a um estado estável e sem risco de comprometimento.

O segundo ciclo autonômico começa quando o sensor do *framework,* representado pela classe *sensorFW*, monitora o crescimento do arquivo de regras ou arquivo de suspeitos. Quando uma nova regra é gerada, sinaliza para ele que o *honeypot* sofreu interação e dever ser finalizado para que nova instância *honeypot* possa ser executada. O IP do *honeypot* é localizado na lista de liberados (*WhiteList*), neste arquivo estão o IP dos *honeypots* e o da controladora.

Após coletar todas essas informações, o sensor as manda para o MonitorFW ele vai buscar o EMI e ID a partir do método "*executeFilter*" e enviar para o *AnalyzerFW*. Em posse do ID ele gera um "*terminate*" para finalizar a execução da instância e em posse do EMI gera um "*run*" para iniciar um nova instância o *honeypot.* A figura 4.12 mostra as "*atuadorTerminate"* e a figura 4.13 um "*atuadorRun*" criado.

```
cdcd credentials/admin && source eucarclseuca-terminate-instances i-D0c344A6
```

Figura 4.12: Código gerado no ***"****atuadorTerminate".*

```
cdcd credentials/admin && source eucarclseuca-run-instances -k chaveacesso -t m1.small  emi-ACBD36Ac
```

Figura 4.13: Código gerado no ***"****atuadorRun*".

A Figura 4.14 mostra o honeypot iniciando no *Dashboard* (ferramenta utilizada para monitoramento e administração da nuvem de forma gráfica via navegador *web*), evidenciando o fim ciclo autonômico.

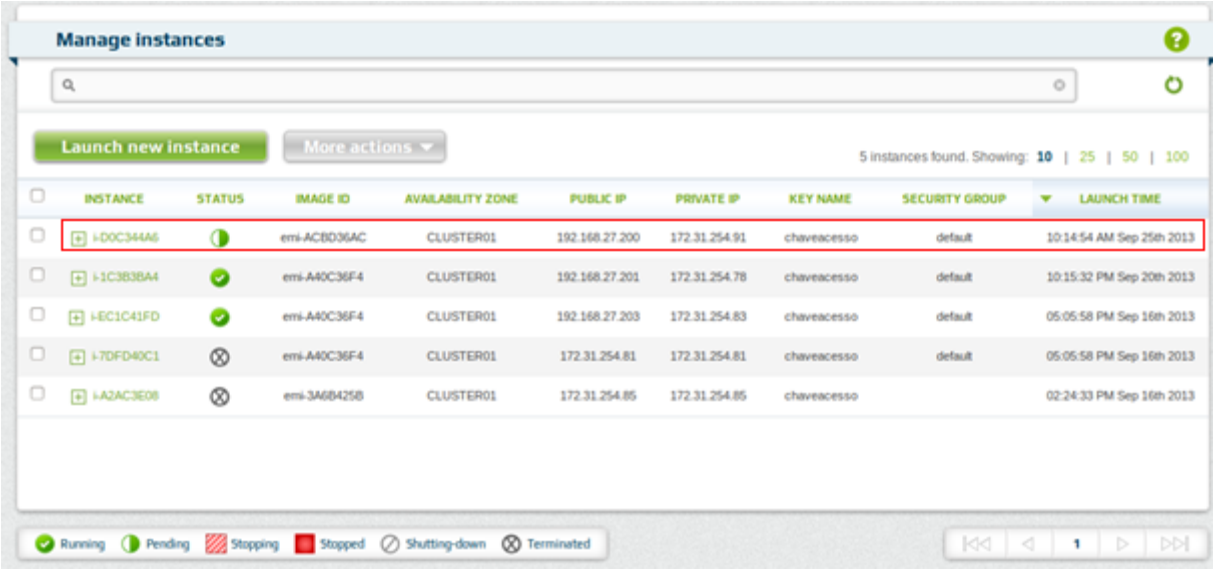

Figura 4.14: honeypot iniciando.

O Quadro 3 mostra uma comparação com todos modelos dos trabalhos relacionados e o AutonomicSec-Cloud. O AutomonicSec-Coud é uma ferramenta que tabalha suas regras de segurança elas para compor listas de controle de acesso (ACL) e Política de Grupo, aumentando a segurança em ambientes de computação em nuvem. Com o framework instalado na coltroladora da nuvem vários *honeypot* podem ser utilizados de mesmo tipo ou não, cabendo ao administrador da nuvem a configuração e disponibilização deles para os clientes da nuvem como SaaS (*Software* como serviço) na forma de HaaS (*Honeypot* como serviço).

Trabalhos	Cliclo Autonômicos	Tipo de Sistema anexo	Tipo Infraestrutura	Tipo de Serviço oferecido
[5]	2	IDS e ACL's	Nuvem	-
[6]	1	IDS e Snort	Rede Tradicional	-
[7]	1	IDS e Firewall	Rede Tradicional	-
[8]	1	IDS e Firewall	Rede Tradicional	-
[9]	1	Apenas capitação	Nuvem	HaaS

		de logs		
[10]	2	ACL e Firewall	Rede Tradicional	-
Autonomic-Cloud	2	ACL e Política de Grupo	Nuvem	HaaS

Quadro 3: Comparação entre os trabalhos relacionados e o AutonimicSec-Cloud.

4.5 Considerações Finais

Neste capítulo foi apresentado o cenário de teste, onde toda a infraestrutura usada para implementação protótipo do modelo proposto. Foi também exposto a implantação da nuvem EUCALYPTUS e suas características que justificam seu uso no protótipo criado.

Os testes realizados tiveram como foco os dois ciclos autonômicos, o primeiro para auxiliar a segurança da nuvem inserindo regras para finalizar instâncias suspeitas e o segundo para garantir a integridade do sistema e do *honeypot*. Tendo êxito na realização dos teste, o modelo proposto é contribui para a computação em nuvem uma vez que atribui características autonômicas na administração e segurança da mesma.

5. CONCLUSÃO

A computação em nuvem é uma tendência no mercado, deixando de ser de ser promessa passando ser realidade. Trabalhos que contribuem para melhorar a segurança bem como disponibilize serviços que possam ser tarifados são de extrema importância.

Neste trabalho foram apresentadas definições de computação em nuvem, modelos de implementação, características desejáveis, modelos de prestação de serviço, os diferentes papeis dentro da prestação de serviço, os desafios e a cenário atual da segurança para computação em nuvem. Formam abordados conceitos de computação autonômica assim como sua utilização na segurança de redes e conceitos de metodologia de decepção com o uso de honeypot.

Nos trabalhos relacionados foram incluídos trabalhos na área de segurança autonômica e segurança autonômica para nuvem computacionais, a fim de embasar teoricamente todo o trabalho proposto assim como facilitar o entendimento do mesmo.

O trabalho teve como proposta é uma contribuição na área de segurança para computação em nuvem. Trouxe conceitos de computação autonâmica, o que contribui para diminuir a interação humana no processo de defesa da nuvem.

Tendo em vista o cenário exposto, as contribuições deste trabalho são:

- Criação de um modelo de segurança autonômica para computação em nuvem baseada nos *logs* gerados por *honeypots* para criação de regras de acesso a nuvem;
- Extensão do framework AutonomicSec para ser usado em ambiente de computação em nuvem.

Trabalhos futuros

Como o campo da computação em nuvem é muito amplo alguns trabalhos futuros poderiam contribuir para o melhorar o modelo proposto e preencheriam lacunas deixadas por essa abordagem, são elas:

- Criação de um IDS que trabalhe com as regras criadas pelos *honeypots*;

- Criação de regras para construção de grupos de monitoramento com base nas interações com o *honeypot*;
- Uso do *honeypot* para criação de assinaturas de ataques;
- Implementação de um *honeypot* de alta interação na nuvem.
- A implementação de mais ciclos autonômicos.

REFERÊNCIAS

[1] SOUSA Flávio R. C., MOREIRA Leonardo O., MACHADO Javam C. "*Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios"*, ERCEMAPI, 2009.

[2] ZHOU, Minqi; ZHANG Rong;XIE, Wei; QIAN,Weining; ZHOU, Aoying. "*Security and Privacy in Cloud Computing: A Survey",* Sixth International Conference on Semantics, Knowledge and Grids, 2010.

[3] PEDROSA, Paulo H. C., NOGUEIRA Tiago. "*Computação em Nuvem"*, 2011.

[4] S. Ramgovind, M. Eloff M, E. Smith. "*The Management of Security in Cloud Computing".*School of Computing, University of South Africa, Pretoria, South Africa, 2010.

[5] JACOB, Benoit. "*Automatic XSS detection and Snort signatures/ACLs generation by the means of a cloud-based honeypot system"*, Edinburgh Napier University, 2011.

[6] XINYU, Tang. "*The Generation of Attack Signatures Based on Virtual Honeypots"*, 2010.

[7] BO, Wang; PING, Zhu; QIAOYAN,Wen; XIAOJUN,Yu. "*A Honeynet-Based Firewall Scheme With Initiative Security Strategies"*, State Key Laboratory of Networking and Switching, School of Science, 2009.

[8] TIAN, Jun-feng; WANG, Jian-ling; LI, Ren-ling; YANG, Xiao-hui. "*A Study of Intrusion Signature Based on Honeypot", Proceedings of the Sixth International Conference on Parallel and Distributed Computing*, Applications and Technologies,2005.

[9]BALAMURUGAN, M; POORNIMA, B Sri Chitra. "*Honeypot as a Service in Cloud"*, International Conference on Web Services Computing (ICWSC),2011.

[10] Teles, Ariel Soares. “AutonomicSec: Um Mecanismo Autonômico para Segurança de Redes baseado em Decepção”, Programa de Pós-Graduação em Engenharia de Eletricidade – UFMA, 2012.

[11] HARIRI, Salim; KHARGHARIA, Bithika; CHEN, Houping; YANG, Jingmei and ZHANG, Yeliang. “The Autonomic Computing Paradigm”, Springer Science + Business Media, Inc. Manufactured in The United States, 2006.

[12] KEPHART, Jeffrey O.; CHESS,David M. “The Vision of Autonomic Computing”, IEEE Computer Society, 2003.

[13] CORRÊA, Sand; CERQUEIRA, Renato. “Computação Autônoma: Conceitos, Infra-estruturas e Soluções em Sistemas Distribuídos”, 27º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos,2009.

[14] MAURERA, Michael; BRANDICA, Ivona; SAKELLARIOU, Rizos. “Adaptive resource configuration for Cloud infrastructure management”, Future Generation Computer Systems, Elsevier, 2012.

[15] CHAMPRASERT, Paskorn; SUZUKI, Junichi and LEE, Chonho. “Exploring self-optimization and self-stabilization properties in bio-inspired autonomic cloud applications”, Wiley Online Library, 2012.

[16] AMORETTI, Michele; ZANICHELLI, Francesco; CONTE, Gianni. “Efficient autonomic cloud computing using online discrete event simulation”, Journal Parallel Distributed Computing, 2013.

[17] ACETO, Giuseppe; BOTTA, Alessio; DONATO, Walter; PESCAPÈ, Antonio. “Cloud monitoring: A survey”, Computer Networks, 2013.

[18] KUMAR, Upendra; MISHRA, Bimal Kumar; SAHOO, G. "Defending Polymorphic Worms in Computer network using Honeynet”, International Journal of Engineering Science and Technology, 2012, Vol.4(4), p.1408.

[19] HORN, Paul; “Autonomic computing: IBMs perspective on the state of information technology”, 2001.

[20] FURTADO, Fabiana C. F.; LIMA, Pablo de Oliveira. “Computação Nas *Nuvens E Sua Aplicação No Gerenciamento De Projetos Usando ZohoProjects*”; Instituto Paraibano De Pós-Graduação-I2P,2010.

[21] KURP, Patrick. “*Green computing*”; Communications of The ACM, Vol. 53, 2010.

[22] RITTINGHOUSE, John W.; RANSOME, James F. “*Cloud Computing: Implementation, Management and Security*”, CRC Press Taylor & Francis Group, 2010.

[23] PEDROSA, Paulo H. C., NOGUEIRA Tiago. “ComputaçãoemNuvem”, 2011.

[24] MELL, Peter & GRACE, Tim: The NIST Definition of Cloud Computing. National Institute of Standards and Technology, 2011.Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf/>.

[25] LI, Xiangdong. “Cloud Computing: Introduction, Application and Security from Industry Perspectives”, IJCSNS, Vol.11 No.5, 2011.

[26] JR Arlindo Marcon, LAUREANO Marcos, SANTIN Altair, MAZIERO Carlos ; “Aspectos de Segurança e Privacidade em Ambientes de Computação em Nuvem”; X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais; 2011;

[27] ZHOU, Minqi; ZHANG, Rong; ZENG, Dadan; QIAN, Weining. “Services in the Cloud Computing Era: A Survey”; IUCS;2010.

[28]RITTINGHOUSE,John W.; RANSOME, James F.“Cloud Computing: Implementation, Management and Security”;Taylor & Francis Group, EstadosUnidos, 2010.

[29] ARMBRUST, Michael; FOX, Armando; GRIFFITH, Rean; JOSEPH, Anthony D.; KATZ, Randy; KONWINSKI, Andy; LEE, Gunho; PATTERSON, David; RABKIN, Ariel; STOICA, Ionand; ZAHARIA ,Matei. “A View of Cloud Computing”; Communications of the ACM, Vol. 53, 2010.

[30] NAYAK, Smitha and YASSIR, Ammar. “Cloud Computing As an Emerging Paradigm”, International Journal of Computer Science and Network Security, Vol.12, 2012.

[31] DOKRAS, Satchit; HARTMAN, Bret; MATHERS, Tim; FITZGERALD, Brian; CURRY, Sam; NYSTROM, Magnus; BAIZE, Eric; MEHTA, Nirav. “O Papel Da Segurança Na Computação Em Nuvem Confiável”, White paper RSA, The Security Division of EMC, 2009;

[32] Coordenação Estratégica de Tecnologia (CETEC). “Segurança de Computação em Nuvem”, III CONGRESSO INTERNACIONAL SOFTWARE LIVRE E GOVERNO ELETRÔNICO,2010.

[33] MATHER, Tim; KUMARASWAMY, Subra; and LATIF, Shahed. “Cloud Security and Privacy”, O’Reilly, EUA, 2009;

[34] CHESS, D. M; PALMER, C. C; WHITE, S. R. “Security In An Autonomic Computing Environment”, IBM Systems Journal, 2003.

[35] QASSRAWI, Mahmoud T.; ZHANG, Hongli. “Deception Methodology in Virtuual Honeypots. Second International Conference on Networks Security”, Wireless Communications and Trusted Computing, 2010.

[36] STALLINGS, W. “Criptografia e segurança de redes: Princípios e práticas”. Prentice Hall, 4 edition, 2008.

[37] PROVOS, N., AND HOLZ, T. “Virtual Honeypots: From Botnet Tracking To Intrusion Detection”, four ed. Addison-Wesley Professional, 2010.

[38] SPITZNER, L. “Honeypots: Definitions and value of honeypots”. In SANS Annual Conference, 2002.

[39] IBM. “An architectural blueprint for autonomic computing”, 2003.

[40] VIRTI, Emerson Salvador. “Implementação de um IDS Utilizando SNMP e Lógica Difusa”, Programa de Pós-Graduação-UFRS, 2007.

[41] BACE, R., MELL, P. (2001). “Intrusion Detection Systems. NIST – National Institute of Standards and Technology”. Disponível em: <http://www.snort.org/docs/nist-ids.pdf/>.

[42] ALLEN, J., CHRISTIE, A., FITHEN, W., MCHUGH, J., PICKEL, J., and E. “State of the practice of intrusion detection technology”. Software Engineering Institute: Carnegie Mellon University, 1999.

[43] CLINCY, Victor A.; ABU-HALAWEH, Nael .“A Taxonomy of free Network Sniffers for teaching and research”, Journal of Computing Sciences in Colleges, 2005.

[45] CHAMALES, George. “Know Your Enemy: Honeywall Cdrom”, Journal IEEE Computer Society, 2004.

[46] HARIRI, S.; KHARGHARIA, B.; CHEN, H.; YANG, J.; ZHANG, Y.; PARASHAR, M.; and LIU,H. “The autonomic computing paradigm”. Cluster Computing, 2006.

[47] HUEBSCHER, M. C. and MCCANN,J. A. “A survey of autonomic computing - degrees, models, and applications”. ACM Comput. Surv, 2008.

[48] 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.

[49] NURMI, Daniel; WOLSKI, Rich; GRZEGORCZYK, Chris; OBERTELLI, Graziano; SOMAN, Sunil; YOUSEFF, Lamia; ZAGORODNOV, Dmitrii. “The Eucalyptus Open-source Cloud-computing System”, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009.