UNIVERSIDADE DO ESTADO DE SANTA CATARINA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO BASTOS RODRIGUES

ANÁLISE DE SEGURANÇA DA PLATAFORMA DE NUVEM COMPUTACIONAL OPENSTACK GRIZZLY

JOINVILLE

NOVEMBRO, 2013

BRUNO BASTOS RODRIGUES

ANÁLISE DE SEGURANÇA DA PLATAFORMA DE NUVEM COMPUTACIONAL OPENSTACK GRIZZLY

Monografia apresentada ao curso de Ciência da Computação da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do título de Bacharelado, orientado pelo Prof. Dr. Charles Christian Miers.

JOINVILLE

NOVEMBRO, 2013

ANÁLISE DE SEGURANÇA DA PLATAFORMA DE NUVEM COMPUTACIONAL OPENSTACK FOLSOM

BRUNO BASTOS RODRIGUES

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação:

Banca Examinadora:

_____

Professor Dr. Charles Christian Miers (Orientador)

_____

Professor Dr. Guilherme Piegas Koslovski

_____

Professor Dr. Rafael Rodrigues Obelheiro

LISTA DE FIGURAS

Figura 1 - Características essenciais NIST.....17

Figura 2 - Configurações de Serviço da Nuvem.....23

Figura 3 - Negociação para Estabelecer SLA.....25

Figura 4 - Tipos de Nuvem.....32

Figura 5 - Arquitetura de Referência do NIST..... 34

Figura 6 - Interação Entre os Atores.....38

Figura 7 - Orquestração de Serviços.....39

Figura 8 - Gerenciamento de Serviços.....41

Figura 9 - Microsoft Azure.....54

Figura 10 - Arquitetura Conceitual OpenStack Grizzly.....79

Figura 11 - Arquitetura Lógica OpenStack Grizzly.....81

Figura 12 - Diagrama 1 de Componentes do Keystone.....78

Figura 13 - Diagrama 2 de Componentes do Keystone.....79

Figura 14 - Diagrama de Requisição de Criação de uma Máquina Virtual.....80

Figura 15 – Ilustração do Capítulo 2. 96

Figura 17 – Questões de Segurança na Nuvem Elencadas pelo NIST. 102

Figura 18 - Questões de Segurança na Nuvem Elencadas pela CSA. 110

Figura 19 - Questões de Segurança na Nuvem Elencadas pela ENISA. 116

Figura 20 – Principais Pontos de Segurança nas VM's..... 144

Figura 23 – Suporte dos Hypervisors no OpenStack 153

Figura 26 - Arquitetura do Xen 158

Figura 27 – Arquitetura do Xen no OpenStack..... 160

Figura 28 – Arquitetura VMWare ESX Server. 162

Figura 29 - Interface do *LaunchPad*. 164

Figura 30 – Bugs do Qemu/KVM por Importância..... 167

Figura 31 – Bugs do Qemu/KVM por Status 168
Figura 32 – Vulnerabilidades Qemu/KVM 169
Figura 33 – Bugs do Xen por Importância 171
Figura 34 – Bugs do Xen por Status 172
Figura 35 – Vulnerabilidades Xen 173
Figura 36 – Bugs do VMWare por Importância 175
Figura 37 – Bugs do VMWare por Status 176
Figura 38 – Vulnerabilidades VMWare 177
Figura 38 – Citações em Trabalhos de Preocupações x Soluções de Segurança na Computação em Nuvem 179
Figura 38 - Taxionomia de Vulnerabilidades na Virtualização 181
Figura 39 – Instalação Arquivos DevStack 185
Figura 40 – Arquivo localrc do Nó do Cluster Controller 186
Figura 41 – Arquivo localrc do Nó Compute 187
Figura 42 – Execução do Script Stack.sh 187
Figura 43 – Instalação Bem Sucedida 188
Figura 44 – Topologia da Rede 189
Figura 45 – Upload de Arquivo para Procedimento de Exclusão 192
Figura 46 – Teste de Recuperação de Dados 192
Figura 47 – Trigger para Exclusão de Objetos e Containers 193
Figura 48 – Blueprint da Cifragem de Blocos 195
Figura 49 – Algoritmo de Hash SHA1 197
Figura 50 – Domínios de Segurança do OpenStack 199
Figura 52 – Separação de Serviços para Tokens não Assinadas 203
Figura 54 – Facilitar Autenticação nas API's 206
Figura 55 – Pseudo-algoritmo para Criação da VM 207
Figura 56 – Adição de Regras no Grupo de Segurança 208

Figura 57 – Criação de Chaves 209
Figura 58 – Verificação das Imagens no Nova 210
Figura 59 – Criação da VM 211
Figura 61 – Verificação dos IP's Flutuantes 212
Figura 62 – Filtros do Wireshark 214
Figura 63 – Pacote de Autenticação Keystone - Glance 215
Figura 64 – Pacote de Autenticação Glance - Keystone 215
Figura 65 – Senha em Texto Puro, Wireshark 217
Figura 67 – Confirmação da Cifragem da Imagem 219
Figura 68 – Configuração da API Keystone 220
Figura 69 – Geração dos Certificados SSL 221
Figura 70 – Autenticação no Keystone 222
Figura 71 – Comunicação Cifrada com o Keystone 223
Figura 72 – Comunicação com SSL Certificada 224
Figura 73 - Isolamento de Processos Fornecido pelo sVirt. 226
Figura 74 – Rótulo das VM's Criado pelo sVirt 227
Figura 75 – Operações Básicas com as VM's Utilizando Virsh 229
Figura 76 – Cronograma Previsto x Realizado TCC-I 227
Figura 77 – Cronograma Proposto x Realizado TCC-II 228
Figura 78 - Tipos de Hypervisors 254
Figura 79 – Chaves Simétricas 255
Figura 80 - Chaves Assimétricas 256

LISTA DE TABELAS

Tabela 1 - Comparação entre Abordagens de Implantação.....30

Tabela 2 - Comparativo entre Arquiteturas de Referência.....35

Tabela 3 - Características S3.....46

Tabela 4 - Características EC2.....47

Tabela 5 - Características CloudStack.....48

Tabela 6 - Características OpenNebula.....50

Tabela 7 - Características OpenStack.....51

Tabela 8 - Características Google App Engine.....53

Tabela 9 - Características Microsoft Azure.....55

Tabela 10 - Características Salesforce.com.....56

Tabela 11 - Características Drupal.....58

Tabela 12 - Comparativo Geral entre as Plataformas.....59

Tabela 13 - Componentes do OpenStack.....65

Tabela 14 - Principais Pontos de Segurança Elencados pelo NIST.....86

Tabela 15 - Recomendações de Segurança do NIST.....90

Tabela 16 - Principais Pontos de Segurança Elencados pela CSA.....92

Tabela 17 - Recomendações de Segurança CSA.....97

Tabela 18 - Principais Pontos de Segurança Elencados pela ENISA.....100

Tabela 19 - Comparativo das Questões Políticas, Legais e Organizacionais 104

Tabela 20 - Comparativo de Questões Técnicas.....105

Tabela 21 – Estrutura do Documento de Segurança em Virtualização do NIST.....119

Tabela 22 – Estrutura do Documento de Segurança em Virtualização da Red Hat.....126

Tabela 23 – Estrutura do Documento de Segurança em Virtualização da PCI. 130

Tabela 24 – Comparativo NIST, RED HAT e PCI.....135

Tabela 25 – Ciclo de Vida da VM nos Hypervisors suportados pelo OpenStack 142

Tabela 26 – Características de Rede nos Hypervisors Suportados pelo OpenStack.....143

Tabela 27 – Características de Monitoramento nos Hypervisors suportados pelo OpenStack.....144

Tabela 28 – Características Técnicas nos Hypervisors suportados pelo OpenStack.....145

Tabela 29 – Características de Segurança nos Hypervisors suportados pelo OpenStack.....146

LISTA DE ABREVIATURAS E SIGLAS

ACL *Access Control List*

API *Application Programming Interface*

AWS *Amazon Web Services*

CLI *Command Line Interface*

CSA *Cloud Security Alliance*

CSMIC *Cloud Service Measurement Index Consortium*

CRM *Customer Relationship Management*

DAEMON *Disk And Execution MONitor*

DOS *Denial of Service*

DDOS *Distributed Denial of Service*

DRDB *Distributed Replicated Block Device*

EGTI Estratégia Geral de Tecnologia da Informação

ENISA *European Network and Information Security Agency*

GSIPR Grupo de Segurança da Informação da Presidência da República

HTTP *Hypertext Transfer Protocol*

IAAS *Infrastructure as a Service*

IN4 Instrução Normativa Número Quatro

IP *Internet Protocol*

ISO *International Organization for Standardization*

JVM *Java Virtual Machine*

JSON *JavaScript Object Notation*

KPI *Key Performace Indicators*

LDAP *Lightweight Directory Access Protocol*

LVM *Logical Volume Manager*

NIST *National Institute of Standards and Technology*

OCCI *Open Cloud Computing Interface*

OSSG *OpenStack Security Group*

PAAS *Platform as a Service*

PCSTI Processo de Contratação de Serviços de Tecnologia da Informação

PDA *Personal Digital Assistant*

PDTI Plano Diretor de Tecnologia da Informação

QOS *Quality of Service*

RBAC *Role Based Access Control*

REST *Representational State Transfer*

SAAS *Software as a Service*

SAML *Security Assertion Markup Language*

SLA *Service Level Agreement*

SOAP *Simple Object Access Protocol*

SISP Sistema de Administração de Recursos de Tecnologia da Informação

SSH *Secure Shell*

TI Tecnologia de Informação

URL *Uniform Resource Locator*

VM *Virtual Machine*

VMT *Vulnerability Management Team*

VPN *Virtual Private Network*

XACML *eXtensible Access Control Markup Language*

XML *eXtensible Markup Language*

ÍNDICE

1 INTRODUÇÃO.....17

1.1 OBJETIVOS GERAIS E ESPECÍFICOS.....18

1.2 ORGANIZAÇÃO DO TRABALHO.....19

1.3 MÉTODO DE EXECUÇÃO DO TRABALHO 21

2 EMBASAMENTO TEÓRICO 22

2.1 CONCEITOS DE COMPUTAÇÃO EM NUVEM 22

2.1.1 DEFINIÇÃO E CARACTERÍSTICAS 23

2.1.1 VIRTUALIZAÇÃO DE RECURSOS..... 25

2.1.2 MODELOS DE SERVIÇO 26

2.1.3 MEDIÇÃO DOS SERVIÇOS DA NUVEM..... 30

2.1.4 ABORDAGENS DE IMPLANTAÇÃO 34

2.1.5 ARQUITETURA DE REFERÊNCIA..... 38

2.1.5.1 ORQUESTRAÇÃO DE SERVIÇOS..... 44

2.2 PLATAFORMAS DE COMPUTAÇÃO EM NUVEM 48

2.2.1 AMAZON WEB SERVICES (AWS) 46

2.2.1.1 AMAZON S3..... 46

2.2.1.2 AMAZON EC2 47

2.2.2 CLOUDSTACK..... 49

2.2.3 OPENNEBULA..... 50

2.2.4 OPENSTACK 51

2.2.5 GOOGLE APP ENGINE 53

2.2.6 MICROSOFT AZURE..... 54

2.2.7 SALESFORCE.COM 58

2.2.8 DRUPAL..... 62

2.2.9 COMPARATIVO ENTRE AS PLATAFORMAS 63

2.2.10 ARQUITETURA OPENSTACK GRIZZLY 64
2.2.10.1 FOLSOM VS GRIZZLY 65
2.2.10.2 OPENSTACK NOVA 80
2.2.10.3 OPENSTACK GLANCE 82
2.2.10.4 OPENSTACK SWIFT 83
2.2.10.5 OPENSTACK HORIZON 84
2.2.10.6 OPENSTACK NEUTRON (QUANTUM NO FOLSOM) 85
2.2.10.7 OPENSTACK CINDER 86
2.2.10.8 OPENSTACK KEYSTONE 87
2.2.11 CONSIDERAÇÕES PARCIAIS 90
3 ANÁLISE DE SEGURANÇA 93
3.1 SEGURANÇA EM COMPUTAÇÃO EM NUVEM 96
3.1.1 GUIA DE SEGURANÇA DO NIST 97
3.1.2 GUIA DE SEGURANÇA CSA 103
3.1.3 GUIA DE SEGURANÇA ENISA 111
3.1.4 COMPARATIVO NIST, CSA E ENISA 117
3.1.5 CONSIDERAÇÕES SOBRE O COMPARATIVO 123
3.2 SEGURANÇA EM VIRTUALIZAÇÃO 124
3.2.1 GUIA DE VIRTUALIZAÇÃO DO NIST 125
3.2.2 GUIA DE VIRTUALIZAÇÃO RED HAT 131
3.2.3 GUIA DE SEGURANÇA EM VIRTUALIZAÇÃO DA PCI SECURITY STANDARDS COUNCIL 135
3.2.4 COMPARATIVO NIST, RED HAT E PCI 141
3.2.5 CONSIDERAÇÕES SOBRE O COMPARATIVO 143
3.2.5.1 PONTOS RELEVANTES AO TRABALHO ABORDADOS NO COMPARATIVO 145
3.2.5.2 ISOLAMENTO 145

3.2.5.3 VISÃO GERAL DOS *HYPERVISORS* SUPORTADOS PELO OPENSTACK 146

3.3 PRINCIPAIS *HYPERVISORS* SUPORTADOS PELO OPENSTACK GRIZZLY (qemu/KVM, Xen, VMWare) 155

3.3.1 QEMU/KVM EM X86 155

3.3.2 XEN 158

3.3.3 VMWARE 161

3.3.4 VISÃO GERAL DOS BUGS E VULNERABILIDADES PARA O QEMU/KVM, XEN E VMWARE 163

3.3.4.1 CRITÉRIOS LAUNCHPAD E CVE (*Common Vulnerabilities and Exposures*) 163

3.3.4.2 QEMU/KVM 167

3.3.4.3 XEN 170

3.3.4.4 VMWARE 174

3.3.4.5 CONSIDERAÇÕES SOBRE OS BUGS E VULNERABILIDADES 178

3.4 PROPOSTA DE UM FRAMEWORK PARA CLASSIFICAÇÃO DE VULNERABILIDADES RELACIONADAS Ã VIRTUALIZAÇÃO 180

3.5 CONSIDERACOES PARCIAIS 183

4 ASPECTOS DE SEGURANÇA NO OPENSTACK GRIZZLY 184

4.1 INSTALAÇÃO DO OPENSTACK COM O DEVSTACK MULTI-NODE 184

4.2 REVISÃO DOS PROBLEMAS ELENCADOS NO TCC-I 189

4.2.1 BACKUP, RECUPERAÇÃO E EXCLUSÃO 190

4.2.2 CIFRAGEM DOS DADOS 194

4.2.3 ARMAZENAMENTO DE SENHAS 196

4.2.4 ARQUITETURA CENTRALIZADA DE CONTROLE DE ACESSO 198

4.2.5 ARQUITETURA PARA O USO INDEVIDO DE TOKENS 200

4.3 CRIAÇÃO DE UMA VM VIA API 206

4.4 MONITORAMENTO DA CRIAÇÃO DA VM COM O WIRESHARK 213

4.5 ASPECTOS DE SEGURANÇA DO QEMU/KVM. 225

4.5.1 ISOLAMENTO E GERENCIAMENTO DAS VM'S 225

4.6 CONSIDERAÇÕES PARCIAIS 230

5. CONSIDERAÇÕES FINAIS 232

5.1 PRINCIPAIS DIFICULDADES.....237

5.2 CONTRIBUIÇOES.....238

5.3 TRABALHOS FUTUROS.....239

REFERÊNCIAS..... 241

APÊNDICE A..... 253

APÊNDICE B..... 254

RESUMO

A computação em nuvem é uma tecnologia relativamente recente com grande potencial a ser explorado no mercado da tecnologia de informação. Contudo, existem diversos desafios que devem ser solucionados para que este potencial se torne realidade. Os desafios de segurança normalmente são os mais importantes e desafiadores, pois impedem várias empresas de migrarem seus serviços para plataformas de computação em nuvem. Este trabalho apresenta uma análise e resolução de problemas de segurança da plataforma OpenStack Grizzly, apresentando os conceitos básicos pertinentes a computação em nuvem, a arquitetura da plataforma OpenStack Grizzly e preocupações de segurança encontradas na pesquisa bibliográfica. A análise em si é constituída de uma avaliação referencial dos principais pontos críticos verificados na pesquisa bibliográfica, elencando problemas que são averiguados na parte prática, que consiste na implementação de melhoria de recursos de segurança à plataforma. A análise focou em seis problemas básicos (1 - *backup,* recuperação e exclusão; 2 - cifragem de bloco e volumes; 3 - segurança no armazenamento de senhas; 4- arquitetura de controle de acesso e políticas centralizadas; 5 – uso indevido de tokens; e 6 – segurança das imagens), além de aspectos relacionados à segurança da VM e comunicação de rede. Os resultados da análise identificaram que parte dos problemas podem ser resolvidos utilizando recursos do OpenStack que não são habilitados por padrão e outros que são configuráveis.

Palavras-chave: Segurança, Vulnerabilidade, Virtualização, OpenStack Grizzly, Computação em Nuvem.

ABSTRACT

The Cloud Computing is a relative new technology with a great potential to be explored in the market of information of technology, but there are several challenges that must be solved to fulfill this potential. The security issues are the most important and challenging, because they prevent several companies to migrate they services to a Cloud platform. This paper presents a security analysis of the OpenStack Grizzly platform, presenting the basics concepts of Cloud Computing, the architecture of OpenStack Grizzly and the security concerns founded in the bibliographic research. The analysis itself will be made of a referential evaluation of the main critical points verified in the bibliographic research, checking if it is corrected or still insecure, presenting the results obtained, so they can be investigated in the practical part, which consists of an implementation security features to the platform. The analysis focused in six basics problems (1 – backup, recovery and deletion; 2 – block and volume cipher; 3 – secure password storage; 4 – centralized architecture of access control and politics; 5 – misuse of tokens; and 6 – image security), and other aspects related to the VM security and network communication. The analisys results revealed that some problems can be solved using OpenStack own resources, that are not enabled by default and others configurable.

Keywords: *Security, Vulnerabilities, Virtualization, OpenStack Grizzly, Cloud Computing*

1 INTRODUÇÃO

No dia a dia os serviços básicos como água, eletricidade e telefone, são cobrados conforme sua utilização pelo consumidor, ou seja, paga-se um preço conforme as demandas pessoais ou empresariais de quem utiliza estes serviços. A computação em nuvem segue o mesmo conceito, fazendo com que o usuário pague pelos serviços de computação que ele utiliza. Este dinamismo oferecido pela computação em nuvem está proporcionando um franco crescimento ao modelo, fazendo com que seja amplamente adotado por empresas e organizações a fim de reduzir custos proporcionados pelo modelo tradicional de *data center*. Por outro lado, este crescimento também expõe algumas das carências que o modelo possui, dentre elas a da segurança, que segundo (SLIPETSKYY, 2011) é um dos principais fatores para a adoção de soluções em nuvem.

A segurança também é um fator que difere de acordo com as abordagens de implantação e modelos de serviço utilizados na nuvem, estes fazem com que o nível de segurança seja maior ou menor dependendo da escolha do consumidor. A escolha adequada da abordagem de implantação e do modelo de serviço que será utilizado é muitas vezes uma tarefa complexa, sendo que para cada modelo de serviço existem diversas plataformas de serviço disponíveis (MELL e GRANCE, 2009). Determinar qual abordagem de implantação, modelo de serviço e a plataforma adequada é frequentemente uma tarefa que necessita de um serviço de auditoria especializado para auxiliar na tarefa de migração para o modelo de nuvem.

Dentre as plataformas de infraestrutura como serviço (IaaS) que compõem o cenário de computação em nuvem, têm-se o projeto OpenStack como uma das principais plataformas (PANETTIERI, 2013), sendo a plataforma *open source* com mais parceiros e colaboradores, com o número de empresas e organizações envolvidas chegando a 161, entre elas grandes empresas como a HP e a IBM (OPENSTACK, 2012). Sendo a segurança um dos principais entraves para a adoção de computação em nuvem, e a plataforma OpenStack uma das mais utilizadas plataformas de IaaS, sanar as questões de

segurança da plataforma acaba se tornando uma grande necessidade para a consolidação da plataforma.

Este trabalho visa contribuir com o desenvolvimento do projeto OpenStack por meio da apresentação da resolução dos problemas de segurança básicos (1 - *backup,* recuperação e exclusão; 2 - cifragem de bloco e volumes; 3 - segurança no armazenamento de senhas; 4- arquitetura de controle de acesso e políticas centralizadas; 5 – uso indevido de *tokens*; e 6 – segurança das imagens) elencados pela análise de segurança. Em diversos pontos do trabalho foram realizadas comparações, com o objetivo de realçar a diferença entre os objetos (definições, plataformas, guias) de comparação, de modo a evidenciar a diferença entre os mesmos e distinguir qual abordagem apropriada para se seguir na execução de uma análise. Os principais comparativos apresentados são o dos guias e recomendações de segurança e privacidade (CSA, 2011), (JANSEN e GRANCE, 2011), (RYCK et al, 2011) e os guias de Virtualização (RED HAT, 2013), (SCARFONE, SOUPPAYA, HOFFMAN, 2011) e (PCI DSS, 2011). Estes comparativos elencam os principais pontos de vulnerabilidade existentes no modelo de nuvem e em virtualização, como também suas respectivas recomendações de segurança. Estes comparativos são importantes para estabelecer qual guia trata de maneira mais completa ou mais específica de uma determinada questão de segurança, provendo uma maior lucidez para a execução da análise e resolução de um ponto de segurança no OpenStack Grizzly, caso seja necessário.

Por meio da análise de segurança do OpenStack Grizzly realizada no TCC-I, foram estabelecidos os principais pontos de segurança pelos guias de recomendações. De modo a estabelecer a base para que seja realizada uma melhoria de segurança na plataforma OpenStack Grizzly. A apresentação da solução dos problemas neste trabalho, é realizada por meio de uma pesquisa aplicada, com averiguações práticas em uma abordagem da plataforma OpenStack Grizzly.

1.1 OBJETIVOS GERAIS E ESPECÍFICOS

O objetivo geral deste trabalho é indicar, por meio da análise de segurança da plataforma no TCC-I, um ou mais pontos em que seja possível realizar uma melhoria ou uma adição de recurso de segurança na plataforma no TCC-II. Para atingir o objetivo geral, diversos objetivos específicos são necessários:

- Compreensão dos conceitos básicos do modelo de computação em nuvem;
- Compreensão dos principais modelos de serviço e abordagens de implantação;
- Entendimento da arquitetura da computação em nuvem, compreendendo a definição dos papéis de cada parte que atua no modelo de nuvem;
- Assimilar a arquitetura da plataforma OpenStack Grizzly bem como a relação entre os componentes que a compõem com ênfase no Keystone;
- Compreender as questões de segurança elencadas pelos guias de segurança e privacidade;
- Realizar a análise de segurança do OpenStack Grizzly;
- Revisar ou aprofundar os conceitos envolvidos na parte prática;
- Melhorar ou adicionar recursos de segurança, implementando, testando, documentando e avaliando os resultados.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho é composto por cinco partes distintas: a introdução (capítulo **1**), o embasamento teórico (capítulo **2**), a parte de segurança em computação em nuvem (capítulo **3**), de questões específicas ao *hypervisor* (capítulo **4**) e as considerações finais (capítulo **5**). O trabalho está organizado

desta maneira para primeiramente, introduzir o que é realizado ao leitor nesta obra, em seguida apresentar os conceitos básicos de modo que seja possível conhecer os conceitos mais importantes da computação em nuvem, adequando, assim, a apresentação da parte de segurança do trabalho, que contém a análise de segurança do OpenStack Grizzly.

A primeira parte tem como objetivo realizar a introdução ao propósito do trabalho, bem como seus objetivos e a maneira como está estruturado. A segunda parte (capítulo **2**) trata os conceitos essenciais ao entendimento do funcionamento do modelo de nuvem, como também da plataforma OpenStack Folsom. A parte de segurança em computação em nuvem (capítulo **3**), descreve as questões de segurança elencadas pelos principais guias de recomendações, compreendendo também o objeto de interesse deste trabalho, a apresentação da resolução dos problemas elencados na análise realizada no TCC-I. A quarta parte (capítulo **4**) que trata de assuntos de segurança específicos do OpenStack Grizzly com uma ênfase no final ao *hypervisor*, como isolamento e gerenciamento de máquinas virtuais e a última parte (capítulo **5**), que apresenta as considerações finais deste trabalho.

1.3 MÉTODO DE EXECUÇÃO DO TRABALHO

O método utilizado para realizar este trabalho é a pesquisa referenciada e a pesquisa aplicada com base no método hipotético indutivo. A pesquisa referenciada é utilizada como referência por meio de diversos artigos, documentos, guias e teses voltadas para a computação em nuvem como também especializados em segurança no modelo de nuvem. A pesquisa aplicada é utilizada para aplicar o conhecimento obtido nas referências para propor soluções aos problemas elencados básicos (1 - *backup,* recuperação e exclusão; 2 - cifragem de bloco e volumes; 3 - segurança no armazenamento de senhas; 4- arquitetura de controle de acesso e políticas centralizadas; 5 – uso indevido de tokens; e 6 – segurança das imagens). O método hipotético indutivo é utilizado na inferência dos fatos observados nas referências para

deduzir situações, teorias ou fatos que se aplicam na pesquisa aplicada de segurança do OpenStack Grizzly.

2 EMBASAMENTO TEÓRICO

Para que seja possível realizar uma análise bem sucedida de uma plataforma de nuvem computacional, o primeiro passo é efetuar uma fundamentação adequada que introduza os principais conceitos relacionados ao tema, considerando isto, este capítulo tem o objetivo de fornecer esta fundamentação teórica que conceda o embasamento necessário à realização da análise de segurança. Primeiramente conceitualiza-se as definições básicas da nuvem, como definições principais, características, modelos de serviço, abordagens, entre outros tópicos essenciais a nuvem. Em seguida, é introduzido um modelo arquitetural que apresenta o que se pode esperar de uma plataforma de nuvem computacional, os papéis desempenhados pelas partes que atuam neste modelo, possibilitando assim, a apresentação das plataformas com um breve comparativo com o intuito de constatar as diferenças entre as mesmas. No final, o capítulo é complementado com uma apresentação mais detalhada do modelo arquitetural da plataforma OpenStack Grizzly, o objeto de estudo deste trabalho.

2.1 CONCEITOS DE COMPUTAÇÃO EM NUVEM

A computação em nuvem por ser um modelo de computação recente, ainda carece de uma padronização geral quanto suas terminologias referentes às suas características e definições, esta seção tem o objetivo de apresentar os conceitos básicos pertinentes à computação em nuvem, passando por sua definição, modelos de serviço e abordagens de implantação, com o objetivo de facilitar o desenvolvimento deste trabalho. Existem maneiras distintas em que é possível classificar os serviços oferecidos pela computação em nuvem, porém é interessante tanto aos usuários quanto aos provedores estar em consenso quanto a estas terminologias, para que fique claro aos interessados, o serviço que está sendo oferecido em cada modelo.

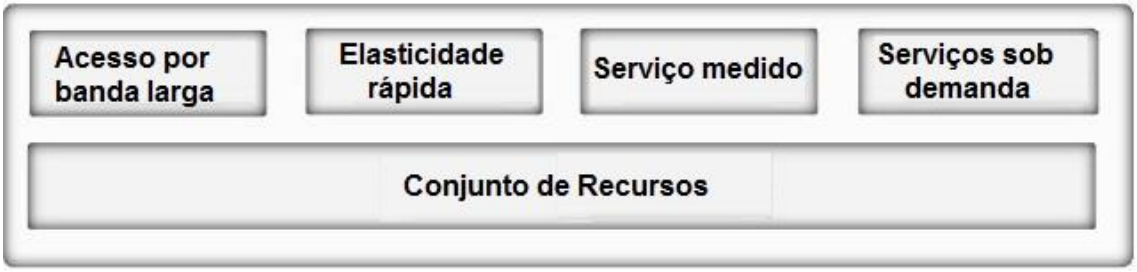
2.1.1 DEFINIÇÃO E CARACTERÍSTICAS

A computação em nuvem é o resultado de um conjunto de tecnologias que chegou ao mercado antes que houvesse sido estabelecida uma terminologia específica a ela, tal fato levou diversas organizações (RYCK et. al., 2011), (MELL e GRANCE, 2009), (CSA, 2011) a estabelecerem suas próprias definições e características referentes à computação em nuvem. Dentre estas, a definição mais aceita por pesquisadores para a descrever é a de (MELL e GRANCE, 2009) *do National Institute of Standards and Technology* (NIST):

> "A computação em nuvem é um modelo que permite acesso de rede ubíquo, conveniente, por demanda a um conjunto configurável de recursos computacionais compartilhados que podem ser rapidamente provisionados e liberados com um mínimo esforço de gerenciamento ou interação do provedor de serviço".

Esta definição fundamenta de maneira sucinta a finalidade da computação em nuvem, determinando que por qualquer aparelho com acesso a *web,* que seja conveniente ao usuário e a qualquer momento, um conjunto de recursos computacionais físicos ou virtuais devem estar a disposição do usuário, liberando ou provisionando mais recursos de acordo com as suas necessidades. Para refinar esta definição, ainda no documento de (MELL e GRANCE, 2009) são citadas cinco características essenciais (**Figura 1**Adaptado de: (CSA, 2011).):

Figura 1 - Características essenciais NIST.



Adaptado de: (CSA, 2011).

1. Serviços sob demanda (*on-demand self-service*): o usuário pode definir automaticamente o fornecimento da capacidade computacional na nuvem sob demanda, como tempo de servidor e armazenamento de rede, conforme a necessidade, sem que haja interação humana com cada prestador de serviço;

2. Acesso por banda larga (*broad network access*): recursos estão disponíveis na rede e são acessados pelo cliente através de variadas plataformas (por exemplo, telefones celulares, laptops e PDAs);

3. Conjunto de recursos (*resource pooling*): os recursos de computação (armazenamento, processamento, memória, largura de banda e máquinas virtuais) do provedor são agrupados para atender múltiplos usuários. Estes recursos (físicos ou virtuais) são atribuídos dinamicamente e de acordo com a demanda do consumidor;

4. Elasticidade rápida (*rapid elasticity)*: recursos podem ser rapidamente fornecidos para que sejam escalados. O usuário tem a impressão de que os recursos disponíveis parecem ser ilimitados e podem ser comprados em qualquer quantidade, em qualquer momento;

5. Serviço medido (*measured service)*: nuvem controla e otimiza o uso de recursos fornecendo métricas de acordo com o tipo de serviço sendo fornecido. Tanto o provedor quanto o consumidor podem monitorar e controlar a utilização dos recursos.

Estas características definem o comportamento que se espera dos serviços prestados na computação em nuvem, fornecendo juntas, flexibilidade e disponibilidade aliadas a um baixo custo para o usuário. O conjunto de recursos e a elasticidade rápida são características que estão relacionadas ao fornecimento e gerenciamento de recursos da nuvem para o usuário, características que são de fundamental importância para a nuvem, pois estas devem passar a impressão ao usuário de que os recursos da nuvem são ilimitados. A característica que se encarrega de atribuir os recursos ao usuário é a de conjunto de recursos, que deve garantir sempre a disponibilidade de *hardware* requisitado pelo mecanismo de elasticidade, este que efetua o

gerenciamento das requisições feitas pelo usuário, alocando ou liberando recursos de acordo com sua necessidade.

A característica de serviços sob demanda evidencia-se como uma das mais relevantes ao usuário, pois este não precisa contratar um pacote de serviços dos quais talvez não utilize completamente, ou que utilize e seja necessário contratar outro pacote inteiro, neste caso somente os recursos utilizados são cobrados. Este tipo de serviço, denominado "pague pelo que utilizar" (*pay-as-you-go*) tem se tornado muito comum em diversas áreas, pois paga-se por utilização e não uma cota fixa mensal como, por exemplo, nos serviços de *hosting* tradicionais (BORGES et. al., 2011).

Tal fato leva a um questionamento de outras características a respeito dos modos de se mensurar os serviços de nuvem. Segundo (BUYYA et. al., 2009) a medição dos serviços é baseada em pontos críticos estabelecidos segundo padrões definidos pela *International Organization for Standartization* (ISO), estabelecendo estes pontos especificamente para a computação em nuvem, tornando possível comparar e medir os serviços disponibilizados pelos provedores (a subseção **2.1.3** apresenta estas métricas detalhadamente). Na subseção **2.1.1** trata-se de um dos recursos mais importantes para a característica de conjunto de recursos, a virtualização.

2.1.1 VIRTUALIZAÇÃO DE RECURSOS

A virtualização de recursos esta englobada na categoria de conjunto de recursos definida pelo NIST, trata-se de um dos mais importantes conceitos utilizados na computação em nuvem, permitindo que em um mesmo hardware sejam executados simultaneamente dois ou mais ambientes distintos e isolados denominados de VM’s (Máquinas Virtuais) (POPEK e GOLDBERG, 1973). Este fato faz com que haja grande eficiência no uso de recursos de *hardware* que estejam ociosos, resultando em economia direta para as infraestruturas de TI (Tecnologia da Informação). Paralelo a isto, existe o risco de disputa por recursos dentro de um mesmo *hardware* caso haja um pico de

usuários em atividade no mesmo instante, o que deflagra a necessidade de cordenação destas VM's. Considerando esta necessidade de gerenciamento dos recursos físicos, a IBM introduziu em 1967 o monitor de máquinas virtuais ou *hypervisor,* cuja função é gerenciar o uso do *hardware,* monitorando e forçando políticas sobre as VM's, controlando ativamente o uso do *hardware* (IBM, 2013).

A tese de (KOSLOVSKI, 2011 apud Garcia-espin et al., 2010) cita quatro paradigmas para compartilhamento de recursos que a técnica de virtualização utiliza:

- **Abstração (1:1)**: representa a classe de recursos físicos expostos como uma entidade única que pode ser reservada e configurada pelo usuário;
- **Particionamento (1:N)**: um recurso físico particionado para N entidades virtuais, é o principal paradigma explorado pelas tecnologias de virtualização;
- **Agregação (N:1)**: consiste em agrupar um conjunto de recursos físicos e os expor o resultado combinado como se fosse de apenas uma única entidade; e
- **Transformação (N**:**M)**: é a combinação de particionamento e agregação. As entidades virtuais expostas pelos recursos físicos particionados pode ser combinada de maneira independente da fonte física.

Estes paradigmas são explorados de diferentes maneiras em uma infraestrutura de TI, incluindo *hardware*, sistemas operacionais, linguagens de programação, equipamentos de rede, entre outros. Todas estas características supracitadas em **2.1** e a virtualização de recursos podem ser disponibilizadas aos usuários através de serviços oferecidos por meio dos modelos de serviço, que representam a maneira em que a nuvem pode ser utilizada para disponibilizar os recursos aos usuários.

2.1.2 MODELOS DE SERVIÇO

Os modelos de serviço referem-se à maneira como o conjunto configurável de recursos computacionais da nuvem (subseção **2.1.1**), pode ser entregue ao usuário, ou seja, a maneira como todo este conjunto gerenciável de recursos pode satisfazer as conveniências do usuário. No entanto, estes modelos carecem de uma padronização geral por parte dos provedores de serviços de nuvem, pois estes criam modelos de serviço de acordo com suas necessidades e objetivos de negócio, dificultando o entendimento do conceito por parte dos principais interessados, os usuários.

Novamente o NIST, no documento de (MELL e GRANCE, 2009), define três modelos de serviço que são utilizados como referência por diversos pesquisadores (SLIPESTKYY, 2011), (MARCON et. al., 2010), (CASTRO et. al., 2012) e importantes organizações, como CSA (*Cloud Security Alliance*) (CSA, 2011), ENISA (*European Network and Information Security Agency*) (RYCK et. al., 2011) e o próprio NIST. A adoção destes modelos de serviço por grande parte dos pesquisadores e organizações relevantes do setor, aponta indícios do início da padronização do conceito partindo dos pesquisadores, cujo objetivo é suprir a carência de padronização criada pela difusão de diversos modelos de serviço, concebidos por empresas que visam atender seus próprios fins comerciais.

Este trabalho adota a linha do NIST por ser a mais aceita da comunidade, sendo os modelos de serviço desta classificação (MELL e GRANCE, 2009):

1. **Infraestrutura como Serviço/IaaS (*Infrastructure as a Service*)**: tem como objetivo entregar os recursos computacionais da nuvem para atender as necessidades dos usuários, trata-se do modelo mais básico dos serviços oferecidos pela nuvem. Nele os usuários podem utilizar e/ou implementar qualquer software, incluindo sistemas operacionais, desde que estes não controlem ou gerenciem a infraestrutura básica da nuvem, também possuem liberdade para implementar e utilizar aplicativos (CIGOJ, 2011). O mecanismo de virtualização é uma das

principais abordagens, pois permite grande flexibilidade no uso da camada de *hardware*. As máquinas virtuais fornecem ambientes de processamento que são isolados e independentes, podendo ser criadas e destruídas facilmente (MARCON et. al., 2010).

IaaS é uma opção interessante quando o objetivo de negócio do usuário, seja este uma empresa ou organização, necessite de uma infraestrutura computacional, como um *data center*, mas que manter esta infraestrutura seja inviável economicamente. A demanda por estas estruturas computacionais e o alto preço para mantê-las gerou uma necessidade por provedores especializados de IaaS (ORLANDO, 2011), fazendo com estas tarefas e despesas de manutenção e gerenciamento (espaço físico, climatização, conectividade, energia elétrica, arquitetura de rede, segurança física e lógica, entre outros) fiquem à cargo do provedor do serviço e o usuário possa focar diretamente em seu objetivo de negócio. Exemplos: Amazon Simple Storage Service [1] (S3), RackSpace Cloud Servers[2], OpenStack[3], CloudStack[4], OpenNebula[5];

2. **Plataforma como Serviço/PaaS (*Plataform as a Service*)**: possui alguns benefícios para desenvolvedores de aplicações, pois estes podem alugar o hardware implementar, testar e disponibilizar suas aplicações em ferramentas e linguagens que são suportadas pelo provedor da nuvem (CIGOJ, 2011). O objetivo deste modelo é reduzir os custos e a complexidade de se lidar com o hardware da camada subjacente (RYCK et. al., 2011). Segundo (CASTRO et. al., 2012), um grande fator inibidor de PaaS é o fato de que as aplicações desenvolvidas em uma PaaS normalmente ficam presas ao fornecedor, fazendo com que as aplicações estejam presas somente a um provedor de nuvem. Tal fato merece atenção de potenciais usuários ao escolher

[1] http://aws.amazon.com/pt/s3/
[2] http://www.rackspace.com/cloud/
[3] http://www.openstack.org/
[4] http://cloudstack.apache.org/
[5] http://opennebula.org/about:about

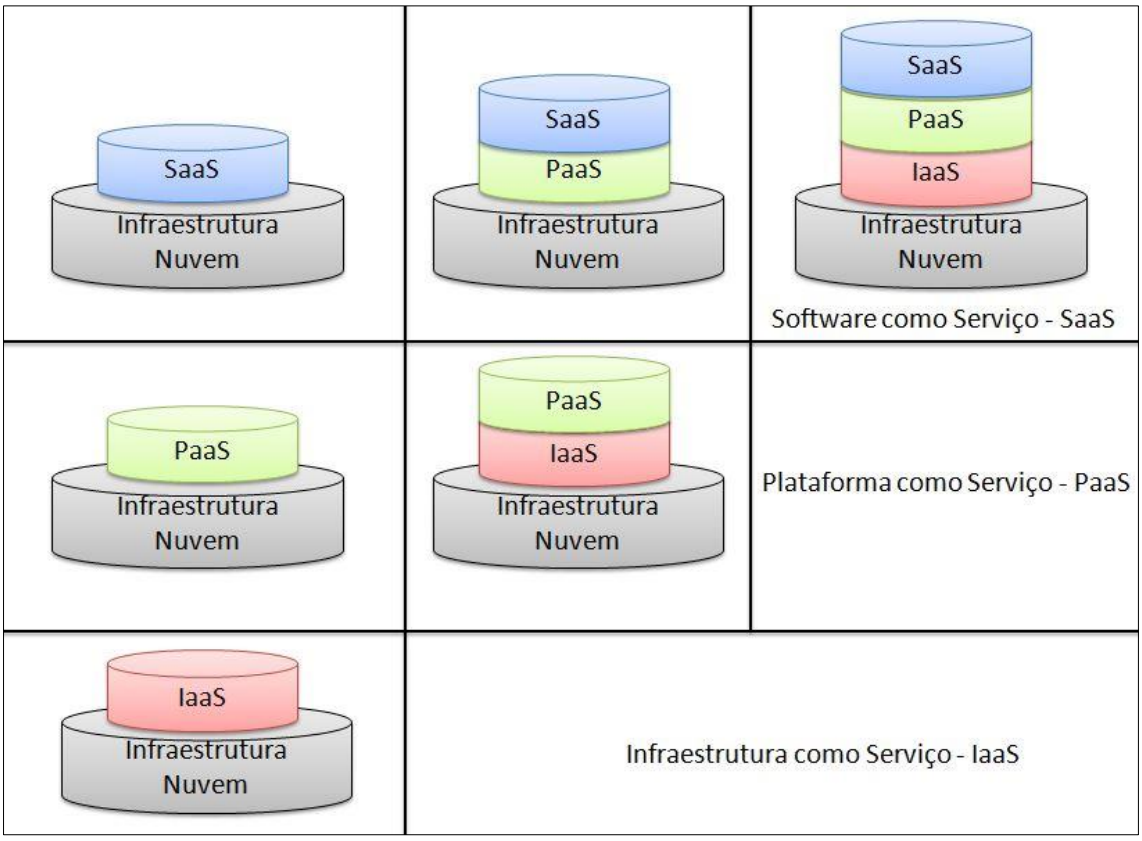
uma solução de PaaS. Exemplos: Google App Engine[6], Microsoft Azure[7], Salesforce Force.com[8];

3. ***Software* como Serviço/SaaS (*Software as a Service)***: O modelo de serviço de software como serviço (SaaS) oferece o produto final da computação em nuvem, a aplicação pronta para ser utilizada pelo consumidor através da web. Neste modelo o consumidor não gerencia ou controla as camadas subjacentes de serviços, somente a própria aplicação (MARCON et. al., 2010) e todo o software e os dados associados são armazenados de maneira centralizada, facilitando o gerenciamento e suporte da nuvem às aplicações. De acordo com (CASTRO et. al., 2012) o SaaS ainda tem uma série de desafios a serem vencidos, com destaque para problemas regulatórios, integração com os recursos internos da organização, disponibilidade e mais especificamente a segurança da informação. Exemplos: Google Docs[9], Microsoft Office Live[10].

Esses modelos de serviço podem ser agrupados de diferentes formas, assim como os recursos para disponibilizá-los. Sendo assim, é possível fornecer um serviço de nuvem a partir de outros serviços de nuvem ou a partir de uma infraestrutura específica para estes fins. Nesse sentido, a **Figura 2** ilustra algumas das possíveis configurações de serviços da nuvem.

[6] https://developers.google.com/appengine/
[7] http://www.windowsazure.com/pt-br/
[8] http://www.salesforce.com/br/
[9] http://docs.google.com
[10] https://skydrive.live.com

Figura 2 - Configurações de Serviço da Nuvem.



Fonte: Próprio Autor.

A **Figura 2** auxilia na compreensão geral de como os modelos de serviço podem ser oferecidos na nuvem. A começar pelos servidores, o hardware no qual todas as abstrações de serviços são implementadas, na **Figura 2** - Configurações de Serviço da Nuvem. é representado pela 'infraestrutura da nuvem'. O modelo de IaaS, sendo a abstração mais básica dos serviços da nuvem, é provisionado através de máquinas virtuais nas quais o usuário pode escolher a configuração a ser utilizada. O modelo PaaS pode ser oferecido como somente a plataforma, apresentando o framework de suporte e gerenciamento aos aplicativos da nuvem, ou ambos PaaS e IaaS em conjunto. Os serviços de SaaS, podem ser oferecidos de três maneiras distintas, somente a hospedagem da aplicação;a hospedagem da aplicação em conjunto com PaaS ou SaaS, PaaS e IaaS.

Estes três modelos de serviço caracterizam o que pode ser oferecido em cada camada de abstração de serviços da nuvem, fazendo com que modelos mais específicos possam ser englobados em uma ou mais destas três categorias de serviço. Para exemplificar tal fato é possível citar (GONZALEZ et. al., 2011) que descreve taxionomias de diferentes autores, por exemplo, taxionomia de dez camadas de Dave Linthieum, taxionomia de seis camadas de Sam Johnston, demonstrando que estas podem ser englobadas no modelo de serviço criado pelo NIST e também propõe uma taxionomia na qual permite a conservação dos principais serviços propostos pelo NIST nos níveis mais altos, e uma categorização de diversos serviços da nuvem nos níveis mais baixos.

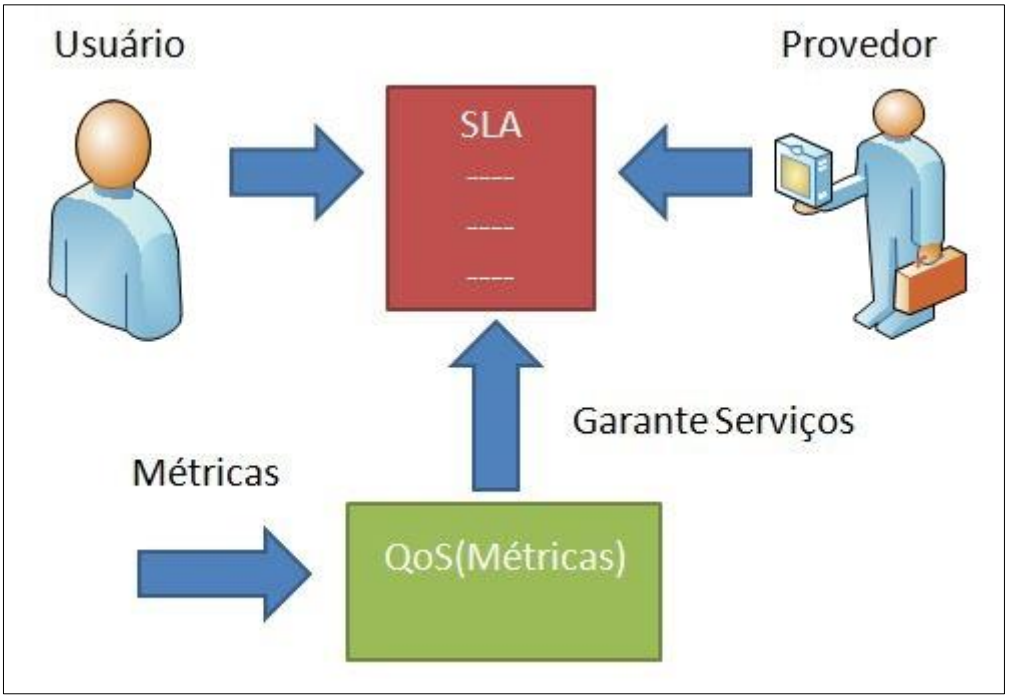
Estas taxionomias "adicionais" visam especificar ainda mais os serviços oferecidos pela nuvem, entretanto não são usualmente citadas por pesquisadores, pois a nuvem pode oferecer diversos tipos de serviço e se houver uma especificação para cada um deles a lista de siglas '*as a service'* pode ficar grande. Este é um dos motivos que justifica a ampla utilização dos modelos de serviço especificados no documento do NIST, pois os três modelos de serviço supracitados (IaaS, PaaS, SaaS) abrangem de maneira geral os serviços oferecidos pela nuvem.

2.1.3 MEDIÇÃO DOS SERVIÇOS DA NUVEM

Os serviços de utilidade pública, como água, luz, telefone, devido sua importância e frequência de utilização no dia-a-dia, devem estar disponíveis a qualquer momento aos consumidores, porém os usuários pagam aos provedores destes serviços apenas a quantidade que foi consumida durante um determinado período, normalmente por mês. De maneira semelhante, os sistemas de gerenciamento da nuvem, fazem a medição dos serviços na nuvem, controlando e aperfeiçoando o uso dos recursos por meio de medições que consideram cada tipo de serviço provido para o usuário (MEIERS, 2011).

Esta medição tem que ser transparente tanto para o provedor da nuvem quanto para o usuário, sendo que normalmente são utilizados contratos referentes aos serviços (SLA - *Service Level Agreement*) para especificar as características dos serviços, parâmetros de qualidade (QoS – *Quality of Service*) para determinar os valores que serão cobrados (BORGES et. al, 2011). A **Figura 3** ilustra a negociação da SLA pelo usuário e o provedor, as métricas utilizadas passadas como parâmetro para QoS, ajudando a estabelecer uma garantia de qualidade aos serviços de nuvem.

Figura 3 - Negociação para Estabelecer SLA.



Fonte: Próprio Autor.

Compreender as definições de SLA's e QoS's da **Figura 3**, é fundamental para estabelecer um serviço de nuvem satisfatório ao usuário:

- **SLA (*Service Level Agreement)***: define os níveis de disponibilidade, funcionalidade, desempenho e outros atributos relativos aos serviços, incluindo inclusive penalidades para o caso de violação das regras por qualquer uma das partes (BORGES et. al., 2011). Em análise feita por (GOLDEN, 2011) é elucidado que estas SLA's devem ser muito bem elaboradas entre os usuários e os provedores da nuvem, pois elas

especificam o que o usuário quer da computação em nuvem, por exemplo: a especificação da seleção de um servidor mais robusto e o pagamento de uma taxa mais elevada para garantir uma resposta mais rápida. (GOLDEN, 2011) também afirma que nestas SLA's também devem estar especificadas as punições e compensações caso o provedor falte com algum termo descrito no contrato;

- **QoS (*Quality of Service*)**: trata da garantia de qualidade dos serviços que são oferecidos neste caso pela nuvem, o provedor deve possuir mecanismos para garantir os diferentes parâmetros de especificados no SLA firmado com cada cliente. Esses parâmetros de QoS podem ser tempo de resposta, confiabilidade, segurança, entre outros (BOGOSSIAN et. al., 2010).

A medição da QoS dos serviços da nuvem segundo (GARG et. al., 2013) é baseada na *International Organzation for Standartization* (ISO) e padronizada pela *Cloud Service Measurement Index Consortium* (CSMIC)[11], consistindo em um conjunto de pontos chaves, que mede aspectos relevantes da nuvem para o usuário denominados de *Key Performance Indicators* (KPI's), que fornecem um método padronizado para medição e comparação dos serviços de nuvem. As KPI's propostas para medir os serviços de nuvem são (GARG et. al., 2013):

1. **Tempo de resposta do serviço (*service response time)***: a eficiência de uma disponibilidade de serviço pode ser medida em termos do tempo de resposta, isto é, no caso de IaaS, quão rápido o serviço pode ser disponibilizado para o uso. Por exemplo, se um utilizador solicita uma máquina virtual de um provedor de nuvem, em seguida, a resposta do serviço irá representar o tempo que o provedor responder esta requisição;

2. **Sustentabilidade (*sustainability*)**: é definida em termos de impacto ambiental do serviço de nuvem utilizado. Ela pode ser medida como a média de emissões de carbono ou de eficiência energética do serviço de

[11] http://www.cloudcommons.com/pt/about-smi

nuvem. A propriedade sustentabilidade é classificada como um atributo d e responsabilidade, que é usado para medir as propriedades relacionadas com o prestador de serviços da própria organização, independente de serviços que estão sendo prestados;

3. **Adequação (*suitability)***: é definida como o grau em que as requisições de um cliente são cumpridas pelos provedores de nuvem. A métrica é definida pelo número de serviços não essenciais fornecidos pela nuvem, dividido pelo número de serviços essenciais requisitados pelo cliente;

4. **Precisão (*accuracy*)**: a precisão do serviço mede o grau de proximidade com os valores reais do usuário ao usar um serviço em comparação com os valores esperados. Para recursos computacionais tais como máquinas virtuais, primeiro indicador de precisão é o número de vezes que o provedor de nuvem se desviou de um SLA prometida;

5. **Transparência (*transparency*)**: indica a medida que a usabilidade dos serviços por parte dos usuários é afetada devido alguma mudança no serviço;

6. **Interoperabilidade (*interoperability*)**: é a habilidade de um serviço de interagir com outros serviços oferecidos pelo mesmo provedor ou outros provedores de nuvem;

7. **Disponibilidade (*availability*)**: é a percentagem total de tempo em que um usuário pode acessar o serviço que foi contratado na SLA;

8. **Confiança (*reliability*)**: reflete como um serviço funciona sem falhas durante certo tempo ou condição. Portanto, é definida baseada no tempo médio até a falha prometido pelo provedor da nuvem e o histórico de falhas anteriores relatados por outros usuários;

9. **Estabilidade (*stability*)**: é definida como a variação no desempenho de um serviço, por exemplo, para o armazenamento seria a variação média de tempo de escrita e leitura, para recursos computacionais seria o desvio do desempenho que foi contratada na SLA;

10. **Adaptabilidade (*adaptability*)**: é a capacidade do provedor da nuvem de ajustar mudanças nos serviços com base nas requisições dos usuários. É definida como o tempo necessário para se adaptar às mudanças e atualização do serviço a um nível mais elevado;

11. **Elasticidade (*elasticity*)**: é definida em termos de quanto um serviço de nuvem pode ser escalado durante horários de pico. Isto é definido por dois atributos: tempo levado para expandir ou contrair a capacidade do serviço, e a capacidade máxima do serviço.

Este conjunto de métricas estabelecidas pela CSMI ajudam a estabelecer a QoS que é esperada dos serviços de nuvem. Os requisitos de qualidades são utilizados para determinar tanto o tipo de nuvem a ser criada por um provedor como ajudar o usuário decidir qual nuvem melhor se adapta às suas necessidades, inclusive ajudando-o a estabelecer critérios de comparação para seleção de uma nuvem. Neste sentido, também há diversas abordagens de implantação que o usuário pode optar para adequar a nuvem de acordo com suas necessidades.

2.1.4 ABORDAGENS DE IMPLANTAÇÃO

As abordagens de implantação são os tipos de nuvens cuja escolha de utilização depende das particularidades, objetivo de negócio ou tipo da informação a ser armazenada de cada empresa ou organização (CASTRO et. al, 2012). Normalmente as nuvens são classificadas de acordo com a localização da infraestrutura e os usuários que a acessam. A localização da infraestrutura determina se os recursos são privados, quando pertencem a uma única organização, ou público, quando os recursos estão em uma estrutura distribuída (KOSLOVSKI, 2011). As quatro abordagens de implantação segundo o NIST são (MELL e GRANCE, 2009):

1. **Pública**: é caracterizada por expor seus recursos computacionais sob a forma de serviços que podem ser contratados por usuários. A infraestrutura da nuvem é disponibilizada através da Internet para o público em geral ou para um grande grupo industrial e é gerida por uma entidade que venda serviços em nuvem. Por este motivo não podem ser aplicadas restrições de acesso quanto ao gerenciamento de redes, e ainda menos, aplicar técnicas de autenticação e autorização (CASTRO et. al, 2012);

2. **Privada**: a infraestrutura da nuvem é utilizada exclusivamente por um único usuário ou organização. Geralmente, é construída sobre um datacenter privado e pode ser gerida pelo usuário/organização ou por terceiros. Os membros da organização têm exclusividade de usados recursos computacionais, obtendo uma maior confiabilidade e confidencialidade sobre informações e dados críticos (KOSLOVSKI, 2011). Embora traga algumas facilidades por estar em ambiente privado, este modelo exige gerenciamento interno, aumentando custos e deixando o sistema engessado em termos de automação de tarefas como atualizações (CASTRO et. al., 2012);

3. **Comunitária**: utilizada quando várias organizações apresentam requisitos semelhantes (condições de segurança, políticas de segurança, etc.) e decidem compartilhar parte das suas infraestruturas.

   Esta pode ser gerenciada pelas organizações participantes da comunidade ou por um terceiro, em implementação local ou remota;

4. **Híbrida**: qualquer tipo de combinação de duas ou mais das categorias anteriores. Caracteriza-se por se comportar como uma nuvem pública e também como uma nuvem privada, combinando estes dois conceitos.

Cada abordagem de implantação possui seu benefício de acordo com os objetivos de quem está contratando o serviço de nuvem. Nuvens privadas garantem uma maior segurança sobre os ativos armazenados e um controle maior sobre os mesmos pois a nuvem é mantida numa rede privada. Já as nuvens públicas oferecem uma eficiência maior para recursos compartilhados, porém estando exposta ao público geral sua segurança deve ser eficiente e bem planejada. A abordagem comunitária pode ser entendida como uma nuvem de nuvens, na qual organizações compartilham suas nuvens e uma pode acessar a outra, e por último, a abordagem híbrida que pode combinar as características de duas ou mais abordagens de implantação.

A **Tabela 1** apresenta uma comparação entre as características de cada abordagem, com o objetivo de simplificar a visualização das características de cada abordagem de implantação quanto à: gerência, propriedade, localização e segurança.

Tabela 1 - Comparação Entre Abordagens de Implantação.

	Gerência	**Propriedade**	**Localização**	**Segurança**
Pública	Terceiros	Terceiros	Externa ou Interna	Baixa
Privada	Própria	Própria ou Terceiros	Externa ou Interna	Alta
Híbrida/Comunitária	Própria ou Terceiros	Própria ou Terceiros	Externa e Interna	Média

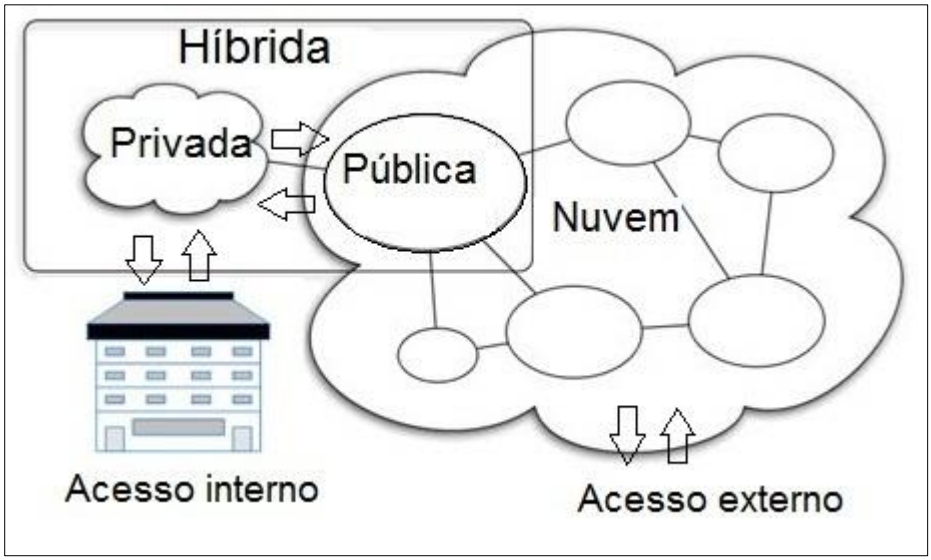
Adaptada de (CSA, 2011).

A gerência diz respeito a quem controla a nuvem, quem organiza e administra os serviços da nuvem, esta gerência pode ser própria para nuvens privadas, de terceiros para nuvens públicas e no caso de nuvens híbridas/comunitárias, própria ou de terceiros. A propriedade informa quem fornece os serviços da nuvem, uma nuvem privada pode possuir a propriedade

da nuvem ou utilizar o serviço fornecido de terceiros, uma pública é fornecida por terceiros, já as híbridas/comunitárias, por ser uma mistura de nuvens públicas e privadas, os serviços da nuvem podem ser de propriedade própria e/ou de terceiros. A localização se refere ao posicionamento da parte física da nuvem, a sala-cofre na qual está localizado o *hardware* sob o qual as camadas da nuvem são construídas para nuvens públicas a localização é externa ou desconhecida, para nuvens privadas a localização pode ser interna no caso em que nuvem seja própria ou externa, e para nuvens híbridas externa e/ou interna.

A **Figura 4** ilustra um exemplo de nuvem privada geralmente utilizada por empresas e instituições com acesso exclusivo interno; um exemplo de nuvem híbrida combinando nuvem privada e pública; e um exemplo de nuvem pública que é aberta tanto para empresas como para usuários comuns.

Figura 4 - Tipos de Nuvem



Adaptado de: (LINKHEAD, 2010).

Ao escolher utilizar uma nuvem privada, a organização mantém seus dados limitados ao acesso interno garantindo que seu nível de segurança seja alto, pois a gerência e utilização da nuvem são realizadas pela própria

organização, fato observado na **Figura 4** e **Tabela 1**. Ao utilizar uma nuvem híbrida esta organização permite que sua nuvem privada comunique-se com uma nuvem (pública ou privada) de gerência desconhecida, tornando o nível de segurança um pouco mais baixo devido à troca de dados com uma entidade cujas políticas de segurança sejam desconhecidas (RYCK et. al., 2011). A nuvem pública é o tipo de nuvem que possui nível de segurança mais baixo, pois está aberta a qualquer perfil de usuário que conheça a localização do serviço. Por isso não podem ser aplicadas restrições de acesso quanto ao gerenciamento de redes, e menos ainda, aplicar técnicas de autenticação e autorização (CASTRO et. al., 2012).

Determinada a abordagem de implantação a ser utilizada, é necessário compreender a arquitetura de referência, ou seja, o modelo básico de nuvem que descreve o papel de cada personagem no cenário de computação em nuvem, suas funções, comunicações e componentes arquiteturais. Esta arquitetura auxilia o usuário a compreender a função das partes envolvidas na computação e nuvem, como também as responsabilidades de cada uma destas partes.

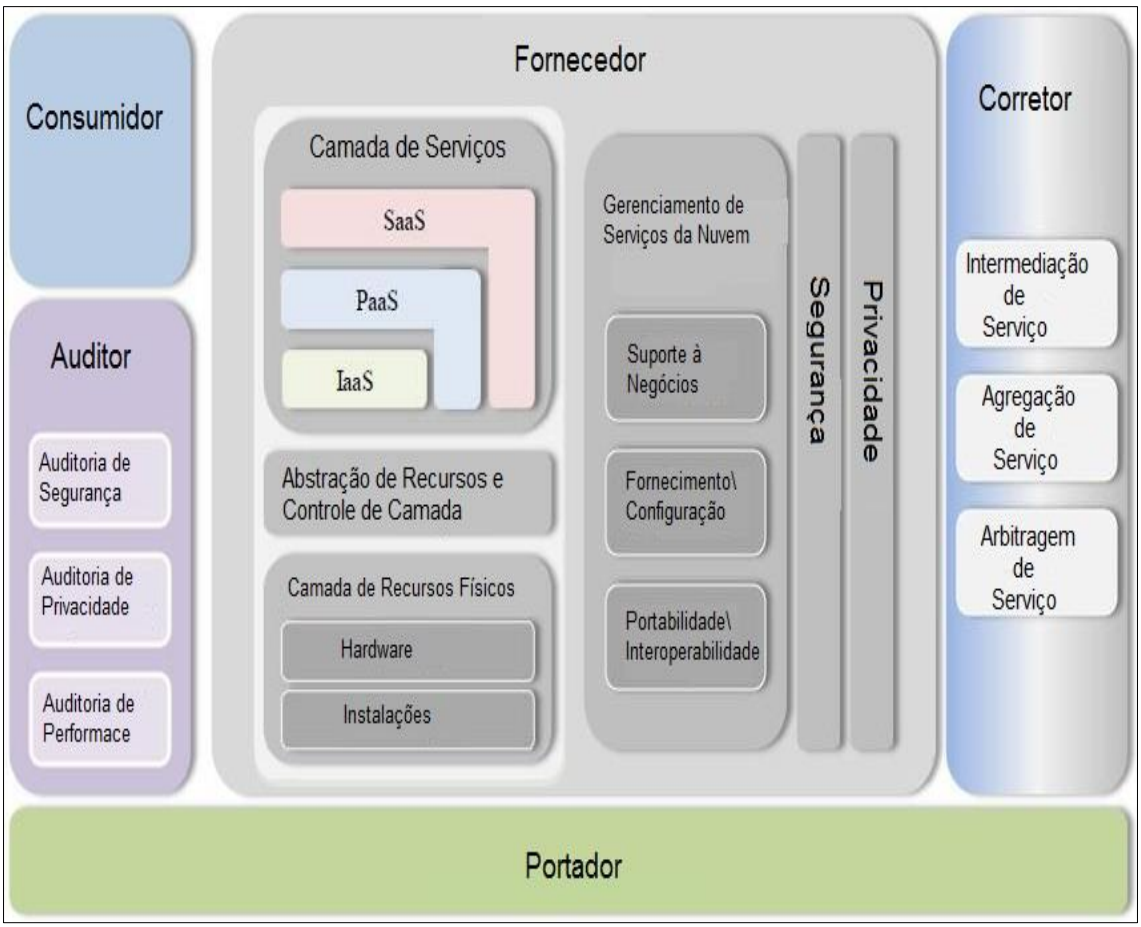
2.1.5 ARQUITETURA DE REFERÊNCIA

Assim como diversas questões sobre computação em nuvem, ainda não existe uma definição única de arquitetura ideal para esse tipo de tecnologia (BOGOSSIAN et. al., 2010), as arquiteturas dos diversos sistemas de computação em nuvem diferem entre si, dependendo da política adotada por cada provedor ou desenvolvedor. Existem diversas definições de arquiteturas de referências de organizações reconhecidas como ORACLE, IBM e NIST (ORACLE, 2012), (BEHRENDT et. al., 2011), (LIU et. al., 2011) respectivamente. Este trabalho novamente seguirá pela linha de pesquisa do NIST, por esta ser uma organização que estabelece padrões bem aceitos por

pesquisadores e organizações, através de seus guias e recomendações de segurança em computação em nuvem.

Definir a arquitetura de referência é o primeiro passo para se garantir uma nuvem com alta maturidade (BEHRENDT et. al., 2011), pois migrar para uma nuvem é uma grande mudança de paradigma para uma organização ou empresa, ter isto bem organizado e estruturado, com os papéis bem definidos é um passo fundamental para o sucesso da adoção da nuvem. A **Figura 5** ilustra a arquitetura de referência do NIST, identificando seus principais atores, suas atividades e funções na computação em nuvem.

Figura 5 - Arquitetura de Referência do NIST.



Adaptada de: (LIU et. al, 2011).

Como ilustrado na **Figura 5**, a arquitetura de referência do NIST define cinco atores principais: consumidor, fornecedor, portador, auditores e os corretores. Cada ator é uma entidade representando uma pessoa ou uma organização que participa no processo ou realiza alguma tarefa da nuvem. A **Tabela 2** lista os atores definidos em (LIU et. al., 2011) e também uma comparação com atores definidos nas arquiteturas de referência de (ORACLE, 2012) e (BEHRENDT et. al., 2011).

Tabela 2 - Comparativo entre Arquiteturas de Referência.

	NIST	**IBM**	**ORACLE**
Ator	**Definição**		
Consumidor	Uma pessoa ou organização que mantém uma relação de negócios e usa os serviços dos Fornecedores de nuvem	Uma organização, humano ou sistema de tecnologia da informação que consome, utiliza e gerencia os serviços da nuvem	O Consumidor representa uma pessoa ou organização que mantém uma relação de negócios e utiliza os serviços do Provedor
Fornecedor	Uma pessoa, organização, ou entidade responsável por fazer um serviço disponível as partes interessadas	Pessoa ou organização que tem a responsabilidade de fornecer os serviços de nuvem ao Consumidor	Deve prover a infraestrutura de nuvem, mantendo e satisfazendo os acordos estabelecidos na SLA com o Consumidor
Auditor	Uma parte que pode conduzir uma assessoria independente dos serviços da nuvem, informações sobre operações do sistema, desempenho e segurança	Não há	Não há
Corretor	Uma entidade que gerencia o uso, desempenho e entrega dos serviços da nuvem, e negocia a relação entre Fornecedores e Consumidores	Não há	Age como intermediário entre Consumidores e Fornecedores. Prove assistência para que Consumidores escolham a plataforma que melhor se encaixa em seus negócios.
Portador	Um intermediário que fornece conectividade e transporte de serviços de nuvem do Fornecedor para o Consumidor	Não há	Não há

Adaptada de (LIU et. al., 2011) (ORACLE, 2012) e (BEHRENDT et. al., 2011).

Cabe observar que (ORACLE, 2012) e (BEHRENDT et. al., 2011) definem somente três atores em suas arquiteturas de referência: Fornecedor, Consumidor e Corretor (ORACLE, 2012) e Fornecedor, Consumidor e Criador (BEHRENDT et. al., 2011). Através de um detalhamento do papel dos atores descritos na **Tabela 2** é possível compreender com mais detalhes o papel de cada autor na visão de cada uma destas organizações:

1. **Fornecedor**: é o ator que mais possui responsabilidades, sendo este o mais importante dos atores, pois é responsável por manter toda a estrutura física e lógica da nuvem, tendo que prover todos os serviços requisitados pelo Consumidor mantendo ao mesmo tempo a disponibilidade, segurança e privacidade (LIU et. al., 2011). A definição do papel do Fornecedor da (ORACLE, 2012) e (BEHRENDT et. al., 2011) estão de acordo com (LIU et. al., 2011) afirmando que este é o papel de maior complexidade, pois implementar e manter a nuvem para estar de acordo com as SLA's dos consumidores, é uma tarefa que exige extenso planejamento e execução precisa;

2. **Auditor**: segundo (LIU et. al., 2011) é responsável por examinar de maneira independente, se os serviços oferecidos pelo Fornecedor cumprem o que está previsto em termos de segurança, desempenho, privacidade, entre outros. Seria um papel de auditoria básica, como em qualquer outro serviço, apenas para verificar se o funcionamento estão de acordo com os padrões estabelecidos na SLA.
   Este papel não é especificado em (BEHRENDT et. al., 2011) e (ORACLE, 2012);

3. **Consumidor**: representa a principal parte interessada nos serviços do Fornecedor, segundo (LIU et. al., 2011) é uma pessoa ou organização que mantém uma relação de negócios, e utiliza os serviços do Fornecedor. O Consumidor necessita da SLA para especificar os detalhes técnicos do serviço e garantir a QoS do que se está sendo contratado, o Fornecedor também pode especificar na SLA os serviços

que não estão inclusos, limitações, obrigações para o Consumidor, entre outras especificações. (BEHRENDT et. al., 2011) diz que o Consumidor é uma organização, um humano ou um sistema que requisita, usa e gerencia instâncias de serviço. (ORACLE, 2012) possui a mesma definição de (LIU et. al., 2011) afirmando que o uso da SLA é fundamental para ditar o escopo dos serviços;

4. **Corretor**: como pode ser observado na Adaptada de: (LIU et. al, 2011)., (LIU et. al., 2011) define que o Corretor é responsável por três serviços:

4.1. Intermediação de Serviço: aumenta um determinado serviço, melhorando alguma capacidade específica e fornecendo serviços de valor agregado aos Consumidores;

4.2. Agregação de Serviço: combina e integra múltiplos serviços em um ou mais novos serviços. Fornece a integração dos dados e garante a movimentação segura dos dados entre o Consumidor e múltiplos Fornecedores;

4.3. Arbitragem de Serviço: similar a agregação de serviços, porém os serviços agregados não são fixos, é dada a liberdade ao Corretor para escolher os serviços de um outro Fornecedor;

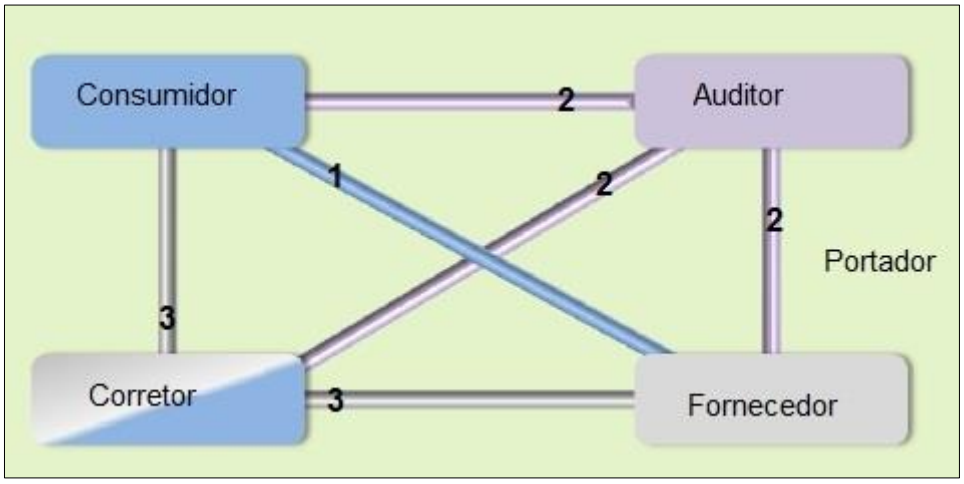
A abordagem de (LIU et. al., 2011) sobre o papel do Corretor é a mais completa do que em (ORACLE, 2012), pois realiza uma análise específica sobre cada função de gerenciamento de serviço que o Corretor pode realizar. (ORACLE, 2012) cita brevemente a função de gerenciamento de serviços do Corretor, afirmando que este pode realizar o intermédio de serviços entre o Consumidor e o Fornecedor. (BEHRENDT et. al., 2011) não cita este papel em sua especificação;

5. **Portador**: funciona como um intermediador que fornece conectividade e transporte para os serviços de nuvem entre o Fornecedor e o Consumidor (LIU et. al., 2011). Fornecem acesso através da rede, telecomunicação e outros serviços. Este papel não é especificado em (BEHRENDT et. al., 2011) e (ORACLE, 2012).

Através desta comparação entre o papel dos atores na arquitetura de referência, percebe-se que (LIU et. al., 2011) realiza uma análise mais detalhada da função de cada ator em comparação com (ORACLE, 2012) e (BEHRENDT et. al., 2011) , que apenas descrevem as funções dos papéis mais básicos: do Fornecedor e Consumidor. Fato compreensível, visto que descrevem arquiteturas que são específicas das suas soluções de nuvem, então a ênfase é dada somente nos atores que possuem relevância dentro de sua arquitetura.

Para ajudar a compreender a relação entre os atores, a **Figura 6** ilustra de que maneira ocorre esta comunicação entre os atores na definição de (LIU et. al., 2011):

Figura 6 - Interação Entre os Atores.



Adaptada de: (LIU et. al., 2011).

O Consumidor pode requisitar os serviços do fornecedor diretamente pelo canal de comunicação 1 ou requisitar os serviços através do Corretor, que realiza a intermediação entre o Consumidor e o Fornecedor através dos canais de comunicação 3. O Auditor, para realizar sua auditoria de maneira independente, pode coletar as informações necessárias dos outros atores

através dos canais de comunicação 2, ligando-se assim com todos outros atores sem nenhum intermediário.

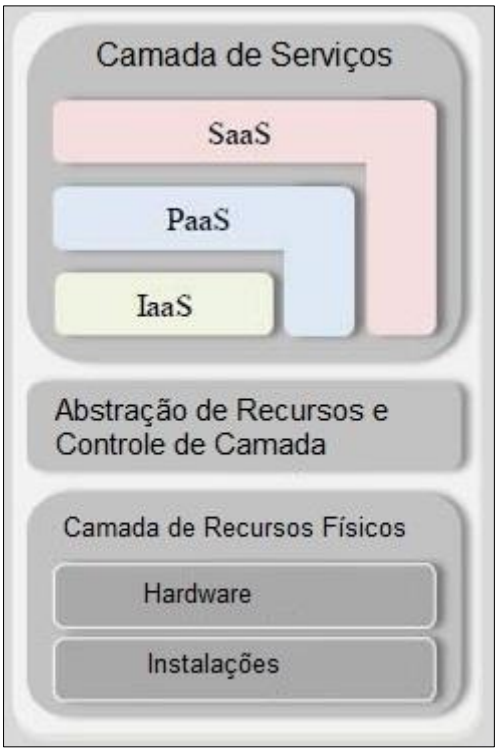
Descritos os papéis de cada ator suas funções e como se comunicam, é necessário compreender detalhadamente como o Fornecedor arranja, organiza, e provisiona os serviços, ou seja, como o Fornecedor orquestra os serviços requisitados pelo Consumidor. O Fornecedor é o ator que possui mais relevância dentre os descritos, pois todos os serviços da nuvem são providos por ele, fazer com que todos estes componentes atuem sinergicamente e apresentem os resultados de maneira confiável e segura, é um grande desafio para os Fornecedores de nuvem, e é muito importante compreender detalhadamente cada componente que torna possível estabelecer os serviços de nuvem para o usuário.

2.1.5.1 ORQUESTRAÇÃO DE SERVIÇOS

A orquestração de serviço refere-se à composição dos componentes que dão apoio ao Fornecedor da nuvem no arranjo, coordenação e gerenciamento dos recursos para prover serviços ao Consumidor (LIU et. al., 2011). A **Figura 7**

Adaptado de: (LIU et. al., 2011). ilustra a pilha básica de três camadas que provisionam os serviços da nuvem, a camada de serviços, abordada na subseção **2.1.2** não será descrita nesta subseção.

Figura 7 - Orquestração de Serviços.



Adaptado de: (LIU et. al., 2011).

A camada de abstração de recursos e controle de camada contém os componentes que fornecem e gerenciam o acesso aos recursos físicos, através de uma abstração de *software* (LIU et. al., 2011). A definição de (ORACLE, 2012) afirma que os recursos físicos precisam ser logicamente particionados e fixados para apoiar a multi-locação (*multi-tenancy*), a elasticidade requer que os recursos sejam rapidamente alocados e desalocados e a responsável por realizar isto é a camada de Abstração de Recursos e Controle de Camada. O objetivo desta camada é criar e implantar novas instâncias em tempo real, aumentando, reduzindo ou corrigindo as instâncias que já foram criadas (ORACLE, 2012). A abordagem de computação em nuvem necessita que não tenha tempo ocioso ao se executar uma destas tarefas, a execução necessita ser *on-the-fly*. Alguns exemplos de componentes: máquinas virtuais, armazenamento de dados, entre outros.

A camada mais baixa é a Camada de Recursos Físicos, que inclui todos os recursos físicos, como computadores (CPU, memórias), recursos de rede

(roteadores, *firewalls*, *switches*, entre outros), componentes de armazenamento (discos rígidos) e outros elementos da infraestrutura física (LIU et. al., 2011). Também inclui elementos das instalações, a sala-cofre, como: resfriamento, ventilação, força, ar-condicionado, entre outros elementos.

Os aspectos de gerenciamento de serviços, observados na **Figura 8**, incluem todos os serviços e funções que são necessárias para o gerenciamento e operação dos serviços que são requisitados pelos Consumidores. As características de segurança e privacidade, por serem o objetivo desta pesquisa, serão abordadas mais detalhadamente na seção **3**.

Figura 8 - Gerenciamento de Serviços.

Adaptado de: (LIU et. al, 2011).

A Camada de Gerenciamento de Serviços da Nuvem é subdividida em três, suporte a negócios, fornecimento\configuração e portabilidade\ interoperabilidade (LIU et. al., 2011):

1. **Suporte a negócios**: envolve o conjunto de serviços às empresas que lidam com clientes e processos de suporte. Inclui os componentes usados para executar operações de negócios:
   - **Gerenciamento do Cliente**: gerencia as contas do cliente, abre/fecha/termina contas, gerencia os perfis de usuário, gerencia as relações, entre outros;

   - **Gerenciamento de Contrato**: gerencia os contratos de serviço, configura/negocia/fecha/termina contratos, entre outros;

   - **Gerenciamento de Inventário**: monta e configura catálogos de serviço;

   - **Contabilidade e Faturamento**: gerencia as informações de faturamento do cliente, envia declarações de faturamento, entre outros;

   - **Relatórios e Auditoria**: monitora operações do usuário gera relatórios, entre outros;

   - **Preços e Avaliação**: Avaliar os serviços de nuvem e determinar os preços, promoções e lidar com regras de preços com base no perfil de um usuário;

2. **Fornecimento e Configuração**:
   - **Provisionamento Rápido**: implanta automaticamente sistemas de nuvem com base na requisição de serviço/recursos/capacidades;

   - **Mudança de Recursos**: ajusta a configuração de recursos para reparos, melhoramentos juntando novos nós a nuvem;

- **Monitoramento e Relatório**: descobre e monitora recursos virtuais, monitorando as operações da nuvem e gerando relatórios de desempenho;

- **Métricas**: prove a capacidade de medir em algum nível de abstração o tipo apropriado de serviço;

- **Gerenciamento de SLA**: abrange a definição de contrato da SLA, monitoramento e aplicação da SLA de acordo com as políticas definidas;

3. **Portabilidade e Interoperabilidade**:

- **Portabilidade**: a capacidade de transferência de dados de um sistema para outro sem necessidade de recriar ou reinserir descrições de dados ou modificar significativamente a aplicação a ser transportada;

- **Interoperabilidade**: capacidade de comunicar, executar programas ou transferindo dados entre várias unidades funcionais sob condições específicas.

Os componentes da arquitetura de referência descrevem os aspectos importantes das abordagens de serviço e orquestração de serviço (LIU et. al., 2011). O gerenciamento geral do serviço na nuvem é reconhecido como um importante elemento no esquema da arquitetura. Os mecanismos de suporte à negócios servem para apoiar o Consumidor em diversas situações, fornecimento e configuração aponta os requisitos que as nuvens devem ter para estarem disponíveis sempre que necessário, estarem medidos garantindo a QoS e cumprindo o estabelecido na SLA. Portabilidade e Interoperabilidade apontam para questões dos dados, apontando estes dois fatores importantes para quem migra para o modelo de nuvem.

A partir do estabelecimento de uma arquitetura de referência, é possível analisar algumas soluções de computação em nuvem, estabelecendo uma comparação bem fundamentada em aspectos observados anteriormente como:

camada em que fornece serviço, tipo de serviço, interface de acesso do usuário, arcabouço de programação, entre outros.

2.2 PLATAFORMAS DE COMPUTAÇÃO EM NUVEM

As plataformas de computação em nuvem são ferramentas e/ou projetos que viabilizam aos usuários os serviços disponibilizados nas camadas de IaaS, PaaS e SaaS (subseção **2.1.2**) (CHAPELL, 2008), prestando um serviço somente para uma camada específica ou todas dependendo da plataforma. Existem diversas opções de plataformas para cada camada de serviço, com benefícios e características diferentes, e o usuário é responsável por determinar qual a que melhor se adapta às suas necessidades.

Sabe-se que a adoção deste modelo de computação representa uma grande mudança de paradigma dentro de uma organização ou empresa (GARTNER, 2012), a escolha da plataforma deve ser resultado de uma análise bem detalhada das características de cada plataforma, para obter sucesso nos objetivos de negócio. Esta seção se propõe a apresentar algumas opções de plataformas de computação em nuvem, comparando-as com base em algumas características. Ao final desta seção é realizada uma comparação geral das características de cada plataforma apresentada, com o objetivo de esclarecer algumas diferenças entre as mesmas.

As plataformas de computação em nuvem apresentadas nesta seção serão as plataformas de destaque na computação em nuvem segundo (BURNS, 2012) (CLOUDTIMES, 2011) e plataformas *open source* por oferecerem diversos benefícios para a disseminação da computação em nuvem, pois facilitam que a comunidade possa colaborar com os projetos através de fóruns e listas de discussões; evitar o potencial *lock-in* das empresas, em um ambiente tão dinâmico é arriscado depender de soluções de uma única empresa, entre outros benefícios. As plataformas apresentadas e comparadas são:

- **IaaS**: Amazon EC2 e S3, CloudStack, OpenNebula, OpenStack;
- **PaaS**: Google App Engine, Microsoft Azure;
- **SaaS**: Salesforce.com, Drupal.

Os critérios estabelecidos (definidos no final da subseção **2.1.5.1**2.1.5.1 ORQUESTRAÇÃO DE SERVIÇOS) para caracterizar as plataformas são:

- Categoria em que a plataforma presta serviço (IaaS/PaaS/SaaS);
- Tipo de serviço (e.g., armazenamento);
- Interface de Acesso do Usuário (e.g. linha de comando, *web*);
- Disponibilidade de API's *web*;
- Arcabouço de programação (e.g., C#, Python, Java).

2.2.1 AMAZON WEB SERVICES (AWS)

A AWS é uma coleção de produtos e serviços que provisiona acesso à infraestrutura de computação oferecida pela Amazon. Inicialmente criada em 2006 e refinada com a popularização da computação em nuvem, as soluções da Amazon destacam-se no cenário de computação em nuvem como uma das mais eficientes dentre as fornecedoras de IaaS (GARTNER, 2012)(BURNS, 2012). Fato que é resultado de uma política que se preocupa aos mínimos detalhes com aspectos de hardware e infraestrutura, objetivando tornar os serviços sempre disponíveis ao consumidor (AMAZON, 2013). Os elementos centrais dessa infraestrutura, em meio a variedade de soluções oferecidas, que oferecem os blocos de construção mais comuns necessários a quase qualquer aplicativo são: Amazon S3 e EC2 abordados nas subseções **2.2.1.1 e 2.2.1.2** respectivamente.

2.2.1.1 AMAZON S3

O Amazon S3 opera na camada de infraestrutura como serviço, fornecendo uma interface de serviço web que pode ser usada para armazenar e recuperar qualquer quantidade de dados. Ele concede acesso a todos os desenvolvedores para a mesma infraestrutura com o objetivo de ser altamente escalável (AMAZON, 2013). O S3 armazena os dados como "objetos" que são agrupados em depósitos (*buckets*), que devem ser formalmente criados antes de serem usados (BURNS, 2012). A API do S3 permite que estes depósitos sejam criados, excluídos e listados, sendo que cada depósito possui uma lista de controle de acesso que permite a leitura e/ou escrita a outros usuários AWS ou para o mundo (GARFINKEL, 2007). A **Tabela 3Fonte: (AMAZON, 2013).** descreve algumas características do S3.

Tabela 3 - Características S3.

Amazon S3	
Categoria de Serviço	IaaS
Tipo de Serviço	Armazenamento
Interface de Acesso do Usuário	Linha de comando
WEB API's	REST, Amazon S3 API
Arcabouço de programação	C#, Java, PHP, Perl Python e Ruby

Fonte: (AMAZON, 2013).

A interface de acesso é através de linhas de comando, que permitem comandos básicos como: listar, criar e excluir depósitos ou objetos em um depósito, adquirir metadados de um item, entre outros. A Amazon disponibiliza implementações REST API do S3 em C#, Java, PHP, Perl Python e Ruby (GARFINKEL, 2007).

O S3 é uma das plataformas de armazenamento mais utilizadas pelos usuários, trata-se de uma plataforma confiável e altamente escalável em que o

usuário paga somente o que utilizar. A facilidade de se utilizar o serviço faz com que ele se destaque entre outras soluções de armazenamento (BURNS, 2012), o objetivo da Amazon é que o usuário se concentre apenas em construir os aplicativos que fazem uso do armazenamento, e não se preocupar em como este armazenamento é feito.

2.2.1.2 AMAZON EC2

O *Amazon Elastic Compute Cloud* (EC2) opera na camada de infraestrutura como serviço fornecendo a capacidade de aumentar ou diminuir seus recursos de computação baseando-se na demanda do consumidor, facilitando assim, o fornecimento de novas instâncias de servidor. Sua principal característica é a facilidade para criar e gerenciar instâncias virtuais com grande flexibilidade de opções para personalizá-las (AMAZON, 2013) , como por exemplo: sistema operacional, configuração de memória, de processamento, entre outros. A **Tabela 4** lista as características da plataforma.

Tabela 4 - Características EC2.

Amazon EC2	
Categoria de Serviço	IaaS
Tipo de Serviço	Gerenciamento da Nuvem
Interface de Acesso do Usuário	Linha de comando
WEB API's	Sim
Arcabouço de programação	Máquina de Imagem da Amazon baseada em GNU/Linux personalizável

Fonte: (AMAZON, 2013).

A interface de acesso do usuário é através de linha de comando, chamada de (CLI - *command line interface*), que permite controlar diversos

serviços e automatizá-los através de scripts (GARFINKEL, 2007). Possui API acessível via *web* permitindo o acesso e a operação das instâncias, seu *framework* de programação é uma versão personalizável de GNU/Linux, chamada de máquina de imagem da Amazon (AMAZON, 2013).

O EC2 é a plataforma de nuvem mais utilizada pelo mercado segundo (BURNS, 2012) para alocar e desalocar instâncias de máquinas virtuais. Possui uma variedade de serviços que permite aos usuários personalizar as instâncias de acordo com suas necessidades, sem se preocuparem com camadas de *hardware*. (BURNS,2012) ainda destaca que o baixo custo do serviço, aliados a flexibilidade, confiança e elasticidade fazem com que esta plataforma seja a mais utilizada entre os consumidores. Tais características tornam o EC2 um dos componentes de gerenciamento de nuvem mais requisitados pelos consumidores.

2.2.2 CLOUDSTACK

O CloudStack é uma plataforma de computação em nuvem desenvolvida inicialmente pela *Cloud.com*[12] e *Citrix*[13] em 2008, empresas especializadas em soluções de computação em nuvem, tendo seu projeto cedido para a *Apache Software Foundation*[14] em abril de 2012 através de uma estratégia de negócio, visando a expansão do projeto em uma era *open source* das nuvens computacionais. Tal fato permitiu acelerar a inovação, interoperabilidade e padronização do CloudStack por meio da comunidade de fornecedores e desenvolvedores (CLOUDSTACK, 2013).

A plataforma visa prover a infraestrutura como serviço para que consumidores possam disponibilizar, de maneira ágil e eficiente, serviços de nuvem para o usuário. Projetada para gerenciar grandes redes de máquinas virtuais, como também ser altamente disponível e escalável, o sistema de

[12] http://www.cloud.com
[13] http://www.citrix.com/
[14] http://incubator.apache.org/cloudstack/

gerenciamento é análogo ao EC2 da Amazon, iniciando e gerenciando instâncias virtuais sob demanda (CITRIX, 2013). A **Tabela 5** permite observar algumas características da plataforma.

Tabela 5 - Características CloudStack.

CloudStack	
Categoria de Serviço	IaaS
Tipo de Serviço	Computação em Nuvem (Armazenamento/Rede/Gerenciamento de Máquinas Virtuais,...)
Interface de Acesso do Usuário	Linhas de comando, Interface Web
WEB API's	CloudStack API, Amazon EC2 e S3 API's
Arcabouço de programação	Python, Java, Perl, C, Ruby, Shell Script

Fonte: (CLOUDSTACK 2013), (ABDUL, 2012), (CITRIX, 2013).

Os usuários podem gerenciar a nuvem executando comandos via linhas de comando ou através da interface *web*, também possui API própria e API's para os serviços da Amazon EC2 e S3 para os usuários que queiram utilizar uma nuvem híbrida (CITRIX, 2013). A plataforma teve seu código desenvolvido em diversas linguagens como Python, Java, Ruby, C e Shell Script, sendo a maior parte desenvolvida em Java (ABDUL, 2012).

Quanto ao serviço de armazenamento é permitido o armazenamento primário e secundário (CLOUDSTACK, 2013). O primário é configurado em um nível de cluster, perto dos hosts para obter melhor desempenho, guarda todos os volumes de disco de máquinas virtuais em um cluster. Secundário é configurado por níveis de zona (uma zona pode ter um ou mais armazenamentos secundários), armazena todos *templates* e imagens.

2.2.3 OPENNEBULA

Criado inicialmente como um projeto de pesquisa em 2005, por dois professores da *Universidad Complutense de Madrid*[15] , Ignácio M. Llorente e Rubén S. Montero, o projeto OpenNebula evoluiu, ganhando maturidade até março de 2008, data em que foi lançado seu primeiro *release* público, desde então a plataforma vem aumentando consideravelmente o número de colaboradores através do lançamento de novos *releases*. São estes colaboradores: desenvolvedores, grupos de pesquisa e empresas, cuja colaboração à plataforma consolidam o OpenNebula como uma opção confiável para o fornecimento de IaaS (OPENNEBULA, 2013).

O OpenNebula é um projeto divido em módulos, que oferece soluções flexíveis para o gerenciamento de *data centers* virtualizados para permitir infraestrutura sob demanda para o consumidor (OPENNEBULA, 2013). Além disso, proporciona isolamento, interface de comunicação para acesso de clouds externas, escalabilidade e possui sites de apoio para resolução de problemas e obtenção de ajuda. Permitindo que as organizações complementem a infraestrutura local para atender às demandas de pico ou implementar estratégias de alta disponibilidade, (LASZEWSKI et. al., 2012). A **Tabela 6** apresenta as características do OpenNebula.

Tabela 6 - Características OpenNebula.

OpenNebula	
Categoria de Serviço	IaaS
Tipo de Serviço	Gerenciamento da Nuvem
Interface de Acesso do Usuário	Linhas de comando, Interface Web
WEB API's	OpenNebula API, libvirt, EC2,S3 OCCI API, REST
Arcabouço de programação	Java, Ruby

Fonte: (OPENNEBULA, 2013), (LASZEWSKI et. al, 2012).

[15] http://portal.ucm.es/en/web/en-ucm/the-complutense-university

Permite o gerenciamento das máquinas virtuais através de várias interfaces de acesso, como REST (*Representational State Transfer*) e OCCI (*Open Cloud Compute Interface*), interfaces de linha de comando e o próprio navegador *web*. A autorização é baseada em senhas, pares de chaves SSH-RSA (*Secure Shell - Rivest-Shamir-Adleman)* e certificados X.509 ou LDAP (*Lightweight Directory Access Protocol*) (LASZEWSKI et. al, 2012). Também permite listas de controle de acesso (ACL's) para controlar o tráfego de rede não desejado, gerenciamento da nuvem baseado em pepéis, usuários e grupos e suporte a isolamento em diferentes níveis (OPENNEBULA, 2013).

2.2.4 OPENSTACK

Criada em julho de 2010, trata-se de uma coleção de componentes de código aberto utilizada por organizações para configurar e gerenciar sua nuvem de computação. O projeto visa construir uma comunidade *open-source* com pesquisadores, desenvolvedores e empresas, que compartilham um objetivo comum: criar uma nuvem simples de ser implementada, altamente escalável e com vários recursos avançados (WEN et al., 2012). A missão do OpenStack é "produzir uma plataforma de computação de código aberto ubíqua que satisfaça as necessidades das nuvens públicas e privadas, independente do tamanho, por ser simples de implementar e altamente escalável" (SLIPETSKYY, 2011).

Fundado pela RackSpace[16] e NASA[17] , o OpenStack cresceu para ser uma comunidade global de *software* de desenvolvedores que colaboram em um sistema de código aberto em uma nuvem padronizada e altamente escalável (OPENSTACK, 2013). Todo o código do OpenStack é aberto aos desenvolvedores, qualquer um pode executar, construir aplicativos sobre a plataforma, ou submeter modificações ao projeto.

---

[16] http://www.rackspace.com/
[17] http://www.nasa.gov/

A plataforma é composta por sete componentes principais, denominados: Swift (armazenamento de objetos), Horizon (painel ou *dashboard*), Cinder (armazenamento de blocos), Glance (serviço de imagens), Nova (gerenciamento da nuvem), Quantum (rede) e Keystone (identidade) (OPENSTACK, 2013). Estes componentes serão apresentados detalhadamente na seção **2.2.10**. A **Tabela 7** apresenta algumas características do OpenStack.

Tabela 7 - Características OpenStack.

OpenStack	
Categoria de Serviço	IaaS
Tipo de Serviço	Computação em Nuvem (Armazenamento/Rede/Gerenciamento de Máquinas Virtuais,...)
Interface de Acesso do Usuário	Linhas de comando, Interface Web
WEB API's	OpenStack API, S3, EC2 API, REST, OCCI, HTTP
Arcabouço de programação	Python, Shell Script

Fonte: (OPENSTACK, 2013), (ABDUL, 2012).

O OpenStack possui uma interface própria para que os consumidores possam acessar e gerenciar seus dados via *web*, interface OCCI, REST (ABDUL,2012). A API própria contém interfaces para cada um de seus objetos (imagem, identidade, armazenamento de bloco, rede, entre outros) (OPENSTACK, 2013). A plataforma foi desenvolvida utilizando a linguagem Python e utiliza muitas bibliotecas externas como EventLet (para programação de eventos), SQLAlchemy (para acesso ao banco de dados) entre outras, também utliza *shell scripts* para automatizar diversas rotinas (OPENSTACK, 2013).

É uma plataforma que está em franco crescimento no cenário de computação em nuvem, que foi inicialmente desenvolvida por grandes organizações (NASA e Rackspace), e posteriormente, tendo seu projeto aderido por diversas organizações. Por ser uma plataforma recente, ainda apresenta alguns entraves para que possa atingir os patamares da plataforma AWS, (ABDUL, 2012) cita alguns pontos críticos como (*release* avaliado foi o

Folsom): medição e faturamento, alta confiabilidade, segurança, monitoramento, entre outros. Como pontos favoráveis podem ser citados: fácil administração, gerenciamento de identidades, rede, custo, entre outros.

2.2.5 GOOGLE APP ENGINE

Lançada em abril de 2008, o Google App Engine é uma plataforma baseada em Python e Java da camada PaaS que permite que os usuários hospedem ou armazenem seus aplicativos em uma infraestrutura, e toda estrutura fica acessível por meio de API's (INFOQ, 2012). O grande benefício é que não há necessidade de manter servidores, os desenvolvedores apenas enviam seus aplicativos e eles ficam disponíveis para atender os usuários (DEVELOPERS GOOGLE, 2012).

Possui dois ambientes para execução dos aplicativos, Java e Python. O ambiente de execução em Java, segundo (INFOQ, 2012) permite criar aplicativos utilizando tecnologias Java padrão, incluindo JVM (*Java Virtual Machine*), *servlets* Java e a linguagem de programação Java, ou qualquer outra linguagem que usa um interpretador ou compilador com base na JVM, como JavaScript ou Ruby. O ambiente de execução em Python possui um interpretador para a biblioteca padrão (compatível com Python 2.5) podendo executar qualquer código em Python (DEVELOPERS GOOGLE, 2012), é necessário um editor de texto simples (por exemplo bloco de notas ou gedit), acesso ao sistema de arquivos via interpretador de comandos e um navegador para visualizar o trabalho (INFOQ, 2012). É um ambiente Python "puro". **Tabela 8** apresenta algumas características do Google App Engine.

Tabela 8 - Características Google App Engine.

Google App Engine	
Categoria de Serviço	PaaS
Tipo de Serviço	Plataforma de Aplicativos
Interface de Acesso do Usuário	Console de Administração baseado na Web, Linhas de Comando
WEB API's	Google App Engine API, Java API´s, Python API's
Arcabouço de programação	Python, Java

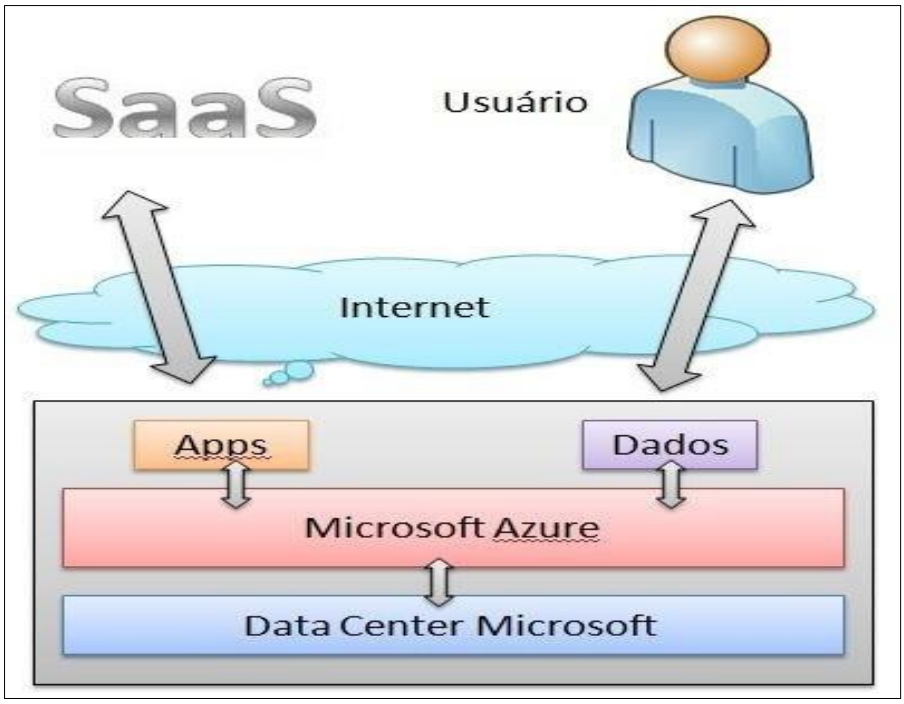
Fonte: (DEVELOPERS GOOGLE, 2012), (ABDUL, 2012).

É uma plataforma simples, porém robusta, que permite o armazenamento e a hospedagem de aplicativos em uma infraestrutura de nuvem, opera na camada PaaS, permitindo aos usuários gerenciarem seus aplicativos através de uma interface *web*. É uma plataforma do tipo *pay-as-you-go*, sendo que o serviço é grátis para aplicativos que utilizem até 500 MB de armazenamento ou cinco milhões de *page views* (DEVELOPERS GOOGLE, 2012). Possui diversas API's Java e Python, para fornecerem suporte na interpretação dos aplicativos, por exemplo: Armazenamento de Dados API (Python, Java,) Armazenamento de Dados Async API (Python, Java) Imagens API (Python, Java.) Serviços de Log API (Python), entre outras.

2.2.6 MICROSOFT AZURE

Apresentado ao público e aos desenvolvedores em outubro de 2008, o Azure é uma plataforma de aplicações para nuvens públicas que atua como ambiente de desenvolvimento, hospedagem de serviços e gerenciamento de serviços para a nuvem (MSDN, 2010). A **Figura 9** ilustra o funcionamento do Azure.

Figura 9 - Microsoft Azure.



Adaptada de: (CHAPELL, 2009).

O desenvolvimento de aplicações para a plataforma requer uma abordagem diferente das outras plataformas, o Azure roda nas máquinas dos *data centers* da Microsoft, ou seja, ao invés de fornecer um *software* que os clientes podem instalar e executar localmente em suas máquinas, a plataforma é oferecida como um serviço (CHAPELL, 2009). Os usuários utilizam o Azure para executar aplicações, armazenar informações em máquinas da Microsoft, e estas aplicações podem ser enviadas para serem disponibilizadas em uma camada SaaS para consumidores. A **Tabela 9** apresenta algumas características do Azure.

Tabela 9 - Características Microsoft Azure.

Microsoft Azure	
Categoria de Serviço	PaaS
Tipo de Serviço	Plataforma de Aplicativos
Interface de Acesso do Usuário	Portal Microsoft Windows Azure, Linha de Comando
WEB API's	Azure API, HTTP/HTTPS, REST, SOAP e XML
Arcabouço de programação	Microsoft .NET, Java, PHP, Python e Node.js.

Fonte: (MSDN, 2010), (CHAPELL, 2009).

A plataforma Azure tem o objetivo de simplificar a manutenção e o gerenciamento dos aplicativos fornecendo computação e armazenamento por demanda para hospedar, colocar em escala os aplicativos O gerenciamento de infraestrutura é automatizado com uma plataforma destinada à alta disponibilidade (MSDN, 2010). O Azure oferece um ambiente aberto, baseado em padrões e inter-operável com suporte a vários protocolos da Internet, inclusive HTTP/HTTPS, REST, SOAP e XML (*eXtensible Markup Language*). A plataforma foi desenvolvida em .NET [18] e suporta diversos ambientes de desenvolvimento para os aplicativos dos usuários, como: .NET, Java, PHP, Python e Node.js.

2.2.7 SALESFORCE.COM

É uma empresa global de *software* fundada em 1999 nos Estados Unidos. O foco dos aplicativos da Salesforce.com é o suporte nos negócios das empresas com os clientes, sendo a empresa líder de mercado em CRM (*Customer Relationship Management*) [19] possuindo o *software* mais robusto para o segmento (CRM FORECAST, 2010). Sua solução de CRM é dividida em

[18] http://www.microsoft.com/net

[19] Classe de ferramentas que automatizam as funções de contato com o cliente, essas ferramentas compreendem sistemas informatizados (CRM FORECAST, 2010).

vários módulos como: *Sales Cloud*, *Service Cloud*, *Chatter,* entre outros (SALESFORCE, 2013):

- ***Sales Cloud***: permite ao usuário gerenciar gastos de *marketing*, desempenho através de diversas opções da aplicação. Incluem macros, tomadores de decisão, comunicações com o cliente, entre outros;

- ***Service Cloud***: fornece às empresas uma visão parecida com a de um *call-center* em que é possível rastrear os serviços que estão chegando, encaminhá-los e classificar em uma escala de importância.

- ***Chatter***: é básicamente um serviço de apoio ao *Sales Cloud*, de troca de mensagens entre os usuários. O serviço envia informações através de um fluxo de notícias em tempo real;

Sales Cloud e *Sales Service* são os principais componentes do serviço de CRM da Salesforce, eles que auxiliam de fato o suporte aos negócios dos clientes, sendo que componentes como o *Chatter*, são serviços de apoio aos componentes principais, complementando os serviços principais de CRM. A **Tabela 10** permite visualizar algumas características específicas da solução de CRM da Salesforce.com.

Tabela 10 - Características Salesforce.com.

Salesforce.com	
Categoria de Serviço	SaaS
Tipo de Serviço	Automatização de Relacionamento com o Cliente (CRM)
Interface de Acesso do Usuário	Interfaces Web, Linha de Comando
WEB APl's	Force.com API, REST, HTTP, JSON, XML
Arcabouço de programação	Serviços e Intefaces RESTfulWEB,

Fonte: (SALESFORCE, 2013).

O aplicativo atua na camada de SaaS, oferecendo os serviços de apoio aos negócios CRM, podendo ser utilizado através da *web*. Possui uma API

própria (Force.com) que pemite a integração com aplicativos através de métodos HTTP (*delete*, *get*, *put*, *post, custom*) e protocolos REST, JSON, XML (SALESFORCE, 2013).

2.2.8 DRUPAL

O crescimento tecnológico e a expansão da web impulsionou a demanda por sistemas gerenciadores de conteúdo, também conhecidos como CMS (*Content Management System*) (TOSSULINO, 2008). Possuir um portal *web* com interface amigável é imprescindível para que as organizações passem uma boa imagem, os gerenciadores de conteúdo são ferramentas de grande utilidade para auxiliar neste sentido. Dentre os gerenciadores de conteúdo existentes no mercado destaca-se o Drupal, que possui como fator de destaque o fato de ser *open source* (KUMAR, 2012).

O Drupal foi lançado em 2001 pelo belga Dries Buytaert[20], é distribuído sob a licença GNU e desenvolvido na linguagem PHP. A sua função é permitir que um administrador possa gerir um site, desde a organização de conteúdos, de utilizadores, permissões, menus, aspecto gráfico, etc (DRUPAL NO MUNDO, 2008). A **Tabela 11** apresenta mais características do Drupal.

Tabela 11 - Características Drupal.

Drupal	
Categoria de Serviço	SaaS
Tipo de Serviço	Gerenciamento de Conteúdo (CMS)
Interface de Acesso do Usuário	Interfaces Web, Linhas de Comando
WEB API's	Drupal API, HTTP, REST, JSON, XML
Arcabouço de programação	PHP

Fonte: (DRUPAL, 2013), (KUMAR, 2012).

[20] http://buytaert.net/

Por ser desenvolvido em PHP, o Drupal é independente de sistema operacional. No entanto, necessita de um servidor HTTP que seja compatível com PHP, como um servidor Apache e um servidor banco de dados como o MySQL, que são recomendados para seu funcionamento (DRUPAL, 2013). O usuário acessa o Drupal por meio de interfaces *web* e possui diversas API's para manipular seu conteúdo, como métodos HTTP, XML, REST, JSON e a API do próprio Drupal, documentada e disponibilizada em seu site (DRUPAL, 2013).

É um sistema é altamente personalizável, através destas várias API's, de ser *open source* e adaptar-se tanto a pequenos blogs quanto a grandes portais. Ela integra funcionalidades como gerenciamento de intranets, blogs, comunidades/wikis, sites de comércio eletrônico e fóruns numa única ferramenta (TOSSULINO, 2008). Outros *softwares* CSM concorrentes do drupal são: Joomla[21] (também *open source*), WordPress[22], ExpressionEngine[23], entre outras.

2.2.9 COMPARATIVO ENTRE AS PLATAFORMAS

Dadas as características de diversas plataformas, o comparativo entre elas tem objetivo de esclarecer algumas diferenças em pontos em comum definidos previamente. A **Tabela 12** apresenta o comparativo geral das características.

[21] http://www.joomla.com.br/
[22] http://wordpress.com/
[23] http://ellislab.com/expressionengine

Tabela 12 - Comparativo Geral entre as Plataformas.

	Categoria de Serviço	**Tipo de Serviço**	**Interface Usuário**	**API's**	**Arcabouço de Programação**
S3	IaaS	Armazenamento	Linhas de comando, Interface Web	REST, Amazon S3 API	C#, Java, PHP, Perl Python e Ruby
EC2	IaaS	Gerenciamento da Nuvem	Linhas de comando, Interface Web	REST EC2 API	Máquina de Imagem da Amazon baseada em linux customizável
CloudStack	IaaS	Computação em Nuvem (Armazenamento/Rede/Gerenciamento de Máquinas Virtuais,...)	Linhas de comando, Interface Web	CloudStack API, Amazon EC2 e S3 API's	Python, Java, Perl, C, Ruby, Shell Script
OpenNebula	IaaS	Gerenciamento da Nuvem	Linhas de comando, Interface Web	OpenNebula APi, libvirt, EC2, OCCI API, REST	Java, Ruby
OpenStack	IaaS	Computação em Nuvem (Armazenamento/Rede/Gerenciamento de Máquinas Virtuais,...)	Linhas de comando, Interface Web	OpenStack API, S3 API REST, OCCI, HTTP	Python, Shell Script
Google App.	PaaS	Plataforma de Aplicativos	Console de Administração baseado na Web, Linhas de Comando	Google Apps E. API, Java API´s, Python API's	Python, Java
Azure	PaaS	Plataforma de Aplicativos	Portal Microsoft Windows Azure (Web), Linhas de Comando	AZURE API, HTTP/HTTPS, REST, SOAP e XML	Microsoft .NET
Drupal	SaaS	Gerenciamento de Conteúdo (CSM)	Interfaces Web, Linhas de Comando	Drupal API, HTTP, REST, JSON, XML	PHP
Salesforce	SaaS	Automatização de Relacionamento com o cliente (CRM)	Interfaces Web, Linhas de Comando	Force.com API, REST, HTTP, JSON, XML	Serviços e Intefaces RESTfulWEB

A **Tabela 12** permite constatar algumas definições e diferenças em relação aos provedores de nuvem. É possível constatar que cada categoria de serviço pode ser montada e combinada de diferentes modos pelo usuário. A

Figura 2 na seção **2.1.2** ilustra que os serviços IaaS, PaaS e SaaS podem ser combinados de diferentes modos, através da **Tabela 12** é possível ver que cada serviço pode ser formado através da união de componentes de diferentes provedores, ou seja, IaaS, PaaS e SaaS além de poderem ser combinados entre si, podem ser combinados por componentes de diferentes provedores. Isto fornece grande flexibilidade para que uma organização/empresa adapte a nuvem às suas necessidades. Também, apesar de poucas plataformas apresentadas, a **Tabela 12** auxilia na avaliação das diferentes características apresentadas pelos provedores, que por sua vez podem ter grande relevância ao se definir uma plataforma para uso. Por exemplo, ao se combinar componentes de uma categoria de serviço, é necessário verificar todos os requisitos (API's, interfaces, entre outros) necessários para que o serviço seja disponibilizado ao usuário final com a maior transparência possível.

Os serviços de IaaS podem ser utilizados pelo usuário de maneiras distintas, que pode optar por utilizar componentes de diferentes provedores ou utilizar todos componentes de IaaS de um provedor. Por exemplo, utilizar o Amazon S3 para o serviço de armazenamento e o OpenNebula para gerenciamento de máquinas virtuais, aliando a outros componentes de IaaS de preferência do usuário. O mesmo acontece com PaaS e SaaS, pois a nuvem pode ter múltiplas plataformas e também suportar múltiplas aplicações, sendo possível combinar diversas plataformas ao mesmo tempo. Desta maneira, o usuário pode montar um serviço de nuvem se adapte as suas necessidades sem se prender as soluções de um único provedor.

A linguagem de programação em que foram escritas as plataformas é a primeira característica que começa a diferenciar uma plataforma de outra. A maioria das plataformas utiliza as linguagens de alto nível de abstração, como Java, Python e Ruby. Estas linguagens predominam por serem pré-compiladas ou interpretadas, apresentando uma maior portabilidade podendo ser executadas em varias plataformas com poucas modificações. Em geral, a programação fica mais fácil por causa do maior ou menor grau de estruturação de suas linguagens (LINDER, 2007).

Quanto às interfaces do usuário observa-se que todas as plataformas apresentam uma interface *web*, e uma interface através de linhas de comando. A interface *web,* por ser uma opção mais visual, simplifica a personalização dos serviços pelo usuário. Nem todo usuário tem facilidade de executar configurações através de linhas de comando, apesar das sintaxes estarem disponibilizadas nos sites dos provedores. Por exemplo, a configuração de um componente de IaaS é uma tarefa que exige certa complexidade para um usuário leigo, se existe disponibilizada uma interface *web* com todas as opções de modo o usuário consiga obter a configuração desejada, sem ter que inserir linhas de código, torna-se uma opção bem mais simples do que aprender a sintaxe para customizar o serviço. Entretanto, as linhas de comando são necessárias para os usuários mais avançados, elas permitem um controle maior sobre arquivos (e.g., num serviço de armazenamento com uma linha de comando pode-se copiar um arquivo de um diretório para outro), maior velocidade para executar tarefas (utiliza-se somente o teclado), mais facilidade para montar scripts, entre diversos outros benefícios.

É possível observar que todas as plataformas possuem API's próprias, para que os usuários possam utilizar funções e recursos que as plataformas possuem, possibilitando assim, a integração destas com diversos outros componentes de nuvem. Plataformas *open source* como OpenStack e OpenNebula implementam a OCCI API (*Open Cloud Computing Interface*), que é um protocolo e uma API que permite o desenvolvimento de ferramentas de interoperabilidade para tarefas comuns da nuvem, como implantação, ampliação e monitoramento. Trata-se de uma API flexível, com um forte foco em integração, portabilidade, interoperabilidade e inovação (OCCI-WG, 2013). A API REST representa um estilo de arquitetura de *software* para sistemas hipermídia distribuídos, como a *web*. Possui um conjunto de protocolos *web*, como XML, HTTP/HTTPs, JSON entre outros, que permitem descrever o comportamento desejado de um sistema (GARFINKEL, 2007). De maneira geral, as plataformas possuem API`s que disponibilizam a utilização de seus recursos, e API's *web* que permitem a construção de conteúdo para os usuários através de diversos protocolos.

A plataforma Microsoft Azure apresenta um modelo arquitetural que a difere das demais PaaS, sendo oferecida como um serviço que é disponibilizado diretamente dos *data centers* da Microsoft, com regras e políticas centralizadas (MSDN, 2010). Em uma plataforma PaaS tradicional, como a do Google App Engine, os usuários enviam os aplicativos à plataforma para serem então disponibilizados, de uma maneira mais descentralizada. Cada plataforma tem seus benefícios e limitações, cabendo ao usuário optar pela que mais se encaixa dentro de seu perfil e de suas necessidades. Já as plataformas de SaaS apresentadas tem finalidades distintas, a da Salesforce apresenta uma aplicação que auxilia os usuários na automatização de relacionamento com o cliente, a plataforma Drupal tem o objetivo de gerenciar conteúdos *web*, ambas possuem características semelhantes, exceto pelo arcabouço de programação e o fato de uma ser de código proprietário (Salesforce) e a outra *open source* (Drupal).

Através de uma comparação entre as plataformas OpenStack e CloudStack é possível observar que ambas são projetos *open* source, que oferecem orquestração e gerenciamento dos serviços de nuvem, ambas possuem apoio de grandes organizações e empresas, e suporte a API's da Amazon, permitindo a integração com nuvens híbridas que utilizem os componentes da AWS, que é a plataforma mais popular de computação em nuvem (GARTNER, 2012)(BURNS, 2012). Em teoria estas duas plataformas são bastante semelhantes, porém a principal diferença entre elas é o apoio, a aderência do projeto por parte de grandes organizações e empresas, fato que acaba alavancando o desenvolvimento do projeto e o crescimento da comunidade de utilizadores da plataforma. Neste quesito, (PANETTIERI, 2013, *apud* Qingye, 2012) realizou um levantamento de popularidade entre plataformas *open source*, demonstrando que o interesse pela utilização do CloudStack vem crescendo cada vez mais, entretanto ainda segue bem atrás do OpenStack. Os dados mostrados em sua pesquisa são (PANETTIERI, 2013, *apud* Qingye, 2012):

- O OpenStack possui a maior comunidade de utilizadores, seguida por Eucalyptus[24], CloudStack e OpenNebula;

- O OpenStack possui a comunidade mais ativa de utilizadores em seus fóruns de contribuição, listas de e-mail, conferências no último ano (2012), seguida por CloudStack, Eucalyptus e OpenNebula;

- O OpenStack também possui a comunidade mais ativa no último mês (no caso em setembro de 2012), seguido por CloudStack, Eucalyptus e OpenNebula.

Estes dados são importantes para demonstrar que embora as plataformas CloudStack e OpenStack apresentem características similares, o apoio da comunidade de desenvolvedores e colaboradores é um fator fundamental para o progresso da plataforma. Esta vantagem em popularidade ainda gera grande influência para empresas que estão a procura de plataformas de nuvem para disponibilizar seus serviços, e o OpenStack por enquanto esta se destacando como a plataforma *open source* mais popular (PANETTIERI, 2013, *apud* Qingye, 2012).

Para realizar um comparativo mais completo e detalhado das plataformas, é necessário efetuar um levantamento profundo de todas características das plataformas, realizando análises de desempenho, segurança, custos, entre diversos outros fatores relevantes para a adoção de uma plataforma de nuvem. O comparativo realizado teve como objetivo demonstrar que embora algumas plataformas sejam semelhantes possuem algumas características que as diferenciam. A seção **2.2.10** apresenta uma análise mais detalhada da arquitetura da plataforma OpenStack Grizzly, que será o objeto da análise deste trabalho.

[24] http://www.eucalyptus.com/

2.2.10 ARQUITETURA OPENSTACK GRIZZLY

Nesta subseção apresenta-se a arquitetura da plataforma OpenStack para compreender os componentes que compõem a plataforma e a relação entre os mesmos. A subseção está dividida em oito partes, a primeira visa explicar as diferenças do último *release* Folsom para o atual Grizzly e a necessidade da atualização da plataforma-alvo para a realização da análise, fazendo uma breve introdução da plataforma e dos componentes, e as demais partes apresentam cada componente da plataforma.

2.2.10.1 FOLSOM VS GRIZZLY

Para realizar a análise e uma proposta de melhoria de segurança mais efetiva no OpenStack é necessário avaliar o *release* mais atual da plataforma, evitando pesquisar algo que eventualmente já foi corrigido. Neste sentido, este trabalho utiliza o *release* Grizzly ao invés do Folsom para minimizar um eventual retrabalho em problemas relatados no Folsom, simultaneamente é necessário evidenciar as diferenças e pontos em que foram melhorados/corrigidos neste *release* em relação ao Folsom. Esta compreensão dos componentes que fazem parte do projeto e suas relações são aspectos fundamentais para a realização a proposta de uma melhoria na plataforma, assim, esta subseção apresenta a arquitetura conceitual e lógica do Grizzly e as principais mudanças para o *release anterior*.

O projeto como inteiro foi concebido para entregar ao usuário um serviço de nuvem massivamente escalável (OPENSTACK, 2013), este é o objetivo do OpenStack como provedor de IaaS. Para alcançar este objetivo cada componente que constitui a plataforma foi projetado para trabalhar em sinergia com os outros componentes da nuvem para fornecer um serviço completo de IaaS. Essa integração é facilitada através das interfaces públicas (API's), em que cada componente pode oferecer seus serviços e também requisitar os

serviços dos outros componentes através das mesmas interfaces (PEPPLE, 2012).

A plataforma é composta pelos mesmos sete componentes do Folsom, denominados: **Swift** (armazenamento de objetos), **Horizon** (painel ou *dashboard*), **Cinder** (armazenamento de blocos), **Glance** (serviço de imagens), **Nova** (gerenciamento da nuvem), **Neutron** (rede) e **Keystone** (identidade) (OPENSTACK, 2013). A **Tabela 13** possui uma breve descrição dos serviços proporcionados por cada um destes componentes.

Tabela 13 - Componentes do OpenStack.

Componente	Descrição
Swift (armazenamento)	Permite armazenar e recuperar arquivos que não estejam montados em um diretório. É um sistema de armazenamento a longo prazo para os dados mais permanentes ou estáticos.
Horizon (painel ou *dashboard*)	Prove uma interface *web* modular para todos os componentes do OpenStack. Com esta interface *web* visual é possível gerenciar mais facilmente os componentes da nuvem.
Nova (gerenciamento da nuvem)	É o *software* que controla a infraestrutura de IaaS, alocando ou liberando recursos computacionais, como rede, autenticação, entre outros.
Glance (serviço de imagens)	Fornece um catálogo e repositório para imagens de disco virtuais. Estas imagens são mais utilizadas no Nova, que gerencia estas imagens.
Cinder (armazenamento de blocos)	Fornece um armazenamento de blocos para as máquinas virtuais
Neutron (antigo Quantum - serviço de rede)	Provisiona serviços de rede para os componentes que são gerenciados pelo Nova. Permite que os usuários criem e anexe sua interface a ele, que possui arquitetura "plugável" própria para isto.
Keystone (serviço de identidade)	Fornece autenticação e autorização para todos os serviços do OpenStack.

Fonte: (OPENSTACK, 2013), (PEPPLE, 2012).

Por meio do *changelog* disponibilizado em (OPENSTACK NOTES, 2013) no site do OpenStack é possível observar algumas das principais melhorias realizadas na plataforma para este novo *release*. A maioria das modificações na plataforma são pequenas melhorias nos componentes, não alterando a arquitetura lógica dos componentes. Para observar claramente as modificações realizadas neste *release*, apresenta-se uma tabela com as melhorias realizadas para cada componente. Iniciando pela mesma ordem em que foram apresentados os componentes na **Tabela 13**, apresenta-se a **Tabela 14** destacando algumas melhorias no Swift.

Tabela 14 – Melhorias no Swift

Pontos Modificados	Folsom	Grizzly
Prevenção de crescimento no uso do disco	Arquivos com extensão tombstone (.ts) antigos podem inserir e atualizar arquivos, causando o preenchimento do disco	Desabilitadas as requisições HTTP: PUT/POST e DELETE para arquivos .ts antigos (intervalo pré-definido na configuração).
Suporte para *upload* e *download* de grandes objetos	Objetos enviados são baixados individualmente.	Método SLO (Static Large Object) permite que o usuário envie vários objetos ao mesmo tempo, e também possa baixar estes objetos como um arquivo único.
Exclusão da própria conta	Permite que usuários possam excluir a própria conta e quotas definidas no Keystone.	Somente usuário autorizado pode efetuar o gerenciamento de contas e quotas no Keystone para o Swift
Bulk de operações		Suporte para Bulk de exclusão de arquivos (exclusão de diversos arquivos ao mesmo tempo).
Replicação de dados	Oferece suporte desde o primeiro *release.*	Introdução do conceito de Regiões.Neste *release* houve melhorias para que os dados possam ser replicados para dispositivos específicos.

Fonte: (OPENSTACK NOTES, 2013).

Nesta última versão do componente de armazenamento (Swift 1.9.1), foram desabilitadas as requisições HTTP: PUT, POST e DELETE por meio de

arquivos *tombstone* antigos, prevenindo a possibilidade de preenchimento desnecessário do disco. No Keystone, não é mais permitido que usuários possam excluir suas próprias contas e quotas, neste esta função fica a cargo de um usuário com papel de administrador. Permite requisições *bulk*, para a execução de diversas operações simultaneamente, inclusão do suporte ao *download* de diversos objetos como um arquivo único e diversas outras melhorias no componente. A **Tabela 15** apresenta as principais melhorias identificadas no Horizon.

Tabela 15 – Melhorias no Horizon

Pontos Modificados	Folsom	Grizzly
***Upload* de imagens diretamente pelo Horizon**	Utiliza-se a interface de linha de comando do Glance para *upload* de imagens	*Upload* de imagens direto pelo Horizon
Melhorias na usabilidade do *front-end*		Reorganização de menus, seção de regras de segurança para grupos, ícones, entre outras melhorias.
Suporte para tokens PKI na autenticação	Tokens UUID	Tokens PKI
Melhorias de Rede		Melhor compatibilidade com as API's de rede do Nova garantindo melhor balanço de carga, infográficos de topologia de rede
Outras melhorias	- Senhas são validadas no lado do servidor; - *Logouts* não eliminam a *token* associada, gerando uma vulnerabilidade.	- Validação de senhas no lado do cliente; - *Logouts* agora excluem a *token* associada com a seção corrente para evitar ataques do tipo *replay;* - Grupos de segurança podem ser inseridos em uma instância em execução; e -Diversas melhorias para compatibilidade e estabilidade nos *browsers*.

Fonte: (OPENSTACK NOTES, 2013).

As principais modificações para o Horizon baseiam-se em melhorias para a utilização do usuário, proporcionando facilidade de uso, melhorias na estabilidade, gerenciamento de instâncias e imagens. Também, houveram boas melhorias de rede neste componente, no balanceamento de carga, adição de infográficos da topologia de rede, melhor compatibilidade com o Nova. Houve melhorias de usabilidade na opção de edição dos grupos de segurança, que era complicado devido ao número de opções possíveis e os termos técnicos envolvidos, ainda na questão de segurança esta versão do Horizon fornece suporte para as tokens PKI do Keystone. A **Tabela 16** apresenta as modificações no componente gerenciador da nuvem, o Nova.

Tabela 16 – Melhorias no Nova

Pontos Modificados	Folsom	Grizzly
Células (experimental)		Esta funcionalidade permite uma nova maneira de se escalabilizar os recursos da nuvem. Em que cada célula (máquina no cluster) podem estar em localizações geográficas diferentes (DOCS OPENSTACK, 2013).
Zonas de Disponibilidade	Configuradas diretamente em um arquivo de configuração	É possível definir zonas de disponibilidade por meio da API do Nova.
API's específicas para admins		Inclusão de uma API específica para que o papel de admin possa executar tarefas administrativas.
Adição de para um grupo de segurança	As regras padrão pré-definidas.	Possibilidade de adição de regras de segurança específicas nas regras pré-definida ao se criar um projeto.
Outras melhorias		- É possível definir quotas para alocação de IP's fixos; - Melhorias no desempenho com a conexão com o banco MySQL, provendo melhor interação com este banco; - Melhorias no *driver* de compatibilidade com o *hypervisor* VMWare; - Entre outras.

Fonte: (OPENSTACK NOTES, 2013).

As alterações exibidas na **Tabela 16** incluem, API's específicas para admins com a declaração de diversos novos métodos para a execução de funções administrativas na nuvem, a disponibilidade por zona foi aprimorada nesta versão, na anterior uma zona de disponibilidade para determinado nó era configurável somente via arquivo, na versão atual é possível configurar e listar zonas de disponibilidade de um determinado nó via API. No que diz respeito à segurança, o Nova agora pode determinar regras padrão para um grupo de segurança quando um *tenant* (container usado para agrupar ou isolar recursos) é criado.

Para o serviço de imagens, o Glance, não são descritas muitas alterações no componente. Houve uma mudança quanto ao compartilhamento de imagem do componente e em sua API, alterando a maneira com que outros componentes se relacionam com o Glance, como o Nova e o Horizon. Por exemplo, o Nova pode copiar diretamente uma imagem do Glance utilizando-se da nova API e o Horizon pode diretamente fazer o *upload* de uma imagem. A **Tabela 17** apresenta as melhorias identificadas para o Cinder.

O Cinder, responsável pelo armazenamento de blocos da plataforma ganhou diversas melhorias técnicas, como suporte a novos *drivers*, múltiplos *backends* no mesmo gerenciador. O componente responsável pelo serviço de rede, o Neutron chamado de Quantum no Folsom, ganhou melhorias por meio da simplificação dos requisitos de configuração de rede, eliminando obstáculos em sua implantação. No quesito de segurança, os grupos de segurança permitem filtro de pacotes para políticas de segurança voltadas para proteção de VM's. E diversas outras melhorias como suporte a IPv6, API XML entre outras. A **Tabela 14** apresenta algumas das melhorias realizadas no Keystone.

Tabela 14 **– Melhorias no Keystone para o Grizzly.**

Pontos Modificados	Folsom	Grizzly
API	**Keystone API Versão 2.0**	**Keystone API Versão 3.0**
	- Conceito de Tenants; -Token é exposta na URL; e - Suporte a PKI somente na autenticação.	- Tenants passaram a ser denominados de projetos; - ID da token removidos da URL; - Papéis podem ser concedidos a níveis de domínios ou em níveis de projeto; e - Entre outras melhorias.
Grupos de Usuários		Gerenciamento de papéis para grupos de usuários (aplicável a API v2.0 e v3.0).
Token	Utiliza o conceito de UUID (Universally Unique Identifier)	Utiliza o conceito de PKI (Public Key Infrastructure)
Domínios	Falta de um mecanismo para dar escopo a usuários em um projeto ou grupo, papéis dentro de um grupo.	Definidos containers para projetos, usuários e grupos fornecendo isolamento e um nível a mais de papel de gerenciamento (aplicável a API v2.0 e v3.0)..
Confiança		Delegação de papéis para projetos específicos (disponível somente na API v3).
Políticas	Cada componente possuía seu arquivo de políticas de maneira descentralizada.	Repositório centralizado de políticas (disponível somente na API v3).
RBAC	Não apoia RBAC.	O arquivo de políticas (policy.json) é forçado para todas chamadas de API.
Autenticação externa		Permite autenticação externa via autenticação *CGI (Common Gateway Interface) – Remote User.*

Fonte: (OPENSTACK, 2013).

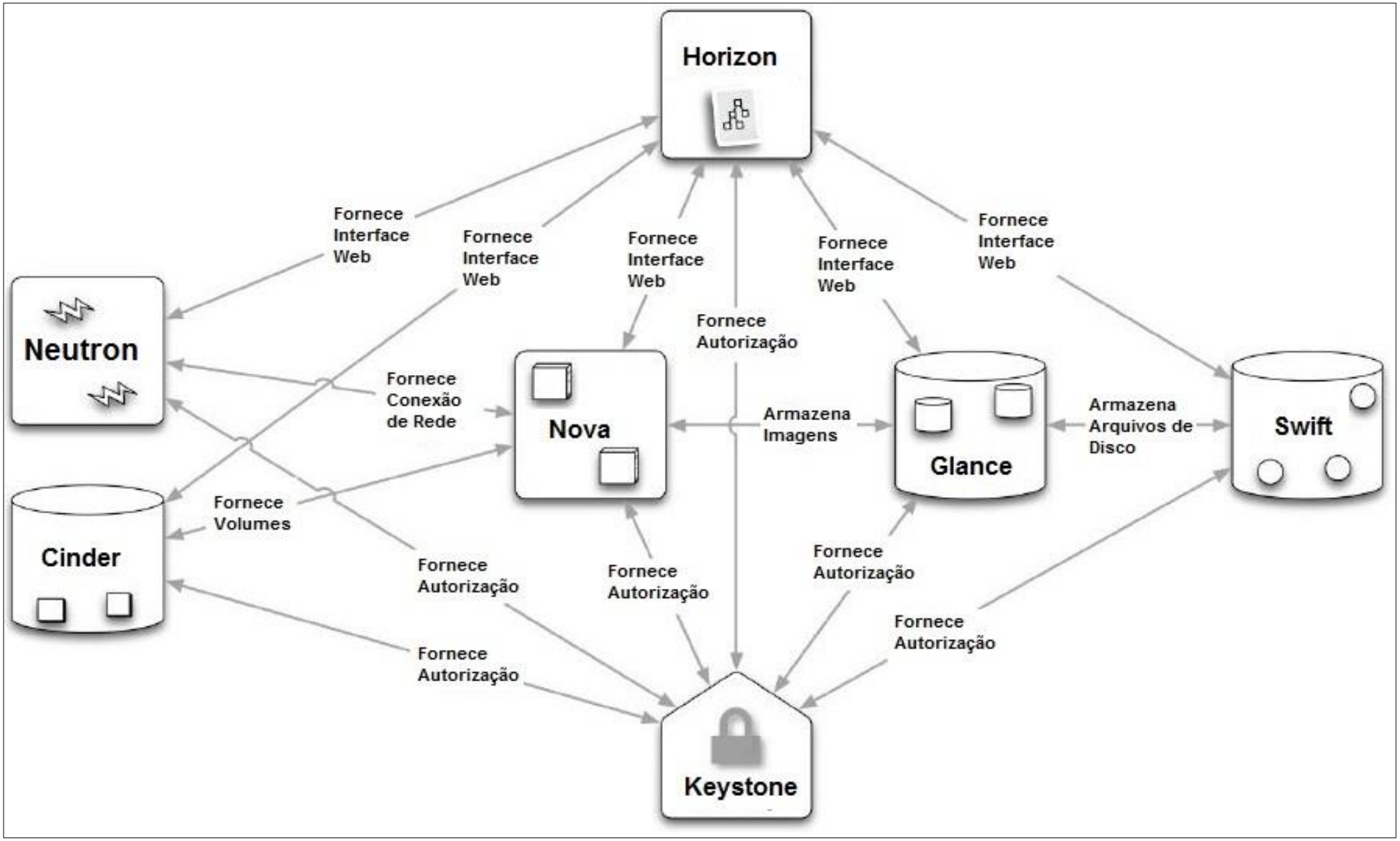
Algumas mudanças importantes foram identificadas para o Keystone no Grizzly, as *tokens* utilizam a infraestrutura de chaves públicas (PKI) e podem ser validadas *off-line* fornecendo mais confiabilidade na segurança da *token* em

relação ao modelo tradicional de *tokens* baseadas em UUID (identificador único universal). Não foram realizadas grandes mudanças na arquitetura lógica do Keystone, o funcionamento é basicamente o mesmo do último *release* (Keystone v2). Como pequena mudança destaca-se a centralização das políticas de acesso no Keystone, que anteriormente eram localizadas (arquivo policy.json) em cada componente da plataforma separadamente. Muitas destas melhorias descritas são suportadas somente na nova API v3 do Keystone, tornando necessária uma atualização do componente para que estas tenham efeito.

Nota-se no *changelog* que foram realizadas diversas alterações técnicas nas plataformas, fazendo com que seja necessário um aprofundamento bem maior para compreender o real impacto destas melhorias na plataforma. Também foi possível notar que ocorreram diversas melhorias focadas em segurança, não só no Keystone, mas também em diversos outros componentes, deflagrando a importância e preocupação que os responsáveis pelo desenvolvimento da plataforma possuem com este tema.

Além destes componentes, também existem uma série de outros projetos em desenvolvimento (Ceilometer – medição, Oslo – Bibliotecas) que podem ser incluídos em um futuro *release* do OpenStack (denominado Havana), foi o caso dos projetos Cinder e Quantum que foram adicionados ao projeto OpenStack no *release* anterior, o Folsom. Todos estes componentes atuando em sinergia fornecem a infraestrutura necessária para o consumidor disponibilizar seus serviços, a arquitetura resultante do relacionamento destes componentes pode ser observada na **Figura 10**.

Figura 10 - Arquitetura Conceitual OpenStack Grizzly. Adaptada de: (PEPPLE, 2012).

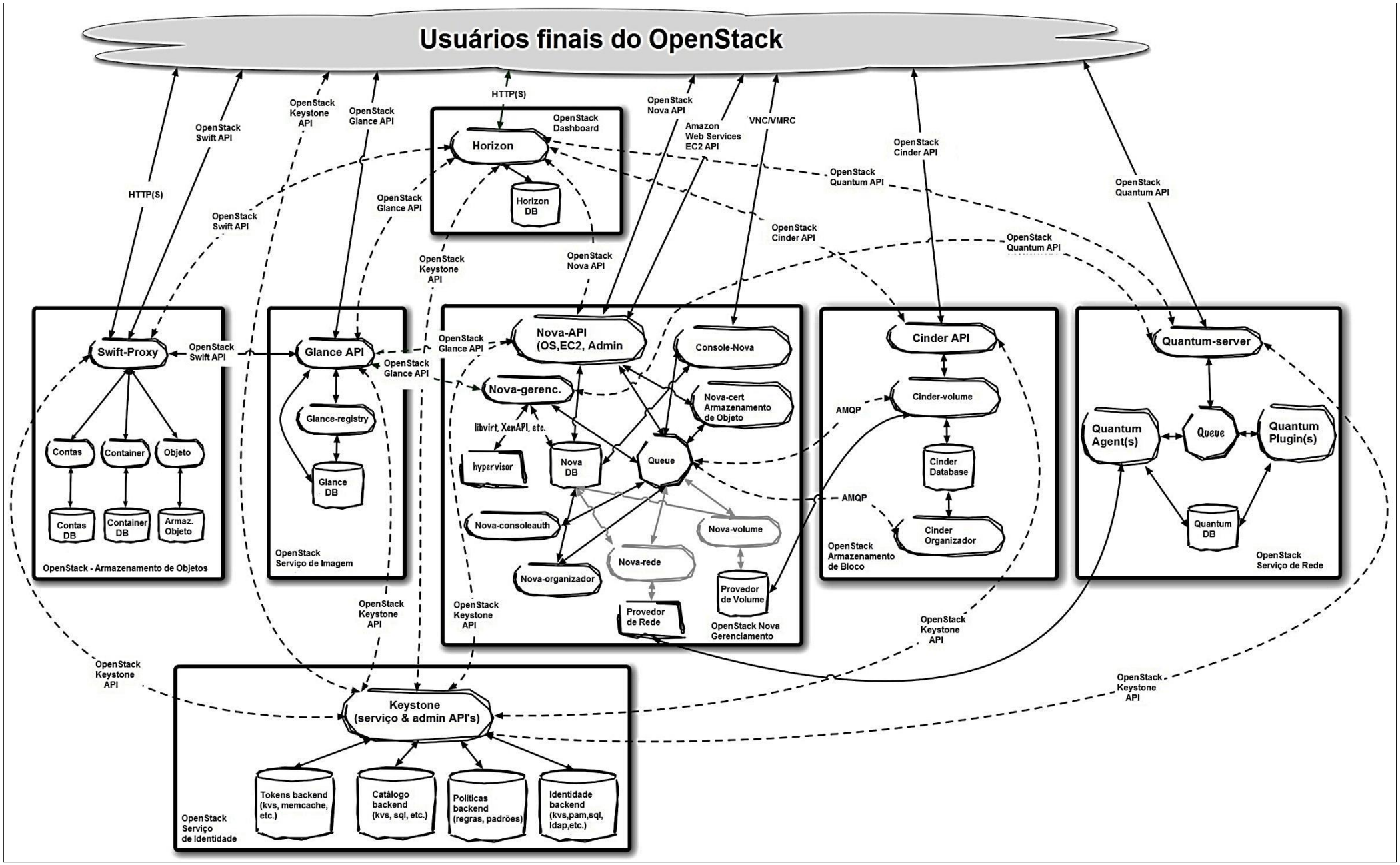


A **Figura 10** ilustra como estes componentes estão dispostos na arquitetura do OpenStack Grizzly. Observa-se que o *dashboard* Horizon fornece interface *web front-end* para todos componentes, coletando a entrada de várias formas do usuário para os componentes. O Nova pode armazenar e recuperar imagens no Glance, que é um repositório de imagens que por sua vez pode armazenar arquivos de disco virtual no Swift, que se encarrega do armazenamento de blocos. O Quantum fornece os serviços de rede para o Nova e por tabela todos componentes que são gerenciados pelo Nova. O Cinder fornece volumes de armazenamento para o Nova.

Esta é uma visão simplificada da arquitetura do OpenStack Grizzly na sua configuração mais comum, assumindo que o usuário utilize todos os componentes fornecidos pelo OpenStack Grizzly, uma vez que os componentes da nuvem podem ser montados e utilizados de acordo com as necessidades do usuário, ilustrado na **Figura 2.** (PEPPLE, 2012) destaca que a **Figura 10** demonstra apenas o lado operacional da nuvem, não demonstrando como os usuários da nuvem podem utilizá-la em uma situação real. Por exemplo, diversos usuários podem acessar massivamente e diretamente o serviço de armazenamento de objetos, sobrecarregando o componente.

A arquitetura lógica é um pouco mais complicada de visualizar que a arquitetura conceitual, pois apresenta mais detalhes a respeito das relações entre os componentes, expondo claramente o ambiente interno de cada um dos componentes que integra a arquitetura e como ocorre a requisição de serviço entre os componentes. A **Figura 11** ilustra esta arquitetura lógica, novamente com base na configuração arquitetural mais comum do OpenStack Grizzly.

Figura 11 - Arquitetura Lógica OpenStack Grizzly. Adaptada de: (PEPPLE, 2012).



A **Figura 11** permite observar que a arquitetura lógica é bem semelhante a arquitetura conceitual, entretanto fornece mais detalhes sobre como isto ocorre. Constata-se que os usuários finais do OpenStack podem interagir com a nuvem através da interface *web* fornecida pelo Horizon, por intermédio de uma conexão HTTPS, para prover o gerenciamento dos componentes da nuvem, também há a opção dos usuários finais comunicarem-se diretamente com os componentes através de suas API's públicas. Todos serviços são autenticados por meio do Keystone, que possui quatro bases de dados: políticas e restrições, *tokens* de acesso, identidade e de catálogos, que em conjunto fornecem as informações necessárias para prover a autenticação dos componentes na nuvem. Os componentes interagem entre si individualmente através de suas API's públicas (exceto quando permissões de administrador são requeridas) ou também diretamente através de interfaces disponibilizadas para o usuário final. Através da descrição de cada um dos componentes representados na **Figura 11**, é possível compreender detalhadamente o papel de cada um deles desempenha na arquitetura do OpenStack.

2.2.10.2 OPENSTACK NOVA

O Nova é o componente mais distribuído e complicado do OpenStack, pois um grade número de processos tem que cooperar para transformar as requisições API dos usuários finais em máquinas virtuais (ABDUL, 2012). Os processos internos do Nova observados na **Figura 11** são (OPENSTACK, 2013b):

- **Nova-API**: aceita e responde as chamadas API do usuário final, suporta a API do Amazon EC2, uma API especial para que os administradores possam realizar as ações administrativas e a própria API;

- **Nova- gerenciador**: é um módulo composto por *daemons*[25] que lidam com o ciclo de vida das instâncias das máquinas virtuais. Em uma implementação do OpenStack existem vários *daemons*, uma instância é criada por qualquer um deles, dependendo do tipo de escalonamento utilizado (PEPPLE, 2012). Eles recebem as requisições do *Message Queue*;

- **Nova-organizador**: responsável pelo escalonamento da nuvem, ele mapeia as chamadas feitas às APIs aos componentes apropriados, disponíveis para ele através de um pool de recursos disponíveis. O escalonamento é feito baseado no tipo de algoritmo de escalonamento configurado;

- **Rabbit MQ Server** – É o *Message Queue* do OpenStack, é através deste módulo que os componentes do OpenStack se comunicam. O Nova usa chamadas assíncronas para as requisições, com um mecanismo de *call-back* que é acionado quando a resposta é recebida. Uma vez que a comunicação é assíncrona, nenhum dos componentes ficam bloqueados por muito tempo em um estado de espera. Isto é especialmente importante dado que muitas ações invocadas através das APIs envolvem tempos longos, como criar uma instância ou fazer o upload de uma imagem;

- **Nova-rede**: é o controlador de rede da nuvem, ele é responsável por distribuir os IPs para as máquinas virtuais, lida com a configuração dos nodos, implementa os grupos de segurança, e faz a comunicação entre o controlador e as máquinas virtuais;

- **Nova-volume:** responsável pelo gerenciamento dos volumes do tipo LVM, que são os volumes de armazenamento das instâncias, todo o processo de se criar, excluir, adicionar ou remover uma LVM (*Logical Volume Manager*) a uma instância é realizado por esse módulo.

---

[25] *Disk And Execution MONitor (Daemon).*São monitores de execução e de disco. http://www.linfo.org/daemon.html

O Nova não inclui software de virtualização, mas define drivers que interagem com mecanismos de virtualização (libvirt, XenAPI, entre outros) executados no sistema operacional hospedado e expõe as funcionalidades em uma API para web (QIAO et. al., 2012). É o componente mais importante da arquitetura, pois efetua todo o gerenciamento dos recursos da nuvem.

2.2.10.3 OPENSTACK GLANCE

Fornece serviços para descobrir, registrar e recuperar imagens de máquinas virtuais. Os clientes podem registrar novas imagens de disco virtual, consultar a API para obter informações sobre a disponibilidade de disco e usar a biblioteca do cliente para transmissão de imagens de disco virtual (PEPPLE, 2012). Também suporta uma grande variedade de formatos de imagens, incluindo VirtualBox[26], Microsoft Hyper-V[27], KVM[28], VMware[29], XEN[30], entre outros. Seus componentes são (OPENSTACK, 2013c):

- **Glance-API**: aceita chamadas de API de imagens para procura de imagens, armazenamento de imagens e recuperação de imagens;

- **Glance-registry**: armazena, processa e recupera *metadata* das imagens (tamanho, tipo, entre outras informações...);

- **Database**: para armazenar de fato os arquivos das imagens.

O Glance atua principalmente no suporte ao Nova, recuperando e armazenando as imagens de acordo com a demanda imposta pelo Nova, através da sua API RESTful que é capaz de descobrir, recuperar e listar

[26] https://www.virtualbox.org/
[27] https://www.microsoft.com/brasil/hyper-v-server/
[28] http://www.linux-kvm.org/
[29] http://www.vmware.com/products/vsphere-hypervisor/overview.html
[30] http://www.xen.org/products/xenhyp.html

metadados de imagens de máquinas virtuais (WEN et. al, 2012). Possui acesso direto aos componentes: Horizon que prove o gerenciamento do Glance por meio de interface *web* para os usuários finais; Swift para recuperar e armazenar imagens; Keystone para autenticação e autorização.

2.2.10.4 OPENSTACK SWIFT

Equivalente ao serviço S3 da Amazon (*Simple Storage Service*), o Swift implementa um sistema de armazenamento de objetos provendo redundância e tolerância a falhas, ele é extremamente escalável tanto em termos de tamanho (vários petabytes) como capacidade (bilhões de objetos). Ele é formado pelos seguintes componentes (OPENSTACK, 2013d) (SLIPETSKYY, 2011) (PEPPLE, 2012):

- **Swift Proxy**: os clientes interagem com o Swift através desse sub-componente, o *proxy* responde as requisições feitas via OpenStack API do Swift ou HTTP. Aceita arquivos para *upload*, modificações em metadados ou criação de novos containers. O Proxy Server também lida com falhas das entidades;

- **Swift Objeto:** servidor de objetos da nuvem, sua função é lidar com o armazenamento, recuperação e remoção de objetos. Objetos são tipicamente arquivos binários, armazenados no sistema de arquivos;

- **Swift Container:** é um compartimento para armazenar dados, contém a lista dos objetos de um *Object Server*. As listas são mantidas em bancos de dados. O Container Server também mantém estatístico como o número de objetos armazenados e o tamanho do espaço de armazenamento ocupado por um *Object Server*;

- **Swift Contas:** O conceito de conta é relativo às permissões de acesso às entidades presentes no sistema de arquivos Swift. Existe um servidor

de contas, semelhante ao servidor de containers, mas que lida com a listagem de containers de acordo com as permissões de acesso de cada conta.

A fim de evitar um ponto único de falha, o Swift implementa o gerenciamento de maneira distribuída (QIAO et. al, 2012). Por conta disso, um nó Swift não pode ser "montado" como um sistema de arquivos. Apesar disso, a API OpenStack fornece o acesso aos arquivos de maneira conveniente. O Swift é equivalente ao serviço de armazenamento S3 da Amazon e pode armazenar diversos objetos que estão distribuídos.

2.2.10.5 OPENSTACK HORIZON

É uma aplicação *web* que fornece uma interface de acesso aos componentes do OpenStack aos usuários finais e administradores (PEPPLE, 2012). Para executar o Horizon, ele deve ter acesso aos servidores no quais os serviços Nova, Keystone e Glance estejam sendo executados como requisitos mínimos, os módulos Swift e Quantum são opcionais (QIAO et. al., 2012). Seus componentes são (OPENSTACK, 2013e):

- **Horizon:** suporta diferentes navegadores *web* para que os usuários possam iniciar instâncias dos componentes do OpenStack. Algumas de suas características são: demonstração de quotas dinâmicas suporte a fuso-horário, tratamento de erros, entre outros;

- **Database**: armazena basicamente dados dos outros componentes iniciados pelo usuário, armazenando poucos dados do próprio componente.

Desenvolvida usando *Django*[31], o projeto do Horizon permite a adição de *plugins* que implementem novas funcionalidades, como um sistema de

[31] https://www.djangoproject.com/

monetarização[32], por exemplo, além de ser personalizável (QIAO et. al., 2012). Por exemplo, é possível trocar o logo para um logo pessoal, ou da organização que está utilizando o sistema, entre outras personalizações para o painel.

2.2.10.6 OPENSTACK NEUTRON (QUANTUM NO FOLSOM)

O Quantum fornece a conexão de rede como um serviço entre a interface de dispositivos que são gerenciados por outros serviços (normalmente o Nova realiza o gerenciamento de outros componentes). Os serviços funcionam permitindo que usuários criem suas próprias redes e anexem suas interfaces ao Quantum. É altamente configurável devido a sua arquitetura conectável, que permite acomodar diferentes equipamentos e *softwares* de rede. Seus componentes são (OPENSTACK, 2013f):

- **Quantum-server**: aceita as requisições API e as encaminha para o *plugin* adequado para a ação;

- **Quantum-plugins/agentes**: realizam ações como conectar e desconectar portas, criar redes ou sub-redes e endereçamento IP. Os *plugins* podem variar de acordo com o fornecedor e as tecnologias utilizadas na nuvem.

O Quantum permite a criação de redes de alta complexidade, se aproveitando da capacidade das máquinas virtuais criarem suas próprias redes virtuais dentro do projeto no qual estão inseridas (PEPPLE, 2012). Interfaces virtuais de diferentes projetos podem se conectar de diversas maneiras, gerando diversas possibilidades de utilização. O Quantum também conta com um *framework* extensível, permitindo a implementação de novos serviços como detecção de intrusos, balanceamento de carga, VPN (*Virtual Private Network*) e *firewall* (QIAO et. al., 2012). O objetivo do Quantum é tornar possível a criação

[32] http://openstackgd.wordpress.com/2012/01/24/billing-plugin-for-horizon/

de diversas redes através de abstrações de redes, subredes, portas, e diversos *plugins* para dar suporte às necessidades do usuário.

2.2.10.7 OPENSTACK CINDER

Fornece dispositivos de armazenamento em nível de bloco para uso das instâncias de máquinas virtuais criadas pelo Nova. O Cinder gerencia a criação, anexação e liberação dos dispositivos de bloco para os servidores (PEPPLE, 2012). Os volumes de armazenamento de bloco são integrados com o Nova e o Horizon, permitindo que o usuário possa gerir suas próprias necessidades de armazenamento. Os seus componentes são (OPENSTACK, 2013a):

- **Cinder-API**: aceita requisições API e as encaminha para o Cinder-volume para tomar a devida ação;

- **Cinder-volume**: age de acordo com as requisições escrevendo ou lendo na base de arquivos do Cinder (*Database*), para manter um determinado estado, interagir com outros processos (como o Cinder-organizador) através de mensagens Queue;

- **Cinder-organizador**: é composto por monitores de execução de disco que tem a função de selecionar o nó ideal no provedor de armazenamento de bloco para criar o volume;

- **Database**: armazena os blocos de informação.

Este módulo surgiu como uma parte do módulo nova, o *nova-volume block service*. Um bloco de disco criado pelo Cinder é um disco sem partição e formatação, cabe a instância que o utiliza formatá-lo e particioná-lo na primeira vez em que for utilizado. Suas principais funcionalidades são (QIAO et. al., 2012): criação de deleção de volumes de disco; criação e deleção de estados

de disco (*snapshot*); criação de um volume a partir de um estado de disco; e recuperação de metadados do volume.

O Cinder pode realizar dois tipos de armazenamento (QIAO et. al., 2012): o armazenamento efêmero, que se refere a uma instância na qual o sistema de arquivos de uma imagem de máquina virtual é montada; e armazenamento de volume, que é o armazenamento persistente e independente de instâncias (OPENSTACK, 2013a). Os volumes podem ser de qualquer tamanho, basta apenas escolher a formatação adequada àquela partição.

2.2.10.8 OPENSTACK KEYSTONE

O Keystone é o componente de mais relevância para a segurança do OpenStack, pois ele é responsável por validar as credenciais e conceder acesso a todos usuários (humanos ou sistemas) da nuvem. O Keystone é um projeto que provê um único ponto de integração para geração e gerenciamento de identidades, *tokens*, catálogos e políticas para os serviços do OpenStack, ou seja toda a segurança da plataforma passa por este componente. Seus componentes são (OPENSTACK, 2013g) (PEPPLE, 2012):

- **Keystone**: trata as requisições da API como também fornece serviços configuráveis de: catálogo, políticas, *tokens* e identidade;

- **Databases (*tokens*, políticas, identidade, catálogos):** cada componente do Keystone tem um *backend* "plugável" que permite diferentes modos de uso do serviço. Permite também que a partir destes pontos de conectividade possam ser implementados e utilizados mecanismos de autenticação próprios.

O Keystone e suas databases (*tokens*, políticas, identidade, catálogos) ilustrados na **Figura 11**, possuem alguns conceitos que são fundamentais para

a compreensão do funcionamento do componente, sendo eles: (ABDUL, 2012, *apud* OPENSTACK, 2013g):

- **Usuário**: representa uma pessoa, sistema, ou serviço que utiliza os serviços de nuvem do OpenStack. Os usuários devem possuir uma credencial, e o Keystone pode atribuir *tokens* para que este usuário acesse os serviços. Os usuários podem ser diretamente atribuídos a estes *tenants* se comportam como se estivessem contidos neste *tenants*;

- **Credencial**: dados que pertencem e são conhecidos somente pelo usuário, este que pode apresentá-los para provar sua identidade (desde que sejam conhecidos somente pelo usuário);

- **Autenticação**: no contexto do Keystone é o ato de confirmar a identidade do usuário ou a verdade de uma afirmação. O Keystone recebe as requisições que são feitas e as validam para confirmar a autenticidade dos usuários;

- ***Token***: *bit* arbitrário de texto que é usado para acessar os recursos. Cada *token* possui um escopo que descreve a quais recursos ela pode acessar. Uma *token* pode ser revogada a qualquer momento e possui um tempo finito para uso;

- ***Tenant***: um container usado para agrupar ou isolar recursos e/ou objetos de identidade. Pode possuir recursos computacionais no Nova ou recursos de armazenamento no Swift;

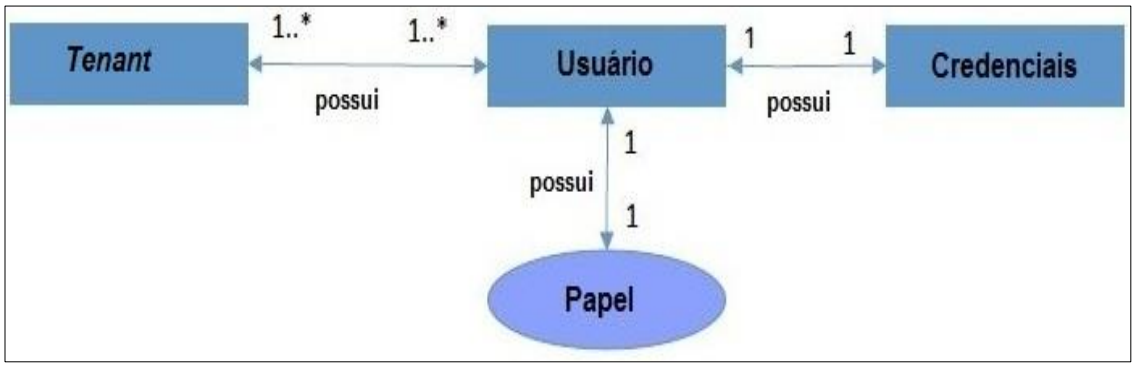
- **Papel**: personalidade que um usuário assume quando está desempenhando um conjunto específico de operações. Um papel inclui um conjunto de direitos e privilégios, e um usuário ao assumir determinado papel ganha estes direitos e privilégios;

- **Serviço**: um serviço do OpenStack, que pode ser o Nova (gerenciamento), Swift (armazenamento) ou o Glance (imagem). Um serviço fornece um ou mais *endpoints* pelo qual os usuários podem acessar recursos e realizar operações;

- ***Endpoints***: um endereço acessível de rede, geralmente descrito por URL (*Uniform Resource Locator*), no qual um serviço pode ser acessado.

A partir da definição dos conceitos básicos que fazem parte do contexto do Keystone, apresentam-se dois diagramas de componentes na **Figura 12** e **Figura 13** que auxiliam na compreensão de como ocorrem as operações internas do Keystone.

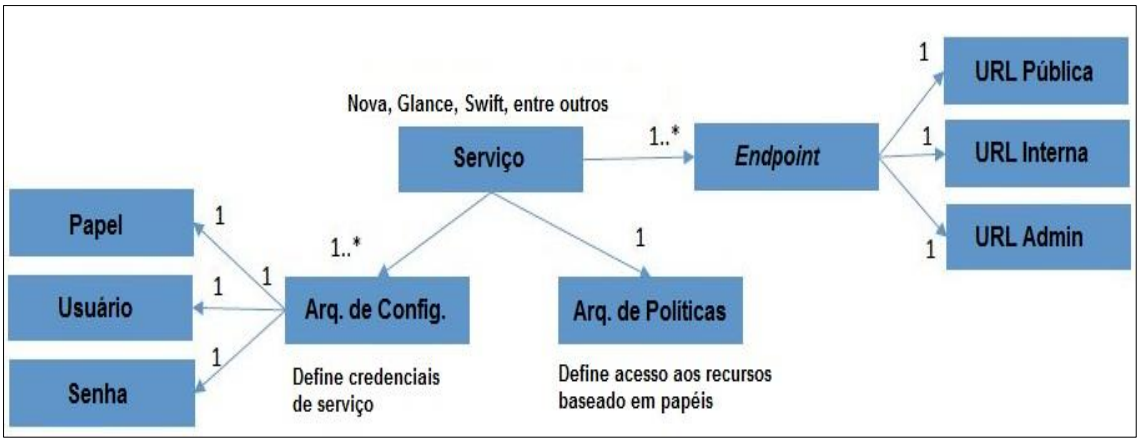
Figura 12 – Diagrama 1 de Componentes do Keystone.



Adaptado de: (ABDUL, 2012).

A **Figura 13** ilustra a relação entre projetos, usuários, papéis e credenciais. Um *tenant* pode ser constituído de diversos usuários, e um usuário pode fazer parte de vários *tenants*. Este usuário pode obter somente um papel e uma credencial para sua identificação. A **Figura 13** ilustra a relação da entidade de serviços (que pode ser o Nova, Swift, entre outros) com as *databases*de configuração (que concede as credenciais aos usuários) e de políticas (define o que o papel pode realizar em um determinado serviço).

Figura 13 – Diagrama 2 de Componentes do Keystone.

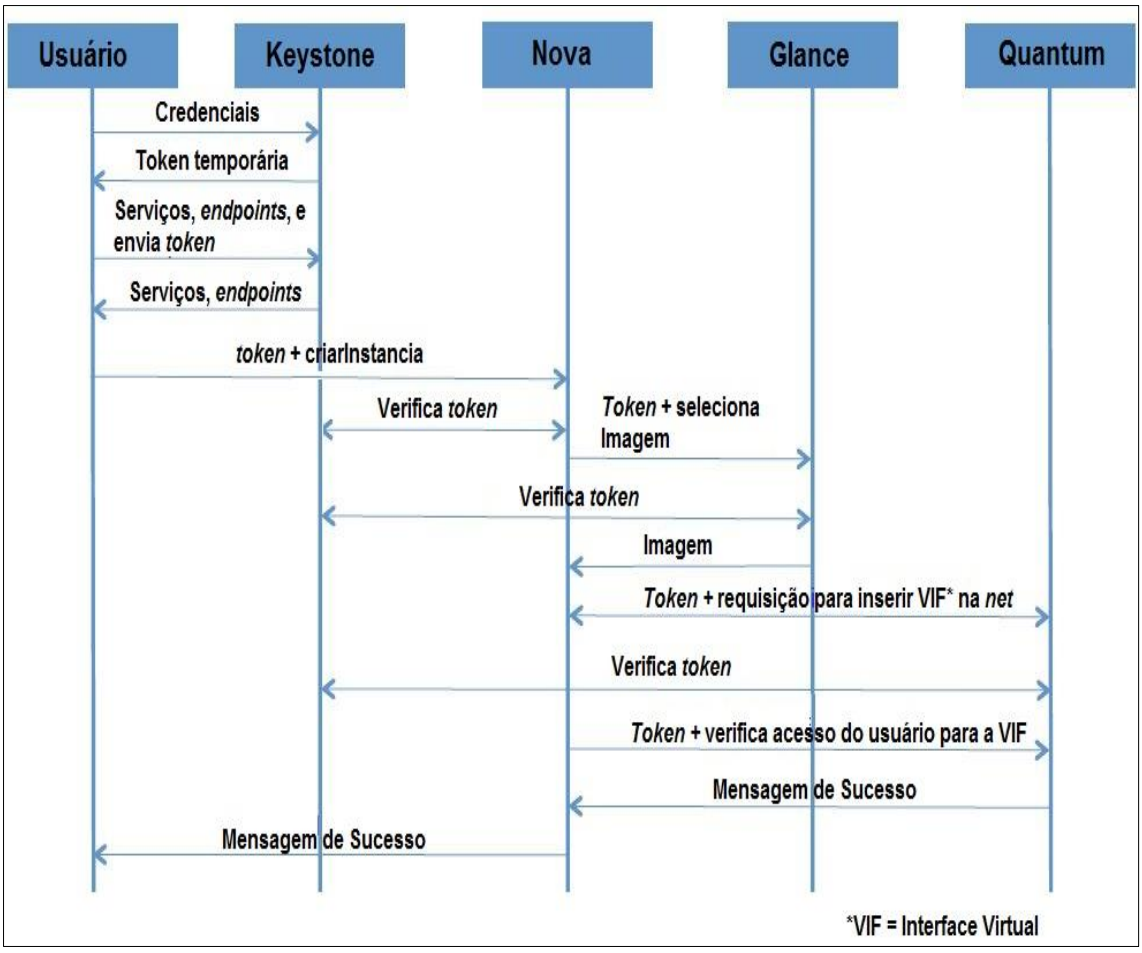


Adaptado de: (ABDUL, 2012).

Um serviço pode compreender diversos arquivos de configuração e somente um arquivo que define as políticas para cada papel. Os arquivos de configuração possuem as credenciais dos usuários, associando a ele um determinado papel. Os serviços apresentam diversos *endpoints* que estão associados à determinada URL.

O Keystone é um componente fundamental para a segurança do OpenStack, pois é responsável por garantir qual usuário deve ter acesso a um determinado serviço baseando-se no papel que é atribuído a seu *tenant*. A **Figura 14** ilustra a função do Keystone ao se requisitar um determinado serviço (disponibilização de uma VM para um usuário) na nuvem, o objetivo é esclarecer como ocorre o processo de requisições e respostas entre o usuário e os componentes, para que o usuário obtenha acesso aos serviços que está requisitando.

Figura 14 – Diagrama de Requisição de Criação de uma Máquina Virtual.



Adaptado de: (ABDUL, 2012).

A **Figura 14** ilustra um processo simples, a requisição de um usuário para criar uma máquina virtual. O processo inicia-se com o usuário enviando suas credenciais para o Keystone, que a autentica e em seguida gera uma *token* temporária para o usuário e a retorna. Com a *token*, o usuário envia a requisição dos serviços e *endpoints* que deseja utilizar para o Keystone, que retorna os serviços e *endpoints* de acordo com o *tenant* do qual o usuário faz parte. Então o usuário é liberado para requisitar a criação de uma instância ao Nova, enviando também a *token*. O Nova autentica a *token* junto ao Keystone e requisita uma imagem ao Glance, que repete o processo de validação junto ao Keystone, e em seguida, envia a *token* euma requisição ao Quantum para a inserção de uma interface virtual na *net.* O Quantum repete o processo de

autenticação junto ao Keystone e retorna a mensagem de sucesso ao Nova, que a replica para o usuário.

Observa-se neste processo que o elemento central para acessar os componentes do OpenStack é a *token* temporária gerada para o usuário, que é autenticada a cada vez que um novo componente se envolve no processo de requisição do serviço. Tal fato permite exemplificar a importância do Keystone para que seja garantida a segurança de processos simples, como esta requisição.

2.2.11 CONSIDERAÇÕES PARCIAIS

A computação em nuvem é uma tecnologia que ainda carece de uma padronização geral em muitas de suas definições. Esta ausência de padronização pode ser creditada devido ao interesse de grandes empresas e organizações na computação em nuvem. Estas aliam seus interesses comerciais ao desenvolvimento e ampliação da tecnologia, o que culmina em algumas diferenças de definições para um mesmo termo. Porém, existem diversos pesquisadores (de ambientes acadêmicos ou não) e organizações cujo único interesse é estabelecer padrões que guiem o desenvolvimento, disseminação e controle de qualidade de diversos conceitos, como é o caso da ISO (*International Organization for Standartization*) e, mais específicamente para a computação em nuvem, o NIST (*National Institute of Standards and Technology*).

Através de comparativos realizados neste capítulo (e.g., arquiteturas de referência, **2.1.5**) foi possível constatar que definições de organizações como o NIST se preocupam em estabelecer padrões isentos de objetivos comerciais, o que não pode ser verificado para uma organização cujo objetivo é obter lucro sobre seu produto. O resultado observado foi uma abordagem muito mais completa para organizações e pesquisadores cujo interesse é estabelecer padrões para as terminologias, pois muitas vezes as terminologias podem ser de acordo com o produto que se deseja vender.

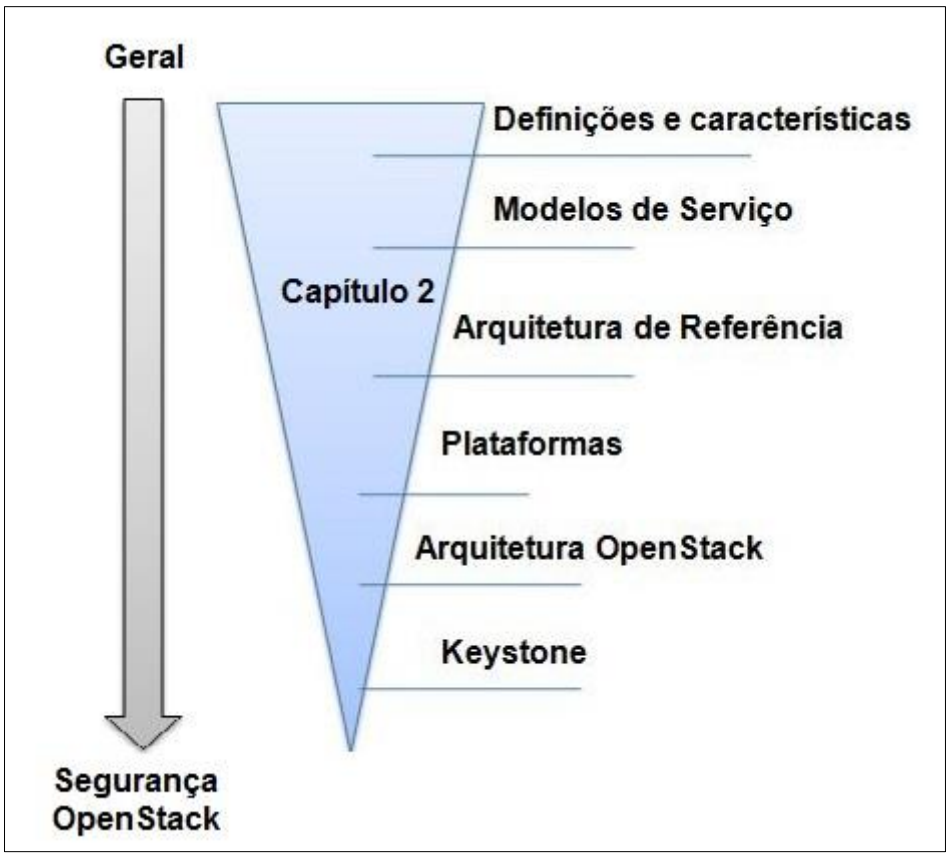
Na subseção de plataformas de computação em nuvem (**2.2**), houve a apresentação e comparação de algumas plataformas de nuvem para demonstrar que podem existir diferentes soluções para um mesmo objetivo ou serviço, e estas diferenças podem ser fundamentais ao se aderir uma solução de nuvem. A comparação entre estas plataformas (**2.2.9**) foi conveniente para dar notoriedade a estes detalhes das plataformas, como API's, interfaces, arcabouço de programação, entre outros que podem ser relevantes ao se aderir uma plataforma para utilização em larga escala. Uma análise mais completa, com simulações de uso e testes de desempenho, além da apresentação das características, seria o ideal para explicitar a diferença entre as plataformas, entretanto foge do escopo deste trabalho que é somente explicitar que existem diferenças e não demonstrá-las.

Entre as plataformas apresentadas, verificou-se que existem soluções que despontam no mercado, como o EC2 (**2.2.1.2**) e S3 (**2.2.1.1**) da AWS (*Amazon Web Services*) de gerenciamento da nuvem e armazenamento, respectivamente, pelo fato da AWS ser uma das empresas pioneiras na computação em nuvem. Contudo também foi verificado que existe uma forte tendência de crescimento para as soluções de nuvem *open source*, alavancadas principalmente pelo OpenStack (**2.2.4**), que é a plataforma de computação em nuvem que possui mais colaboradores dentre as plataformas *open source*.

Em seguida foi apresentada a arquitetura da plataforma OpenStack Grizzly (**2.2.10**) com o objetivo de evidenciar a plataforma, que é o objeto de análise deste trabalho. Foi realizada uma ilustração de suas arquiteturas, a conceitual (**Figura 10**) que ilustra como os componentes estão dispostos, e a lógica (**Figura 11**) que ilustra com mais detalhes a disposição e a relação entre os componentes do OpenStack. Em seguida, houve uma explanação de cada componente que compõe a plataforma, descrevendo basicamente seu funcionamento e sub-componentes. Uma explanação maior foi dispensada ao componente Keystone (**2.2.10.7**), responsável pela autenticação e autorização da nuvem, por este ser o elemento central de segurança no OpenStack. Isto faz com que seja de vital importância compreender como este componente é composto e como ocorre de fato o seu funcionamento.

Até a subseção de plataformas de computação em nuvem (**2.2**) foram apresentadas características e propriedades que são pertinentes ao modelo de computação em nuvem (definições, modelos de serviço, abordagens de implantação, arquitetura, entre outras). A partir desta subseção de plataformas de computação em nuvem iniciou-se uma especialização do conteúdo, apresentando diferentes plataformas, a arquitetura do Openstack Grizzly, até compreender o componente responsável pela segurança do OpenStack. **Figura 15** ilustra o andamento deste capítulo.

Figura 15 – Ilustração do Capítulo 2.



Esta abordagem permite introduzir de modo adequado a análise de segurança, apresentando primeiro conteúdos introdutórios à computação em nuvem para depois abordar assuntos específicos relacionados à análise de segurança do OpenStack. No capítulo **3** é abordada especificamente a segurança do modelo de computação em nuvem, introduzindo a análise de segurança do OpenStack Grizzly.

3 ANÁLISE DE SEGURANÇA

A computação em nuvem por ser uma tecnologia recente, traz consigo diversas vulnerabilidades provenientes de tecnologias que lhe deram origem (CASTRO et. al., 2012), grande parte destas tecnologias que são utilizadas para viabilizar este modelo computacional, como por exemplo, os mecanismos de virtualização, não foram desenvolvidas especificamente para o modelo de computação em nuvem, e sim adaptadas para sua utilização visando a maximização dos recursos disponíveis (GONZALEZ et al., 2012). Tais fatos podem ocasionar pontos de vulnerabilidade que expõem os dados armazenados e compromete, assim, a confiança no modelo de computação em nuvem. Esta insegurança em relação confiabilidade dos dados armazenados no modelo é um grande entrave para a adoção da computação em nuvem por parte de grandes organizações e empresas, sendo alvo de diversas pesquisas que visam propor soluções adequadas para os problemas.

Neste sentido, este capítulo tem o objetivo de realizar uma análise de segurança da plataforma OpenStack, uma das plataformas de IaaS *open source* mais utilizadas (PANETTIERI, 2013, *apud* Qingye, 2012), para elencar os pontos inseguros encontrados. Este capítulo está dividido em quatro partes, a primeira (**3.1**) identifica e compara (**3.1.4**) os problemas que são pertinentes à computação em nuvem, elencados por organizações que buscam prover padrões e guias de seguranças específicos para a computação em nuvem, como: NIST (**3.1.1**), CSA (*Cloud Security Alliance* - **3.1.2**) e ENISA (*European Network and Information Security Agency* -**3.1.3**). A segunda parte deste capítulo (**3.2**) elenca os problemas decorrentes da virtualização de recursos na computação em nuvem, de maneira análoga a primeira parte, porém direcionando o tema a um assunto mais específico ao trabalho. Com o objetivo de fornecer uma visão geral em relação a segurança da virtualização na plataforma, a terceira parte deste capítulo realiza uma análise quantitativa dos problemas relatados na plataforma de suporte aos *bugs* do OpenStack. A quarta e última parte, aborda os problemas discutidos no trabalho de conclusão I.

3.1 SEGURANÇA EM COMPUTAÇÃO EM NUVEM

Para realizar a análise de segurança do OpenStack, primeiramente é necessário identificar questões de segurança que devem ser abordadas para o uso da computação em nuvem, para isto, é necessário discutir e comparar as questões elencadas pelas organizações utilizadas como referência para esta pesquisa: NIST, CSA e ENISA.

Nas próximas subseções realiza-se uma curta apresentação geral dos documentos:

- **NIST (subseção 3.1.1)**: intitulada "*Guideline on Security and Privacy in Public Cloud Computing*" (JANSEN e GRANCE, 2011);
- **CSA (subseção 3.1.2)**: intitulada "*Security Guidance for Critical Areas of Focus in Cloud Computing v3.0*" (CSA, 2011);
- **ENISA (subseção 3.1.3)**: intitulada "*Benefits, Risks and Recommendations for Information Security*" (RYCK et. al, 2011).

Algumas obras como (GONZALEZ et al., 2012), (HUBBARD; JR.; SUTTON., 2010), (SLIPESTKYY, 2011) entre outras, são utilizadas visando apoiar estas referências principais, fornecendo detalhes mais específicos de modo que possa enriquecer a pesquisa. No final desta subseção é apresentado um comparativo dos guias de segurança apresentados, com o intuito de estabelecer e visualizar diferenças entre os guias de segurança.

3.1.1 GUIA DE SEGURANÇA DO NIST

Em resposta a um pedido do governo dos Estados Unidos para acelerar a adoção governamental da computação em nuvem nos Estados Unidos (BADGER et al., 2011) , o NIST elaborou um guia voltado para segurança e privacidade (JANSEN e GRANCE, 2011, pág. 6) cujo propósito é “fornecer uma visão geral sobre os desafios de segurança e privacidade envolvidos na computação em nuvem”.

O guia de (JANSEN e GRANCE, 2011) afirma que organizações que utilizam uma abordagem privada de computação em nuvem possuem um maior nível de controle sobre os dados, por este motivo o documento é focado em abordagens públicas de computação em nuvem. Ele fornece uma visão geral dos serviços de nuvem, discutindo benefícios e problemas do modelo de nuvem do ponto de vista de segurança, indicando pontos chave de segurança e privacidade, e fornecendo recomendações sobre terceirização de dados para um provedor qualquer de computação em nuvem. A Tabela 14 apresenta os pontos chave de segurança elencados pelo documento.

Tabela 14 – Principais Pontos de Segurança Elencados pelo NIST.

Pontos Chave	Subtópicos Abordados
1 – Governança	
2 - Conformidades[33]	**2.1** - Leis e regulamentos **2.2** - Conformidades de Localização dos Dados **2.3** - Descoberta Eletrônica
3 – Confiança	**3.1** - Ameaças Internas **3.2** - Propriedade dos Dados **3.3** - Serviços Compostos **3.4** - Visibilidade **3.5** - Dados Auxiliares

[33] Relacionado às normas e leis em que as organizações devem estar de acordo. Fonte: http://www.gartner.com/it-glossary/regulatory-compliance/

Pontos Chave	Subtópicos Abordados
	3.6 - Gerenciamento de Risco
4 – Arquitetura	**4.1** - Proteção do Monitor de VM's (*hypervisor*) **4.2** - Proteção de Rede Virtual **4.3** - Proteção Auxiliar dos Dados **4.4** - Proteção do Cliente
5 - Identidade e Controle de Acesso	**5.1** - Autenticação **5.2** - Controle de Acesso
6 - Isolamento de Software	**6.1** - Monitor de Qualidade da VM **6.2** Ameaças de VM's de Inquilinos (*co-tenant*)
7 - Proteção dos Dados	**7.1** - Isolamento dos Dados **7.2** Exclusão Segura dos Dados
8 – Disponibilidade	**8.1** - Interrupções **8.2** - Ataques de Negação de Serviço
9 - Responsabilidade sobre Acidentes	

Adaptada de: (SLIPETSKYY, 2011), (JANSEN e GRANCE, 2011).

Os pontos 1 e 2 da Tabela 14 tratam da Governança e Conformidades. A Governança[34] é o conjunto de políticas, funções, responsabilidades e processos que devem ser estabelecidos para orientar, direcionar e controlar como a organização usa tecnologias para atingir as metas corporativas, ou seja, um conjunto de normas estabelecidas para atingir um determinado objetivo. A conformidade abrange tópicos que visam verificar se a organização atende às normas, leis e regulamentos em questão. O item 3 abrange tópicos como: Ameaças Internas (item 3.1), de usuários internos ou inquilinos que representam um risco sobre a propriedade dos dados internos da organização; Propriedade dos Dados (item 3.2) que estão na nuvem e sua propriedade intelectual dos dados que residem na nuvem; o Ganho de Visibilidade (3.4) sobre os controles de segurança utilizados pelo provedor da nuvem; Gerenciamento de Risco (3.5).

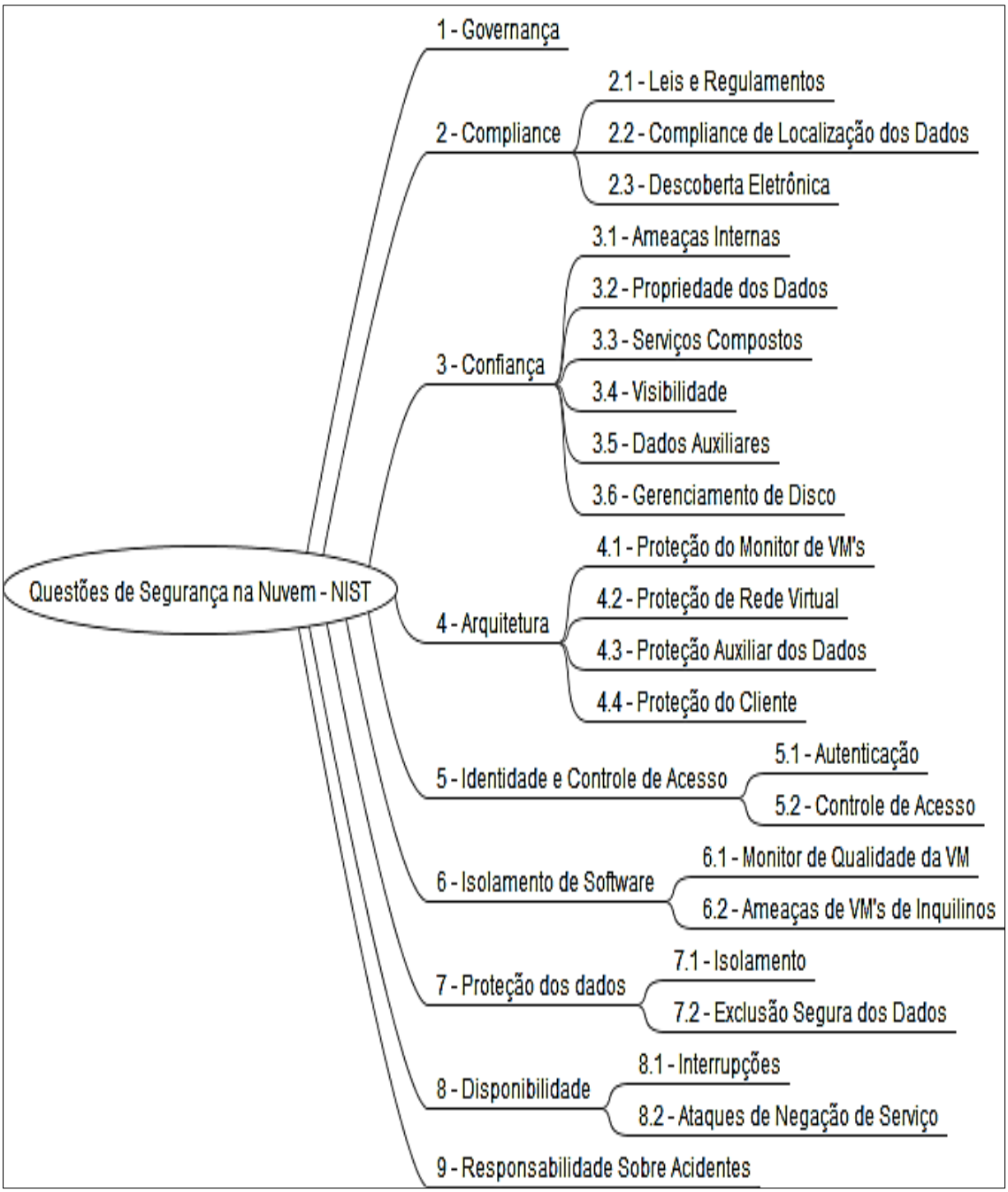
[34] Fonte: http://www.gartner.com/it-glossary/it-governance/

As questões sobre Arquitetura (item 4 da Tabela 14) levantadas por (JANSEN e GRANCE, 2011), discutem os sistemas de *software* utilizados na computação em nuvem. A descrição deste ponto de segurança inicia alertando que a introdução de monitores de máquinas virtuais (*hypervisors,* item 4.1) aumenta a superfície de ataque (definido como *Attack Surface* no documento) na computação em nuvem, em comparação com o modelo tradicional de *data centers*. O subtópico 4.2 afirma que o tráfego sobre redes virtuais não devem ser visíveis ferramentas de segurança tradicionais de rede, como *firewalls*, detectores de intrusão/prevenção, entre outros. É afirmada a necessidade de ferramentas especiais que sejam dedicadas somente pra abordar estas redes virtuais. Em 4.3 é afirmado que os dados sobre as imagens de máquinas virtuais e serviços da nuvem podem conter informações relevantes para atacantes, e por isso, necessitam de cuidados especiais. O item 4.4 discute basicamente sobre a segurança do navegador *web* do cliente.

Identidade e Controle de Acesso (item 5 da Tabela 14) focam no uso de Autenticação (item 5.1) SAML (*Security Assertion Markup Language*), que é um padrão para a troca de dados de autenticação e autorização entre domínios, e XACML (*eXtensible Access Control Markup Language*) para Controle de Acesso (item 5.2), que descreve uma linguagem para políticas de e também um formato para mensagens de requisição e resposta. O item 6, isolamento de *software*, discutem ameaças que são relacionadas à característica de *multi-tenancy* (multi-inquilinos). A abordagem sobre proteção dos dados (item 7) inicia com o destaque das diferenças entre o modelo *multi-tenancy*, no qual os dados de diferentes usuários são colocados em um único repositório, e o modelo multi-instâncias (*multi-instance*) no qual cada usuário usa e administra um repositório de dados separado (item 7.1). Após isso, é abordada a questão da Exclusão Segura dos Dados (item 7.2). Na discussão sobre Disponibilidade (item 8) foi elaborado um informe sobre ameaças indiretas provenientes do compartilhamento de um mesmo fornecedor de nuvem, com uma organização que tem um grande potencial de sofrer um ataque de negação de serviço (DoS, item 8.2). Como consequência do suporte a múltiplos inquilinos, é possível que estes tipos de ataque resultem em problemas para a disponibilização dos serviços da nuvem. A última questão abordada é a de responsabilidade sobre

acidentes, que deve resultar em uma análise para confirmar a ocorrência de um acidente ou determinar os métodos para fornecer a documentação com os detalhes necessários e cuidados para garantir a rastreabilidade e que a integridade dos dados está mantida. A Figura 17 ilustra a Tabela 14.

Figura 17 – Questões de Segurança na Nuvem Elencadas pelo NIST.



Além do levantamento de questões de segurança, observadas na Figura 17 e Tabela 14**,** que afetam os serviços da nuvem, (JANSEN e GRANCE, 2011) também definiram recomendações para cada ponto chave definido na Tabela 14. As recomendações podem ser observadas na **Tabela 15**.

Tabela 15 – Recomendações de Segurança do NIST.

Áreas	Recomendações
Governança	Aumentar práticas organizacionais pertencentes às políticas, procedimentos, e padrões utilizados para o desenvolvimento de aplicações e provisionamento de serviço na nuvem.
Conformidades	Entender os diversos tipos de leis e regulamentos que impõem obrigações de segurança e privacidade na organização e potencialmente impactos nas iniciativas de computação em nuvem, particularmente os que envolvem localização dos dados, controles de privacidade e segurança, entre outros.
Confiança	Garantir que os acordos de serviço possuam meios suficientes para permitir a visibilidade nos processos de controle de privacidade e segurança e seu desempenho. Estabelecer claramente os direitos de propriedade sobre os dados. Instituir um programa de gerenciamento de risco que seja flexível o suficiente para se adaptar constantemente. Monitorar continuamente o estado de segurança da informação para dar suporte ao gerenciamento de risco.
Arquitetura	Compreender as tecnologias de camadas inferiores que a nuvem utiliza para provisionar os serviços, incluindo as implicações que os controles técnicos envolvidos possuem na segurança e privacidade do sistema, sobre todo ciclo de vida do sistema e seus componentes.
Identidade e Controle de Acesso	Estabelecer medidas adequadas para garantir a segurança da autenticação, autorização, e outras funções de identidade e controle de acesso.
Isolamento de *Software*	Entender a virtualização e outras técnicas lógicas de isolamento que o provedor da nuvem utiliza em sua arquitetura de múltiplos inquilinos, e avaliar os

Áreas	Recomendações
	riscos envolvidos para a organização.
Proteção dos Dados	Avaliar a adequação de soluções de gerenciamento de dados do provedor de nuvem para os dados organizacionais envolvidos, e a capacidade para o acesso aos dados, para proteger os dados armazenados, em trânsito, e em uso, e para exclusão dos dados.
Disponibilidade	Compreender as previsões e procedimentos no contrato para disponibilidade, *backup* de dados, recuperação de desastres, e garantir que elas respeitem a continuidade da organização e necessidades de planejamento de contingência.
Responsabilidade sobre Incidentes	Compreender as previsões e procedimentos no contrato para responsabilidade sobre acidentes, garantindo que eles estejam de acordo com os requerimentos da organização. Garantir que o provedor contenha uma resposta transparente e mecanismos suficientes para disponibilizar informações antes e durante o acidente.

Adaptada de: (JANSEN e GRANCE, 2011).

Com a **Tabela 15** e a Tabela 14 o documento de (JANSEN e GRANCE, 2011) fornece as principais questões de segurança com as quais os fornecedores de nuvem devem se preocupar, como também estabelece as recomendações necessárias para garantir a segurança e a privacidade dos serviços destes pontos de segurança. O documento é um dos mais completos guias de segurança em computação em nuvem, sendo também um dos mais referenciados por diversos autores como: (SLIPESTKYY, 2011), (MARCON et al., 2010), (CASTRO et. al., 2012), entre outros.

3.1.2 GUIA DE SEGURANÇA CSA

A CSA (*Cloud Security Alliance)* é uma organização sem fins-lucrativos fundada[35] no final de 2008 em uma iniciativa para reforçar a segurança da computação em nuvem após uma conferência de segurança da informação. Seus membros fundadores são principalmente representantes industriais, corporações, associações, como: Dell[36], HP[37] e eBay[38] (GONZALEZ et al., 2012).

Um de seus objetivos é promover a adoção de boas práticas para promover a segurança nos ambientes de computação em nuvem.

O guia (CSA, 2011) é iniciado com uma nota editorial que descreve brevemente os passos necessários para verificar se uma organização está pronta para a transição para o modelo de nuvem. Esta nota editorial consiste em alguns itens: identificação e avaliação dos ativos para a implantação do modelo de nuvem, mapeamento dos recursos para verificar modelos de implantação possíveis (público/privado/híbrido/comunitário), avaliar potenciais provedores de nuvem, entre outros. As questões de segurança são apresentadas em seguida (no documento são chamados de domínios).

O documento está dividido em três seções: (i) arquitetura da nuvem; (ii) governança na nuvem e (iii) operações na nuvem. Ao todo o documento identifica treze domínios entre estas seções e diversos subtópicos dentro destas seções. A **Tabela 16** contém os domínios e os subtópicos apresentados em (CSA, 2011), a seção de arquitetura da nuvem não está apresentada pois contém as definições de computação em nuvem.

---

[35] Fonte: https://cloudsecurityalliance.org/about/
[36] http://www.dell.com/
[37] http://www.hp.com
[38] http://www.ebay.com

Tabela 16 - Principais Pontos de Segurança Elencados pela CSA.

Pontos chave	Subtópicos Abordados
1 - Governança e Gerenciamento de Risco Empresarial	**1.1** - Governança; **1.2** - Gerenciamento de Risco Empresarial; **1.3** - Gerenciamento de Informações; **1.4** - Gerenciamento de Terceirizações.
2 - Aspectos Legais: Contratos e Descoberta Eletrônica	**2.1** - Aspectos Legais; **2.2** - Considerações de Contrato; **2.3** - Questões Levantadas pela Descoberta Eletrônica.
3 - Conformidades e Auditoria	**3.1** - Conformidades; **3.2** - Auditoria.
4 - Gerenciamento de Informação e Segurança dos Dados	**4.1** - Segurança dos Dados; **4.2** - Localização dos Dados. **4.3** - Gerenciamento de Informação; **4.4** - Ciclo de Vida da Segurança dos Dados; **4.5** - Governança da Informação.
5 - Interoperabilidade e Portabilidade	**5.1** - Introdução a Interoperabilidade; **5.2** - Introdução a Portabilidade.
6 - Segurança Tradicional, Continuidade de Negócios e Recuperação de Desastres	**6.1** - Ameaças Internas; **6.2** - Segurança Física dos Recursos Humanos; **6.3** - Continuidade de Negócios; **6.4 -** Recuperação de Desastres.
7 - Operações de Datacenter	Não há subtópicos.
8 - Responsabilidade sobre Acidentes, Notificações e Correções	**8.1** - Características de Nuvem que geram Responsabilidade sobre Acidentes; **8.2** - Modelo Arquitetural de Segurança como Referência; **8.3** - Ciclo de Examinação de Responsabilidade sobre Acidentes
.9 - Aplicações Seguras	**9.1** - Ciclo de Desenvolvimento Seguro **9.2** - Autenticação, Autorização e Conformidades **9.3** - Identidade e Gerenciamento de Acesso para Aplicações **9.4** - Testes de Invasão para Aplicações
10 - Gerenciamento de Chaves e Criptografia	**10.1** - Cifragem de Dados em Trânsito; **10.2** - Cifragem de Dados em Repouso; **10.3** - Cifragem de Backup de Dados; **10.4** - Armazenamento Seguro de Chaves; **10.5** - Acesso ao Armazenamento de Chaves; **10.6** - Backup e Recuperação de Chaves.
11 - Identidade e Gerenciamento de	**11.1** - Provisionamento de Identidade; **11.2** - Autenticação;

Pontos chave	Subtópicos Abordados
Acesso	**11.3** - Federação; **11.4** - Autorização e Gerenciamento de Perfis de Usuário.
12 – Virtualização	**12.1 -** Arquitetura do Hypervisor.
13 - Segurança como Serviço	**13.1** - Ubiquidade do Serviço; **13.2** - Preocupações na Implementação; **13.3** - Benefícios de se Implementar Segurança como Serviço; **13.4** - Diversidade de Serviços que Podem Ser Categorizados.

Adaptada de: (SLIPETSKYY, 2011), (CSA, 2011).

A Governança, item 1 da **Tabela 16**, e o Gerenciamento de Risco Empresarial preocupam-se com os processos organizacionais da segurança da informação. Os acordos de níveis de serviço (SLA's) são discutidos em detalhes neste domínio, especificando que requisições de segurança devem ser especificadas na SLA e reforçadas por um contrato. Para o Gerenciamento de Risco, é advertido no documento para não avaliar somente o fornecedor da nuvem, mas também os terceiros (os serviços que são terceirizados pelo provedor) com quem o provedor está envolvido.

Os itens 2 e 3 da **Tabela 16** discutem as conformidades com regulamentações governamentais, normas da própria organização, aspectos legais (e.g., propriedade intelectual, responsabilidades), entre outras, estabelecendo como estas regras podem influenciar a segurança interna da organização. O item 4, descreve diversos subtópicos relacionados a manipulação dos dados, como Segurança (Item 4.1), Localização (item 4.2), Gerenciamento da Informação (item 4.3), entre outros, estabelecendo os modos em que os dados devem ser gerenciados para evitar vulnerabilidades que possam expor os dados armazenados. Uma discussão sobre Portabilidade e Interoperabilidade (item 5) inicia com uma especificação de possíveis razões para migrar os dados para o modelo de nuvem, também estabelece que a computação em nuvem ainda carece de uma padronização de diversos fatores, causando uma dificuldade para a interoperabilidade de serviços, e

consequentemente, a dificuldade de transição entre provedores de nuvem. Vale ressaltar a contribuição das plataformas de nuvem *open source* para este fator, que contribuem para o aumento da interoperabilidade entre provedores através da divulgação de seus códigos e interfaces (como a OCCI - *Open Cloud Computing Interface*).

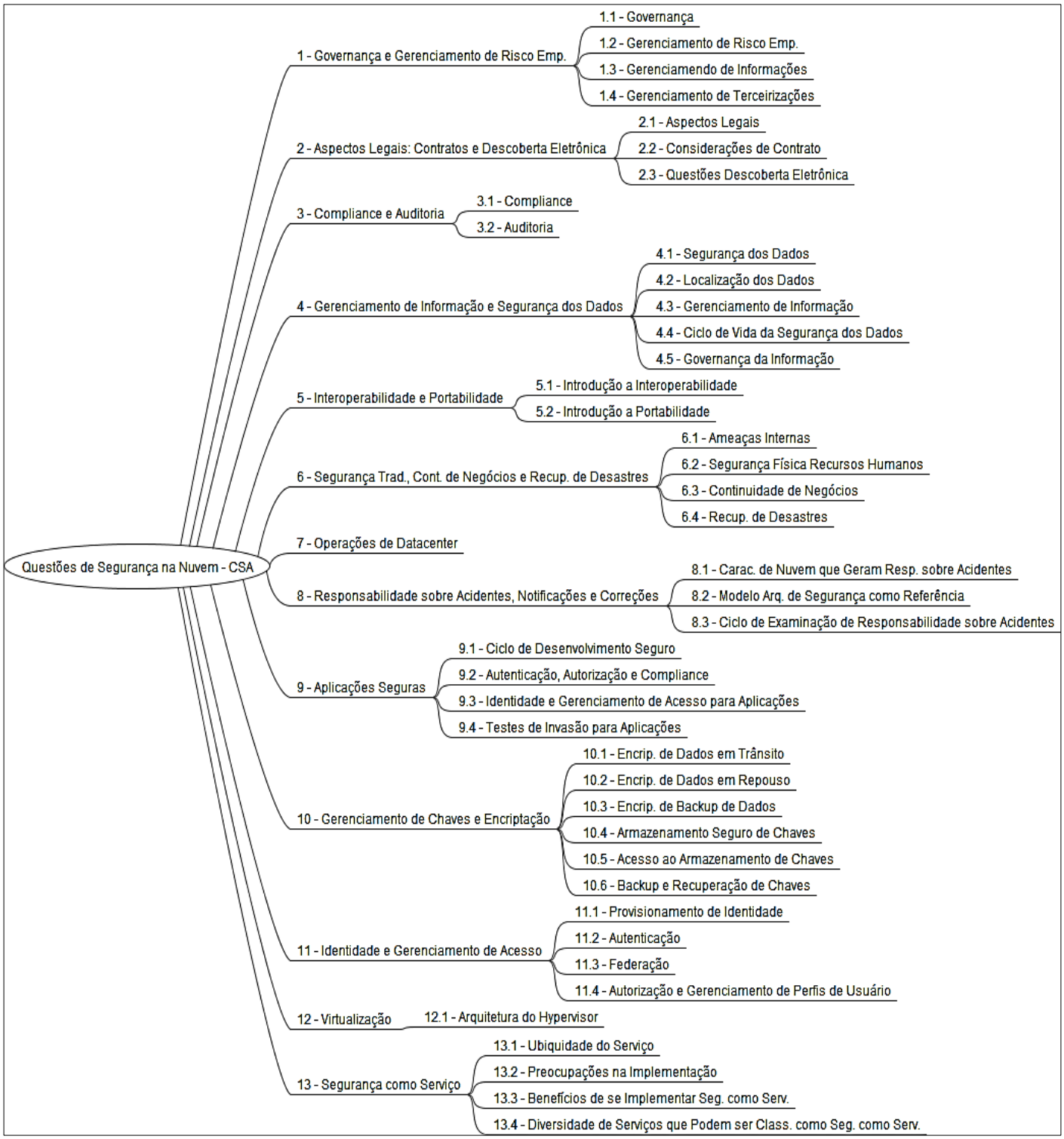
No item 6 da **Tabela 16** os pesquisadores da CSA discutem sobre as Ameaças Internas da organização, recuperação de desastres, continuidade de negócios alegando que a SLA por si só não é suficiente para satisfazer as necessidades do usuário em relação a segurança, sendo necessário uma constante verificação dos pilares da segurança (confidencialidade, integridade e disponibilidade). As Operações do *Datacenter* (item 7) tratam principalmente das questões que devem ser considerada no momento da escolha do provedor de nuvem, e a CSA recomenda a atenção nas seguintes questões: auditoria, disponibilidade, desempenho, gerenciamento de *patchs*, compartimentalização e suporte. Na discussão do próximo assunto, Responsabilidade sobre Acidentes (item 8), é elaborado um informativo sobre ferramentas de detecção de incidentes e análises para obtenção de informações das ferramentas para a prevenção dos incidentes.

No item 9, de Aplicações Seguras, é descrito o Ciclo de Desenvolvimento de Aplicações (item 9.1) e nos subtópicos seguintes técnicas para garantir a segurança da informação como (Autenticação, Autorização e Conformidades - item 9.2), gerenciamento de identidade (item 9.3), entre outros. É recomendada pela CSA neste domínio, a avaliação de vulnerabilidades através de testes de invasão (item 9.4) para a garantia da segurança da aplicação. No tópico sobre Cifragem e Gerenciamento de Chaves (item 10), é dada ênfase para a segurança da comunicação entre *hosts,* até mesmo com a comunicação interna na rede do provedor, alertando que devido à característica de *multi-tenancy* a questão do gerenciamento de chaves torna-se ainda mais complicada e é necessária uma atenção do consumidor da nuvem neste aspecto.

O item 11, realiza uma introdução à verificação de identidade na nuvem identificando aspectos que devem ser levados em consideração ao

implementar um mecanismo de identidade para a nuvem, destacando a dificuldade da escalabilidade no modelo. Ainda neste domínio é tratado especificamente sobre o Provisionamento de Identidade (item 11.1), Autenticação (item 11.2), entre outros. O item 12 trata as questões de segurança relacionadas às máquinas virtuais, a segurança do *hypervisor,* entre outros. O documento é finalizado com uma discussão a respeito da Segurança como Serviço (item 13), que realça a diferenciação do modelo de segurança da computação em nuvem em relação ao modelo tradicional de *datacenters*. O domínio aborda questões como práticas de implementação, benefícios de se implementar a segurança como serviço e a variedade de serviços que podem ser categorizados como segurança como serviço. A **Figura 18** lista os pontos abordados pelo documento da CSA.

Figura 18 - Questões de Segurança na Nuvem Elencadas pela CSA.



Além do levantamento dos aspectos políticos/legais, e técnicos que envolvem o conceito de computação em nuvem, a CSA realizou uma recomendação para cada tópico abordado, chamados de domínios em seu

documento. A **Tabela 17** exibe a recomendação de segurança para cada domínio.

Tabela 17 - Recomendações de Segurança da CSA.

Pontos Chave	Recomendações
1 - Governança e Gerenciamento de Risco Empresarial	Reinvestir o dinheiro economizado com a adoção da nuvem para melhorar a segurança do provedor de nuvem, das aplicações, vistorias e auditorias. Métricas e padrões para medir desempenho e efetividade do gerenciamento de segurança da informação.
2 - Aspectos Legais: Contratos e Descoberta Eletrônica	Não encontrada no documento.
3 - Conformidades e Auditoria	Utilizar auditores especializados no modelo de nuvem; Contratos devem prover uma revisão de terceiros nas métricas, SLA e conformidades;
4 - Gerenciamento de Informação e Segurança dos Dados	Compreender a arquitetura de armazenamento em questão, escolher o tamanho da dispersão de dados quando disponível; monitorar bases de dados internas; cifrar todos dados importantes ao realizar uma migração, entre outras.
5 - Interoperabilidade e Portabilidade	Compreender como uma VM pode ser capturada e inserida em novos provedores de nuvem; identificar e eliminar quaisquer extensões específicas de provedores de nuvem; compreender os custos de migração de dados de um servidor para o outro; entre outras.
6 - Segurança Tradicional Continuidade de Negócios e Recuperação de Desastres	Consumidores de nuvem não devem depender de um único provedor; fornecedores de IaaS devem possuir contratos com múltiplos fornecedores de plataforma e possuir ferramentas para rápida recuperação em caso de incidentes; Validação dos dados automática ou permitir que os usuários possam realizar através de protocolos de validação; entre outras.
7 - Operações de Data Center	Possuir gerenciamento de processos, máquinas, práticas, e softwares para entender e reagir a tecnologia executando dentro do datacenter; Compreender a missão do que esta sendo executado dentro do datacenter; Compreensão de quais partes devem ser responsáveis pelas conformidades estabelecidas na SLA; entre outras.
8 - Responsabilidade sobre Acidentes, Notificações e Correções	Consumidores de nuvem devem compreender como provedores definem os eventos de interesse em relação aos incidentes de segurança e quais eventos/incidentes são reportados aos usuários; Consumidores devem possuir um meio de comunicação com o provedor em caso de incidentes; entre outras.
9 - Aplicações Seguras	Realizar análises de risco para as aplicações; vistoria detalhada dos vetores de ataque e riscos no ambiente de nuvem; procurar desenvolver e manter uma arquitetura de software segura; entre

Pontos Chave	Recomendações
	outras.
10 - Gerenciamento de Chaves e Criptografia	Utilizar boas práticas de gerenciamento de chaves; Chaves de escopo podem ser mantidas em um nível individual e de grupo; utilizar algoritmos padrões de cifragem, entre outras.
11 - Identidade e Gerenciamento de Acesso	Todos os atributos utilizados devem possuir um nível de confiança; todos os atributos devem ser conectados a uma identidade; desenvolvedores devem garantir que os serviços possuam função de importar/exportar seguindo padrões como XACML[39]; entre outras.
12 – Virtualização	Consumidores devem identificar o tipo de virtualização que o provedor utiliza; desenvolvedores devem cifrar imagens de VM's quando não estiverem em uso; configurações padrões das VM's devem seguir ou possuir os requisitos mínimos de segurança estabelecidos; entre outras.
13 - Segurança como Serviço	Desenvolvedores devem um canal de comunicação seguro entre os *tenants*; fornecedores devem prover notificação automática de segurança; consumidores devem requisitar terceiros para intermediar a negociação da SLA e auditar os serviços; entre outras.

Fonte: (CSA, 2011).

Na **Tabela 16** a CSA lista as questões de segurança, subdivididas em treze áreas, que devem ser consideradas por usuários e provedores de nuvem e na **Tabela 17**, elabora as respectivas recomendações de segurança para cada uma das treze área. Cabe observar também que na **Tabela 17** não foram listadas todas as recomendações de segurança apresentadas no documento, somente uma parte para cada área. O guia é um conjunto de boas práticas para cada um dos treze domínios definidos, com o objetivo de estabelecer, nesta terceira versão, uma base para se conseguir um modelo seguro de computação em nuvem.

[39] XACML (*eXtensible Access Control Markup Language*)

3.1.3 GUIA DE SEGURANÇA ENISA

A ENISA (*European Network and Information Security Agency*) é uma organização que visa (RYCK et al., 2011, pág. 5.):

> "Aumentar a capacidade da União Européia, os estados membros da União Européia e a comunidade de negócios para prevenir, endereçar e responder a problemas de segurança da informação e redes."

O documento inicia com uma especificação dos benefícios de computação em nuvem sob o prisma da segurança, listando os benefícios da segurança em computação em nuvem. Em seguida, é realizada a descrição de um processo de análise e avaliação de riscos com baseando-se em três cenários de caso de uso: (i) computação em nuvem em pequenas e médias empresas; (ii) impacto da computação em nuvem na resiliência de serviço e (iii) computação em nuvem *eGovernment* (computação em nuvem para administração pública). Esta análise é baseada na probabilidade de cenários de incidentes com uma estimativa de impacto nos negócios e níveis de risco (baixo, médio, alto).

A lista de questões de segurança contém riscos que são específicos para o modelo de computação em nuvem, como também riscos que não são (SLIPETSKYY, 2011). Neste trabalho, apenas as questões de segurança serão listadas, e os problemas não relacionados serão omitidos. As questões de segurança (ou riscos, como definidos no documento) relevantes para computação em nuvem estão apresentadas na **Tabela 18**. Ao contrário dos documentos de (JANSEN e GRANCE, 2011) e (CSA, 2011) a ENISA define as questões de segurança especificamente, por este motivo não existem subtópicos.

Tabela 18 - Principais Pontos de Segurança Elencados pela ENISA.

Área	Tópicos
Questões Políticas e Organizacionais	**1** - *Lock-in*;
	2 - Governança;
	3 - Conformidades;
	4 - Perda de Reputação de Negócios devido à *tenants*;
	5 - Finalização ou Falha do Serviço de Nuvem;
	6 - Aquisição de Provedor de Nuvem;
	7 - Falha de Cadeia de Fornecimento.
Questões Técnicas	**8** - Exaustão de Recursos;
	9 - Isolamento de Falhas;
	10 - Ameaças Internas;
	11 - Gerenciamento de Interface;
	12 - Interceptação de Dados em Trânsito;
	13 - Vazamento de Dados em Upload e Download;
	14 - Exclusão Sem Efeito ou Insegura dos Dados;
	15 - Negação de Serviço Distribuída;
	16 - Negação de Serviço Econômica;
	17 - Perda de Chaves Criptográficas;
	18 - Captação de Sondas ou Escaneamentos Maliciosos;
	19 - Comprometimento de Motor de Serviço;
	20 - Conflitos entre Endurecimento de Procedimentos do Cliente e o Ambiente de Nuvem.
Aspectos Legais	**21** - Descoberta Eletrônica e Intimações Judiciais;
	22 - Riscos de Mudanças de Jurisdição;
	23 - Risco de Proteção dos Dados;
	24 - Riscos de Licenciamento.

Adaptada de: (SLIPETSKYY, 2011), (RYCK et al., 2011).

A lista de questões políticas e organizacionais inicia com a discussão da questão do *lock-in* (item 1 da **Tabela 18**), no qual é relatado a situação de portabilidade entre fornecedores de nuvem em cada categoria de serviço na nuvem (IaaS, PaaS e SaaS). Após isto, Governança (item 2), Conformidades (item 3) e ameaças indiretas de usuários que estão utilizando o mesmo fornecedor (item 4 - perda de reputação devido a inquilinos) são discutidas. É

destacado que desafios de disponibilidade podem ser resultado de problemas do próprio fornecedor da nuvem (itens 5 e 6) ou terceiros em que o fornecedor confia (item 7).

A lista de questões técnicas inicia com ameaças resultantes da Exaustão de Recursos (item 8) como: incapacidade de fornecer recursos (acordados na SLA ou adicionais) aos clientes, indisponibilidade de serviços, entre outros. O item 9 trata de falhas nos mecanismos que isolam recursos compartilhados entre usuários (falhas no *hypervisor*). Em seguida são apresentadas as vulnerabilidades que causam podem vir a causar as ameaças internas, como: papéis e responsabilidades mal definidas, vulnerabilidades no sistema operacional, procedimentos físicos de segurança inadequados, entre outros.

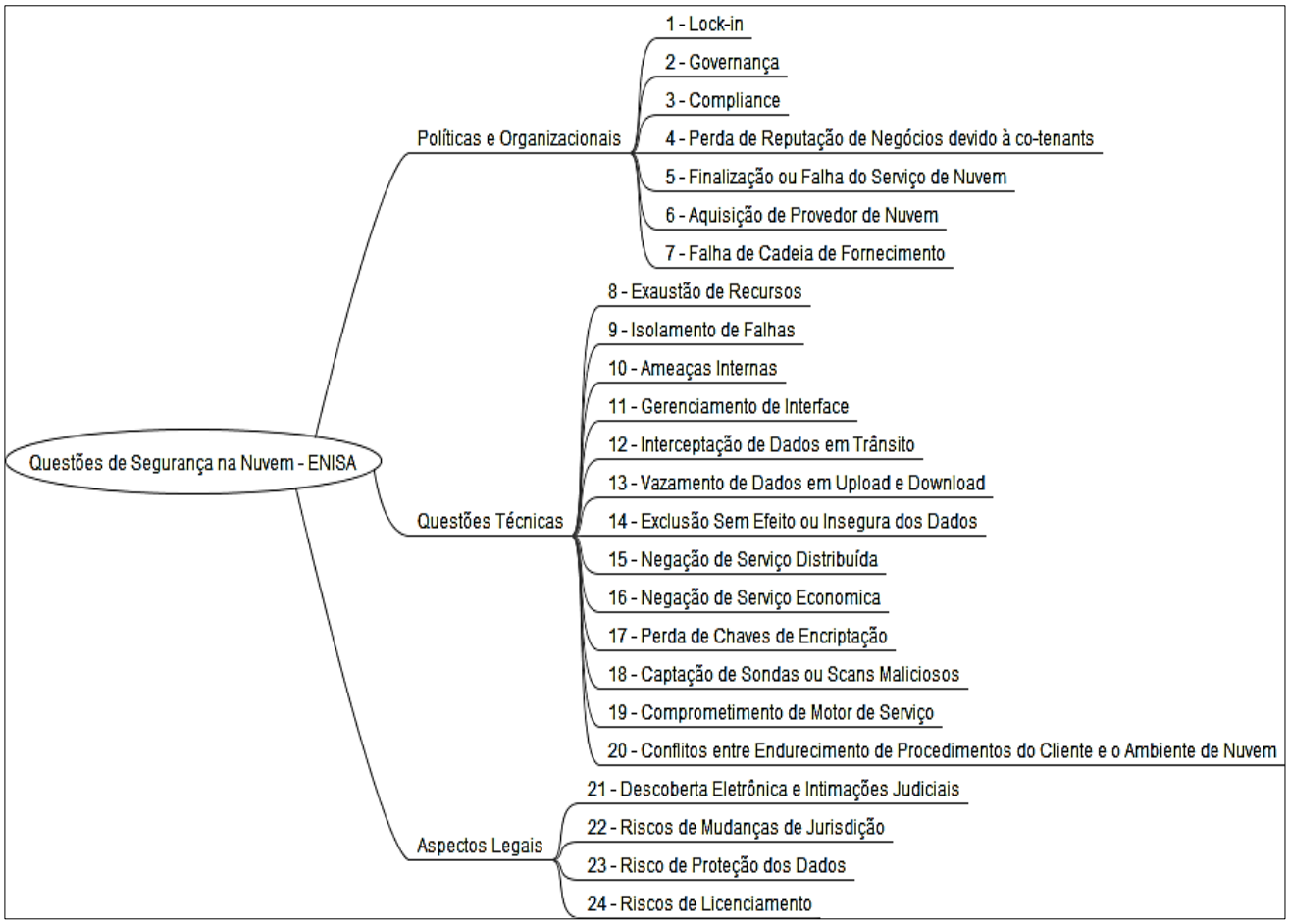
Alguns dos riscos apresentados podem ser agrupados mais amplamente, como gerenciamento de interface (item 11) e comprometimento de motor de serviço (item 19) que tratam da segurança de aplicações listando vulnerabilidades no *software* do provedor da nuvem. A interceptação de dados em trânsito na rede (item 12) também é análoga ao vazamento de dados em *upload* e *download*, discutindo a interceptação dos dados por vulnerabilidades na infraestrutura do fornecedor da nuvem ou no enlace com o usuário.

As ameaças de negação de serviço (*Denial of Service*) são apresentadas como dois riscos no documento: negação de serviço distribuída (item 15 - *Distribucted Denial of Service*), quando um ataque causa problemas de disponibilidade, e negação de serviço econômica (item 16), quando requisições maliciosas de serviço aumentam o tráfego na rede causando uma amplificação no custo dos serviços para o usuário da nuvem. Outros assuntos técnicos como detecção de sondagens e/ou *scanners* maliciosos (item 19) internos à infraestrutura da nuvem, e esforços insuficientes do consumidor para garantir a segurança da nuvem (item 20) completam a lista de questões técnicas abordadas pela ENISA.

A lista de aspectos legais inicia com procedimentos para responder a processos de descoberta eletrônica e intimações judiciais (e.g., o governo pode ter requisitar, por algum motivo, algum dado armazenado na nuvem - item 21). As outras três questões (itens 22,23 e 24) descrevem preocupações com a

manipulação dos dados, como Conformidades de Localização (item 22), Proteção (item 23) e riscos provenientes da perda de propriedade intelectual dos dados armazenados na nuvem (item 24). A **Figura 19** apresenta os pontos abordados elencados pela ENISA.

Figura 19 - Questões de Segurança na Nuvem Elencadas pela ENISA.



A **Figura 19** ilustra o modo em que as questões de segurança elencadas pela ENISA foram dispersas em seu documento. As recomendações de segurança foram realizadas na seção de “Recomendações e Mensagens Chave”, do documento da ENISA. Estas são realizadas utilizando uma estrutura diferente da empregada para descrever as disponibilidades, dificultando a localização da recomendação para uma determinada vulnerabilidade. Adicionalmente, as recomendações de segurança são

realizadas através de questões, nas quais leitor deve responder para se guiar evitar cada vulnerabilidade.

3.1.4 COMPARATIVO NIST, CSA E ENISA

Para constatar as diferenças e sumarizar as questões elencadas nos documentos de (JANSEN e GRANCE, 2011), (CSA, 2011) e (RYCK et. al., 2011), a **Tabela 19** apresenta um comparativo com questões políticas, organizacionais e legais, e a **Tabela 20** com questões técnicas listadas nas subseções do NIST, CSA e ENISA. As tabelas apresentadas verificam se as questões são ou não abordadas nos documentos, em caso positivo a coluna será marcada com “ok”, e em caso negativo com “n.a.“ (não abordado). Para o comparativo de questões políticas, organizacionais e legais da **Tabela 19** os parâmetros são (SLIPETSKYY, 2011):

- Governança e Gerenciamento de Risco;
- Conformidades e Auditoria;
- Aspectos Legais (eletrônicos e contratos);
- Ameaças Internas;
- Tratamento dos Dados;
- Relatório de Acidentes;
- Gerenciamento de *Patchs; e*
- Vistorias.

Os parâmetros utilizados como base para este comparativo foram selecionados de acordo com a tese de (SLIPETSKYY, 2011), entretanto, cabe observar que tais parâmetros não fornecem a noção de qual documento é mais completo, pois representam apenas uma pequena parcela dos tópicos que são abordados nos documentos. Os parâmetros podem variar em função das necessidades de segurança, dos objetivos das organizações em relação à

segurança, entre outras variáveis. Estes parâmetros, embora reduzidos, apresentam tópicos fundamentais para garantir a segurança da computação em nuvem, mas dependendo das necessidades podem ser adicionados ou removidos parâmetros que sejam mais relevantes para as necessidades de uma organização.

Tabela 19 - Comparativo das Questões Políticas, Organizacionais e Legais.

Questões Políticas, Organizacionais e Legais	Subtópicos	NIST	CSA	ENISA
1 - Governança e Gerenciamento de Risco		Abordado	Abordado	Abordado
2 - Conformidades e Auditoria	**2.1** - Conformidades de Leis e Regulamentos	Abordado	Abordado	Abordado
	2.2 - Conformidades de Localização dos Dados	Abordado	Abordado	Abordado
3 - Aspectos legais: contratos e descoberta eletrônica		Abordado	Abordado	Abordado
4 - Ameaças Internas		Abordado	Abordado	Abordado
5 - Tratamento dos Dados	**5.1** - Proteção dos Dados a respeito dos Usuários	Abordado	N.A.	N.A.
	5.2 - Propriedade Intelectual	Abordado	Abordado	Abordado
	5.3 - Disponibilidade dos Dados p/ Análise Forense	Abordado	Abordado	Abordado
6 - Relatórios de Acidentes		Abordado	Abordado	Abordado
7 - Gerenciamento de Patch		Abordado	Abordado	Abordado
8 - Vistorias	**8.1** - Vistoria do Provedor	Abordado	Abordado	Abordado
	8.2 - Vistoria das Dependências do Provedor	Abordado	Abordado	Abordado

Adaptado de: (SLIPETSKYY, 2011), (CSA, 2011), (RYCK et.al., 2011), (JANSEN e GRANCE, 2011).

A única diferença encontrada entre os documentos foi no item 5.1 de Tratamento dos Dados dos Usuários, enquanto a CSA e ENISA se preocupam com a segurança dos dados que o usuário pretende armazenar na nuvem, o NIST dá ênfase na segurança dos dados do próprio usuário, como suas informações pessoais que podem ser alvos de ataques (SLIPETSKYY, 2011). A questão do item 2.2 de Conformidades de Localização dos Dados embora a ENISA não faça uma discussão direta sobre o assunto, indiretamente este assunto é tratado.

Todos os três documentos fazem menção em relação aos outros tópicos, como Governança, Compliances, Aspectos Legais, Ameaças Internas, Relatórios de Acidentes, Gerenciamento dos *Patchs* lançados e as Vistorias. Portanto a única diferença entre os documentos (para estes tópicos observados) é o item 5.1 de Proteção dos Dados a Respeito do Usuário.

Os parâmetros de comparação para as questões técnicas da **Tabela 20** também foram estabelecidos de acordo com (SLIPETSKYY, 2011):

- Disponibilidade;
- Portabilidade;
- Gerenciamento de Dados;
- Identidade e Gerenciamento de Acesso;
- Segurança de Aplicação; e
- Virtualização.

Tabela 20 - Comparativo de Questões Técnicas.

Questões Técnicas	Subtópicos	NIST	CSA	ENISA
1 - Disponibilidade	**1.1** - Interrupções	Abordado	Abordado	Abordado
	1.2 - Exaustão de Recursos	N.A.	Abordado	Abordado
	1.3 - Ameaças de Disponibilidades por Ataques de DoS	Abordado	NA	Abordado
	1.4 - Ameaças por Compartilhamento de Dados	Abordado	Abordado	Abordado
2 - Portabilidade	**2.1** - Portabilidade de Dados	Abordado	Abordado	Abordado

Questões Técnicas	Subtópicos	NIST	CSA	ENISA
	2.2 - Portabilidade de Imagens de VM's	N.A.	Abordado	Abordado
	2.3 - Portabilidade de Aplicações	Abordado	Abordado	Abordado
3 - Gerenciamentos de Dados	**3.1** - Isolamento	Abordado	Abordado	Abordado
	3.2 - Backup e Recuperaçãp	Abordado	Abordado	Abordado
	3.3 - Exclusão	Abordado	Abordado	Abordado
	3.4 - Cifragem	Abordado	Abordado	Abordado
	3.5 - Gerenciamento de Chaves	Abordado	Abordado	Abordado
	3.6 - Verificação de Integridade	Abordado	Abordado	N.A.
4 - Identidade e Gerenciamento de Acesso	**4.1 -** Provisionamento e Desprovisionamento de Identidade	N.A.	OK	Abordado
	4.2 - Federação de Identidade	Abordado	Abordado	Abordado
	4.3 - Autenticação	Abordado	Abordado	Abordado
	4.4 - Autorização e Controle de Acesso	Abordado	Abordado	Abordado
5 - Segurança de Aplicação	**5.1** - Proteção do Cliente	Abordado	Abordado	Abordado
	5.2 - Proteção do Servidor	Abordado	Abordado	Abordado
	5.3 - Proteção da Imagem da VM	Abordado	Abordado	Abordado
	5.4 - Proteção dos Arquivos de Log	N.A.	Abordado	Abordado
6 - Virtualização	**6.1** - Proteção do Hypervisor	Abordado	Abordado	Abordado
	6.2 - Proteção do Sistema Operacional Visitante	Abordado	Abordado	Abordado
	6.3 - Proteção da Rede Virtual	Abordado	Abordado	Abordado

Adaptado de: (SLIPETSKYY, 2011), (RYCK et al., 2011), (CSA, 2011), (JANSEN e GRANCE, 2011).

O comparativo é iniciado com a questão da disponibilidade (item 1), subdividida nos tópicos: interrupções, exaustão de recursos, ameaças de disponibilidade causadas por ataques de negação de serviço, e ameaças resultantes do compartilhamento de dados. As interrupções (item 1.1) podem ser causadas por falhas de *hardware* ou nas instalações (falta de energia elétrica), ou desastres naturais (CSA, 2011), todos estes assuntos são abordados nos documentos do NIST, CSA e ENISA. A exaustão de recursos (item 1.2) pode acontecer quando o provedor de nuvem não se prepara adequadamente para prever o quanto de recursos os usuários podem necessitar, e a demanda por esses recursos supera a capacidade de fornecimento do provedor (JANSEN e GRANCE, 2011). Na teoria o fornecimento dos recursos deveria ser feita automaticamente (pela demanda do usuário), no entanto (CSA, 2011, pág. 7) alerta: “a teoria da computação em nuvem ainda é muito maior do que a parte prática, e muitos consumidores se confundem com isto”. Durante a análise, não foi encontrada uma discussão no documento de (JANSEN e GRANCE, 2011) sobre a exaustão de recursos.

A negação de serviços (DoS - item 1.3) é outra ameaça relacionada à disponibilidade, que é discutida nos documentos do NIST e da ENISA, porém é omitida no guia da CSA. Além do ataque de negação de serviço tradicional, que tem a intenção é causar problemas de disponibilidade, o documento da ENISA descreve um tipo de ataque de negação de serviço econômica, na qual o objetivo é causar um aumento no custo dos serviços do usuário, nos casos em que o preço do serviço depende do número de serviços requisitados (SLIPETSKYY, 2011) (RYCK et al., 2011). As ameaças indiretas de outros usuários (causadas pela característica de *multi-tenancy*) são discutidas nos três documentos. Um exemplo seria um ataque de DoS em um usuário com quem se está compartilhando algum recurso, tornando este serviço indisponível.

A questão da portabilidade (item 2) ou (ausência de) *lock-in*, continua a tabela de questões técnicas. Como afirmado em (CSA, 2011), (BORGES et al., 2011) entre outros, os usuários podem trocar de provedor por diversos motivos, por exemplo: altos custos, baixa qualidade de serviço, questões de disponibilidade, entre outros aspectos. A questão da portabilidade pode variar

também de acordo com o modelo de serviço (IaaS/PaaS/SaaS). Segundo (SLIPETSKYY, 2011) para o usuário transferir dados e aplicações para outro provedor de nuvem, o usuário necessitará de um nível suficiente de Portabilidade de Dados (item 2.1) e Aplicações (item 2.3). No caso de IaaS, a Portabilidade de Imagens de VM's (item 2.3) também deve ser considerada. Avaliando os três documentos foi possível notar que o NIST não discute em sua seção de arquitetura a questão de portabilidade de imagens de VM's.

Analisando os documentos sob o prisma do Gerenciamento de Dados (item 3), observa-se os subtópicos definidos por (SLIPETSKYY, 2011): Isolamento em Ambiente *Multi-tenancy* (item 3.1), Exclusão Segura dos Dados (item 3.3), e *Backup* e Recuperação de Dados (item 3.2), Cifragem de Dados em Trânsito e em Repouso (item 3.4), *Backup* e Recuperação de Chaves (item 3.5) e Verificação de Integridade (item 3.5). É dada maior atenção à questão de *backup* e recuperação dos dados no documento da CSA, enquanto nos documentos da ENISA e NIST, é dada maior importância na cifragem dos dados no *backup* e a exclusão dos dados das cópias de *backup*. Todos os documentos abordam estes temas, exceto a ENISA que não faz menção sobre a verificação de integridade das chaves criptográficas.

No Gerenciamento de Acesso e Identidade (item 4) os seguintes subtópicos foram utilizados: Fornecimento e Remoção de Identidade (item 4.1), Federação de Identidade (item 4.2), Autenticação (item 4.3) e Autorização e Controle de Acesso (item 4.4). No comparativo entre os documentos foi observado que não existe um tópico no documento do NIST que aborde o item 4.1, os outros itens são abordados por todos os três guias de segurança.

A Segurança de aplicação (item 5) incluí os seguintes subtópicos: Proteção do Cliente (item 5.1) e Servidor (item 5.2), Proteção da Imagem da VM (item 5.3) e Proteção dos Arquivos de Log (item 5.4). As informações geradas nos arquivos de log podem conter informações importantes, e (MARCON et al, 2010) recomenda é necessário uma atenção extra no gerenciamento destes arquivos, pois as informações que são armazenadas nos logs podem pertencer a diversos usuários que compartilham recursos de um provedor. O comparativo destes tópicos demonstrou que todos estes

subtópicos estão sendo discutidos nos documentos, exceto o gerenciamento de arquivos de log, que não é abordado no documento do NIST.

A questão de Virtualização (item 6) finaliza o quadro de questões técnicas, com o seguintes subtópicos: Proteção do *Hypervisor* (item 6.1), Proteção do Sistema Operacional Visitante (item 6.2) e Proteção da Rede Virtual (item 6.3). Todos os documentos apresentam tópicos discutindo os problemas relacionados à virtualização.

3.1.5 CONSIDERAÇÕES SOBRE O COMPARATIVO

Este comparativo fornece uma visão geral sobre a abordagem de cada organização em relação aos parâmetros definidos por (SLIPETSKYY, 2011), indicando qual inclinação o documento em questão possui sobre determinado parâmetro abordado. Para os parâmetros definidos neste trabalho, foi possível constatar que para questões políticas, organizacionais e legais, o documento do NIST obtém maior destaque, abordando todos os parâmetros utilizados. Para questões técnicas, o documento da ENISA apresentou o melhor resultado, abordando todos os parâmetros utilizados.

Portanto, constata-se que para aspectos legais, políticos e organizacionais é importante basear-se principalmente no documento do NIST e para questões técnicas o documento da ENISA. Entretanto, pelo fato do escopo dos parâmetros utilizados para o comparativo serem limitados, caso seja necessário, é interessante revisar todos os documentos do comparativo, a fim de verificar qual a melhor solução para a questão de segurança verificada. A subseção 3.2 apresenta a análise de segurança dos guias especializados em virtualização, tratando de um tema mais específico no cenário geral da segurança e de grande importância para a computação em nuvem.

3.2 SEGURANÇA EM VIRTUALIZAÇÃO

Um dos recursos mais importantes envolvidos na computação em nuvem é a virtualização, pois tem influência direta na redução dos custos de uma infraestrutura computacional por meio de um maior aproveitamento dos recursos físicos (IBRAHIM et. al., 2011). A virtualização diz respeito à criação de ambientes virtuais, tipicamente máquinas virtuais (SCARFONE, SOUPPAYA, HOFFMAN, 2011). As máquinas virtuais possuem um papel importante no suporte à IaaS, e com isto, também torna-se um alvo em potencial para *softwares* maliciosos. Segundo (IBRAHIM et. al., 2011) soluções tradicionais de segurança, se baseando na confiança do *kernel*[40] do sistema operacional, não são soluções efetivas para garantir a segurança da infraestrutura virtual do modelo de IaaS.

Considerando isto, esta seção realiza uma apresentação dos principais guias de segurança em virtualização de recursos, e em seguida um comparativo avaliando os pontos relatados pelos guias. Nas próximas subseções realiza-se uma curta apresentação dos seguintes guias de segurança em virtualização:

- **NIST (subseção 3.2.1)** – Guide to Security for Full Virtualization Technologies (SCARFONE, SOUPPAYA, HOFFMAN, 2011);
- **Red Hat (subseção 3.2.2)**– Virtualization Security Guide (RED HAT, 2013); e

  PCI Security Standards Council (subseção 3.2.3) – Data Security Standards (DSS) Virtualization Guideline (PCI DSS, 2011).

Também há outras obras como (IBRAHIM et. al., 2011), (TRENDMICRO, 2013), (VMWare, 2013) que podem ser utilizadas para reforçar alguns conceitos e pontos específicos relacionados a virtualização. No final desta seção é apresentado um comparativo ressaltando as diferenças entre os guias de segurança.

[40] Componente central do sistema operacional

3.2.1 GUIA DE VIRTUALIZAÇÃO DO NIST

Com o crescente uso de produtos e serviços resultantes da virtualização de recursos, o NIST elaborou um guia de segurança em virtualização por meio de (SCARFONE, SOUPPAYA, HOFFMAN, 2011), visando complementar o guia para computação em nuvens públicas, apresentando preocupações e recomendações para garantir a segurança na virtualização de recursos. A **Tabela 21** apresenta as seções e os tópicos abordados pelo guia.

Tabela 21 – Estrutura do Documento de Segurança em Virtualização do NIST.

Seção	Tópicos
1 – Introdução	Apresentação do documento, estruturação dos tópicos, apresentação do público e do propósito e escopo.
2 – Introdução a Virtualização Total	**2.1 -** Motivações para virtualização; **2.2 -** Tipos de virtualização total; **2.3 -** Virtualização de *hardware*, rede e armazenamento; e **2.4** - Casos de virtualização total (servidores e *desktop).*
3 – Visão Geral da Segurança na Virtualização	**3.1 -** Isolamento dos sistemas operacionais virtualizados; e **3.2 -** Monitoramento e gerenciamento de imagens e *snapshots*.
4 – Recomendações de Segurança para Virtualização de Componentes	**4.1 –** Segurança do Hypervisor; **4.2 –** Segurança do S.O. *Guest*; **4.3 –** Segurança da Infraestrutura Virtualizada; **4.4 –** Segurança na Virtualização de Desktops.
5 – Virtualização Segura, Planejamento e Utilização	**5.1** – Iniciação; **5.2 –** Planejamento e projeto; **5.3 –** Implementação; **5.4 –** Operações e Manutenção; e **5.5 –** Disposição.

Fonte: (SCARFONE, SOUPPAYA, HOFFMAN, 2011).

O guia (SCARFONE, SOUPPAYA, HOFFMAN, 2011) inicia apresentando uma introdução a virtualização de recursos, primeiramente com uma rápida introdução que expõe de modo direto, o conceito e tipos de virtualização que podem ser empregados tanto para servidores como para *desktops*. Em seguida, apresenta uma motivação para o uso da virtualização, apresentando alguns benefícios proporcionados por sua utilização, sendo o principal deles a grande eficiência na utilização dos recursos de *hardware*. Como uma motivação para o uso da virtualização em servidores, (SCARFONE, SOUPPAYA, HOFFMAN, 2011) diz que as organizações podem utilizar os recursos de *hardware* já existentes de maneira mais eficiente colocando mais carga em cada máquina, fazendo com que em um *hardware* funcione dois ou mais sistemas operacionais executando serviços distintos. Isto acaba exigindo do *hardware* mais poder de processamento e memória do que se estivesse executando um único sistema operacional, portanto é preciso que o *hardware* seja devidamente preparado para receber a carga de instâncias de sistemas operacionais que se propõe.

Uma segunda motivação de uso apresentada é a virtualização em *desktops*, na qual um único computador executa mais de uma instância de sistema operacional. (SCARFONE, SOUPPAYA, HOFFMAN, 2011) evidencia que existem diversas razões para o uso da virtualização em *desktops*, ela proporciona suporte para aplicações que executam em um sistema operacional específico, também permite que mudanças realizadas em um sistema operacional que afetam negativamente a segurança do sistema operacional sejam revertidas, por outro sistema operacional.

Em seguida, o documento aborda os tipos de virtualização total. Apresentando dois modos de virtualização total, *bare-metal*, conhecida como baseada no hardware, e *hosted*. Na virtualização *bare-metal* o *hypervisor* executa diretamente na camada superior ao *hardware*, sem necessitar de um sistema operacional. O *hypervisor* pode até ser implementado no *firmware* do computador. Na arquitetura de virtualização *hosted,* o *hypervisor* executa sob o sistema operacional, que pode ser qualquer um dos sistemas operacionais mais comuns (MS-Windows, GNU/Linux ou Apple MacOS-X). O item 2.3 do

Guia diz respeito à virtualização de rede, em que são descritos algumas capacidades de rede que os *hypervisors* podem prover, como:

- **Ponte de Rede (Bridge)***:* em que um sistema operacional virtualizado pode ter acesso direto a interface de rede;
- **Tradução de Endereço de Rede (NAT)**: o sistema operacional virtualizado recebe uma interface de rede virtual que é conectado a um NAT simulado no *hypervisor*; e
- **Rede Apenas para o Host**: o sistema operacional virtualizado recebe uma interface de rede virtual, que não é conectada diretamente a uma interface de rede virtual.

No armazenamento virtualizado, os sistemas de *hypervisors* possuem diversas maneiras de simular o armazenamento de disco para os ambientes virtualizados. Todos *hypervisors* possuem no mínimo discos rígidos virtuais, enquanto alguns possuem mais algumas opções de armazenamento virtual. Alguns *hypervisors* podem utilizar interfaces de armazenamento, como *network attached storage* (NAS) e *storage área networks* (SAN) apresentando diferentes opções de armazenamento para os sistemas operacionais virtualizados. No item 2.4 são descritos alguns casos de virtualização total, como “servidor único para virtualização”. “múltiplos servidores para virtualização” e a virtualização para *desktops*.

O capítulo 3 inicia com uma visão geral sobre a segurança na virtualização. Migrar recursos computacionais para um ambiente virtualizado tem pouco ou nenhum efeito no que diz respeito as suas vulnerabilidades e ameaças (SCARFONE, SOUPPAYA, HOFFMAN, 2011). Por exemplo: se um serviço que executa em um servidor não virtualizado possui vulnerabilidades, migrar ele para um ambiente virtualizado não resolve suas vulnerabilidades. No entanto, o documento descreve que a virtualização pode ajudar a reduzir a exploração destas vulnerabilidades – outro porém, é que também proporciona outros tipos de ataque. Contanto o documento determina que antes de adotar uma solução de virtualização é necessário avaliar os riscos e benefícios envolvidos para determinar a melhor solução a ser implantada. A seção 3.1

discute o isolamento de g*uests*, a camada do *hypervisor* e o sistema operacional. A seção 3.2 explica o propósito e os mecanismos para o monitoramento dos *guests*. A seção 3.3 discute o gerenciamento de imagens e *snapshots*.

O capítulo 4 trata das recomendações de segurança para os componentes de virtualização. Assim como na computação em nuvem, que é o resultado da junção de diversas tecnologias (CASTRO et. al, 2012), a virtualização também é o resultado da junção de algumas tecnologias, e por isto sua segurança depende da segurança individual de cada um destes componentes, como *hypervisors*, os *hosts*, os g*uests*, aplicações, entre outros. A seção 4.1 diz respeito a segurança do *hypervisor* descrevendo algumas recomendações de segurança para o componente, como:

- Instalar todos os *updates* de segurança lançados pelo fornecedor;
- Restringir acesso administrativo para a interface de gerenciamento do *hypervisor*;
- Sincronizar a infraestrutura virtualizada para um *time server* confiável;
- Desabilitar todos serviços como “copiar e colar”, compartilhamento de arquivos entre *host* e *guests*, entre outros serviços a menos que estes sejam necessários. Estes serviços podem gerar probabilidades de ataque ao *hypervisor;*
- Monitorar cuidadosamente o *hypervisor* observando por sinais de comprometimento de sua integridade. E outras recomendações.

A seção 4.2 descreve recomendações para os g*uests*. O guia descreve que todas as considerações de segurança aplicadas a um SO que executa em um *hardware* real, também devem ser aplicadas ao *guest*, mas, no entanto devem existir algumas considerações a mais para estes SO’s. Para executar em uma máquina virtual, um SO pode utilizar vídeo, som, teclado, *mouse* e *drivers* de rede que são específicas ao *hypervisor.* Não existem problemas específicos de segurança com estes *drivers* a menos que estes possuam *bugs* de programação que não estão presentes nos *drivers* normais.

Outro ponto importante destacado pelo documento, é que em muitas virtualizações é permitido ao SO visitante trocar informações com o SO *host,* por meio de discos ou diretórios compartilhados. Em caso de um SO estar infectado com um *malware*, este pode se espalhar pelo diretório ou disco compartilhado. Este é um tipo de vulnerabilidade que é específica aos ambientes virtualizados e merece atenção do administrador de rede. Mais algumas recomendações são feitas na seção:

- Seguir as recomendações práticas para gerenciar o SO físico, e.g., sincronização de tempo, gerenciamento de *logs*, autenticação, entre outros;
- Instalar todos *updates* para os g*uests* rapidamente;
- Realizar *backup* dos *drivers* virtuais utilizados pelos g*uests* regularmente, seguindo a mesma política para os ambientes não virtualizados; e
- Utilizar diferentes soluções de autenticação para cada SO visitante, a menos que exista uma razão para que dois SO's compartilhem suas credenciais; etc.

A seção 4.3 discorre sobre a segurança da infraestrutura virtualizada. A infraestrutura de virtualização está relacionada à simulação de mecanismos de armazenamento e rede (SCARFONE, SOUPPAYA, HOFFMAN, 2011), esta infraestrutura virtual é tão importante quanto uma infraestrutura do ambiente físico. Muitos sistemas de virtualização possuem dispositivos que fornecem controle de acesso ao *hardware* virtual, particularmente os de armazenamento e rede. O documento recomenda que o acesso a estes dispositivos virtuais deve ser estritamente realizado pelos dispositivos virtuais que vão utilizá-los. Por exemplo, se um disco rígido virtual é compartilhado por dois SO's virtuais, então apenas os dois SO's devem ter acesso a este disco rígido. E para os dispositivos virtuais de rede algumas recomendações também são realizadas, como estabelecer políticas (semelhantes às redes físicas), realizar constantemente monitoramento entre outras tarefas.

O item 4.4 alerta sobre a segurança referente à virtualização em *desktops*, estabelecendo que a principal diferença entre a virtualização em servidores e em *desktops* é a capacidade de gerenciamento de imagens. No ambiente de servidores o gerenciamento das imagens é usualmente limitado para administradores, enquanto que em um ambiente de *desktop*, os usuários finais podem criar, modificar, duplicar e excluir imagens. As recomendações são feitas para organizações que desejam adotar este tipo de virtualização, delimitando que estas devem estar atentas, sob o prisma da segurança, em quais cenários podem ser aplicadas estas soluções descentralizadas (de *desktop)* de virtualização e em quais podem ser aplicadas as soluções centralizadas (de servidores).

O capítulo 5 apresenta cinco fases de planejamento e abordagem para garantir uma utilização segura da virtualização. As cinco fases para o planejamento e abordagem da virtualização são:

1. **Iniciação**: inclui tarefas que uma organização deve realizar para projetar/planejar a solução de virtualização, identificando os requisitos, desenvolvendo uma política de virtualização, identificando plataformas e aplicações que podem ser virtualizadas, entre outras tarefas;
2. **Planejamento e projeto**: especificar as características técnicas da solução de virtualização e os componentes relacionados. Isto inclui os métodos de autenticação e mecanismos de cifragem utilizados para proteger a comunicação;
3. **Implementação**: nesta fase, os equipamentos são configurados para estarem de acordo com as requisições de segurança estabelecidas, com um protótipo instalado e devidamente testado, e então ativado e colocado em produção. A implementação inclui alterar a configuração de outros controles de segurança e tecnologias como, eventos de log de segurança, gerenciamento de rede e integração dos servidores de autenticação;
4. **Operação e manutenção**: esta fase inclui as tarefas de segurança relacionadas que uma organização deve realizar constantemente, como revisão de logs, detecção de ataques, entre outras operações de monitoramento;

5. **Disposição**: esta fase reforça as tarefa que ocorrem quando uma solução de virtualização está sendo removida, isto inclui preservação das informações para estarem de acordo com os requerimentos legais, sanitização adequada dos dados, e descarte apropriado para os equipamentos.

Estas fases definidas no documento, apresentadas resumidamente neste trabalho, definem os cuidados e considerações que devem ser dispensadas a cada etapa para a implantação, utilização e descarte de uma determinada solução de virtualização visando garantir a segurança dos dados armazenados na infraestrutura virtualizada. O documento ainda expõe que estas fases representam apenas uma abordagem geral para a utilização segura do recurso de virtualização, e que existem diversas outras opções complementares que podem ser utilizadas em cada uma das etapas definidas.

3.2.2 GUIA DE VIRTUALIZAÇÃO RED HAT

A Red Hat é uma empresa norte americana reconhecida por sua distribuição de GNU/Linux, o Red Hat Linux e diversos outros produtos como uma distribuição própria do OpenStack. Estes produtos são voltados tanto para empresas quanto para usuários, e para oferecer suporte a seus produtos a Red Hat elabora guias e manuais visando o uso seguro de seus produtos (RED HAT, 2013). Neste sentido, a Red Hat elaborou um guia de segurança em virtualização para dar suporte a seus produtos de virtualização. A **Tabela 22** apresenta a estrutura contida no documento.

Tabela 22 – Estrutura do Documento de Segurança em Virtualização da Red Hat.

Seção	Tópicos
1 - Introdução	**1.1 –** Ambiente virtualizado e não virtualizado; **1.2 –** Porque segurança de virtualização é importante.
2 – Segurança do Host	**2.1 –** Porque é importante a segurança do *host*; **2.2 –** Práticas recomendadas para a versão do Red Hat Enterprise Linux; **2.3 –** Práticas recomendadas para a segurança do Host.
3 – Segurança dos Guests	**3.1 –** Porque é importante a segurança dos Guests; **3.2 –** Práticas de segurança recomendadas.
4 – Segurança de rede em um ambiente virtualizado	**4.1 –** Visão geral; **4.2 –** Práticas recomendadas; **4.3 –** Segurança de conectividade; **4.4 –** Segurança de armazenamento.

Fonte: (RED HAT, 2013).

O guia inicia explicitando as oportunidades para a descoberta de vulnerabilidades em ambientes virtualizados e não virtualizados, destacando que em ambientes virtualizados há uma maior possibilidade de tipos de ataques, apresentando novas vulnerabilidades além das existentes no ambiente não virtualizado. De maneira geral, a seção de introdução destaca a importância de se assegurar a confiança nos ambientes virtualizados, tomando medidas de segurança tanto para os *hosts* físicos e os g*uests*. A seção 1.2 desta as considerações (riscos à) de segurança no ambiente virtualizado, elas são (RED HAT, 2013):

- O *host/hypervisor* se torna o principal alvo. De fato, estes são os únicos pontos de falha para os *guests* e seus dados;
- VM's podem interferir uma com a outra de maneiras indesejáveis;

- Recursos e serviços podem se tornar difíceis de se rastrear e manter. Com a rápida adoção dos sistemas virtualizados necessita-se de um aumento do gerenciamento de recursos, incluindo lançamento de *patches*, monitoramento e manutenção;
- Recursos como armazenamento podem estar espalhados por diversas máquinas, e dependentes uma da outra. Isto pode tornar o sistema altamente complexo, gerando uma enorme dificuldade de se gerenciar e manter; e
- A virtualização não remove nenhum dos riscos de segurança apresentados pelos ambientes tradicionais, a pilha inteira de solução de virtualização, não só uma camada, deve estar segura.

O capítulo 2 trata da segurança do *host* destacando que ao se implantar as tecnologias de virtualização, a segurança do *host* deve ser um fator primordial, pois os *guests* são tão seguros quanto seu *host*. Neste sentido a segurança dos g*uests* depende da segurança do *host.* A seção 2.2 e 2.3 listam as práticas recomendadas (RED HAT, 2013):

- Execute apenas os serviços recomendados para dar suporte ao uso e gerencia dos g*uests*;
- Limite o acesso direto ao sistema para apenas aqueles usuários que tem necessidade de gerenciar o sistema;
- Garantir que seja possível realizar auditorias no sistema; e
- Garantir que qualquer acesso remoto ao sistema seja realizado apenas com canais seguros. Ferramentas como SSH e protocolos de rede como TLS ou SSL fornecem autenticação e cifragem dos dados para ajudar a garantir que apenas administradores aprovados possam gerenciar o sistema.

Ainda são destacadas algumas práticas que devem ser realizadas considerando-se a abordagem de nuvens públicas. O documento destaca que as nuvens públicas estão expostas a um número de riscos à segurança muito

além da virtualização tradicional. O isolamento entre o *host* e os g*uests* é crucial devido às ameaças proporcionadas pelo grande número de usuários da nuvem e os requisitos de segurança sobre os dados, como confidencialidade e integridade em toda a infraestrutura de virtualização. Algumas práticas recomendadas pelo guia para a segurança de virtualização em nuvens públicas (RED HAT, 2013):

- Desabilitar qualquer acesso direto de *hardware* partindo dos visitantes; e
- Tráfego de rede deve ser separado tal que os operadores de nuvem privada possam ter uma rede de gerenciamento isolada dos usuários da rede, ajudando a garantir que os g*uests* não possam acessar os sistemas do *host* pela rede.

A terceira seção do guia, diz respeito à segurança dos *guests.* O documento diz que enquanto a segurança do *host* é crucial para garantir a segurança dos SO's executando no *host*, isto não remove a necessidade de segurança individual para cada SO visitante. Ainda é explicitado que todos os riscos de segurança associados com um sistema convencional, não virtualizado ainda existem quando o sistema é executado como um SO virtual. Para isto, qualquer recurso acessível por este SO virtual, como dados críticos de negócios, ou qualquer outro dado relevante pode estar vulnerável se um dos SO's virtuais estiverem comprometidos. Algumas das práticas elencadas pelo documento são (RED HAT, 2013):

- Com todo gerenciamento do *guest* sendo realizada remotamente, deve-se garantir que o gerenciamento do sistema seja realizado por canais seguros. Ferramentas como SSH e protocolos de rede como SSL e TLS devem ser utilizadas;
- Algumas tecnologias de virtualização utilizam agentes virtuais especiais ou *drivers* para permitir alguns recursos especiais para a virtualização. Deve-se garantir que estas aplicações sejam seguras; e
- Em ambientes virtualizados existe um grande risco de dados relevantes serem acessados de fora dos limites de proteção do SO visitante. Estes

dados devem ser protegidos utilizando técnicas de cifragem como *dm-crypt* e GnuPG.

O capítulo 4 finaliza o guia tratando sobre a segurança de rede em um ambiente virtualizado. A seção inicia realizando uma introdução sobre a importância da rede nestes sistemas, de maneira clara, pois a rede é o único meio de acessar sistemas, aplicações e interfaces de gerenciamento. A rede possui um papel crítico no gerenciamento de sistemas virtualizados e a disponibilidade das suas aplicações hospedadas é outro fator importante. Portanto, torna-se um fator crucial garantir a disponibilidade dos canais de segurança do *host* para os sistemas virtualizados e vice-versa. As práticas de segurança recomendadas são para assegurar os canais de comunicação utilizando protocolos de rede seguros (práticas já citadas para a segurança do *host* e dos g*uests*), como também garantir que as aplicações que estejam transferindo dados relevantes o façam por meio de canais seguros, configurar *firewalls* e garantir que estes estejam ativados somente para as portas necessárias, realizar testes e revisões de segurança para o *firewall*, políticas, entre outras.

3.2.3 GUIA DE SEGURANÇA EM VIRTUALIZAÇÃO DA PCI SECURITY STANDARDS COUNCIL

A PCI DSS (Payment Card Industry - Data Security Standard) é uma organização proprietária de segurança da informação que presta serviços para empresas que realizam operações com cartões de crédito nos Estados Unidos (Visa, Mastercard, American Express, entre outras). O objetivo desta organização é garantir a segurança da informação nas tecnologias que estas transações utilizam, dentre elas estas a virtualização. O guia é dividido em quatro partes, como apresenta a **Tabela 23**.

Tabela 23 – Estrutura do Documento de Segurança em Virtualização da PCI.

Seção	Tópicos
1 - Introdução	**1.1 –** Público; **1.2 –** Uso do guia.
2 – Visão geral de virtualização	**2.1 –** Virtualização: conceitos e classes; **2.2 –** Componentes da virtualização e escopo do guia.
3 – Riscos para ambientes virtualizados	**3.1 –** Vulnerabilidades no ambiente físico aplicado em um ambiente virtual; **3.2 –** Vulnerabilidades proporcionadas pelo *hypervisor;* **3.3 –** Aumento da complexidade dos sistemas virtualizados e redes; **3.4 –** Mais de uma função por sistema físico; **3.5** – Mistura de VM's de diferentes níveis de confiança; **3.6 –** Falta de separação de funções; **3.7 –** VM's em estado de espera; **3.8 –** Imagens e Snapshots; **3.9 -** Imaturidade das funções de monitoramento; **3.10 -** Vazamento de informações entre segmentos virtuais de rede.
4 – Recomendações	**4.1 –** Recomendações gerais; **4.2 –** Recomendações para ambientes mistos; **4.3 –** Recomendações para ambientes de computação em nuvem; **4.4 –** Guia para avaliar riscos em ambientes virtuais.

Fonte: (PCI DSS, 2011).

O guia inicia com uma introdução dividida em duas seções: público alvo e como o guia deve ser utilizado por este público alvo. O público alvo é de provedores de serviços que utilizam ou devem utilizar as tecnologias de virtualização em seus ambientes. O documento provê um guia suplementar para a utilização da virtualização, definindo alguns riscos de segurança e realizando recomendações.

O segundo capítulo introduz os conceitos gerais de virtualização, como por exemplo: a definição de virtualização (PCI DSS, 2011): a virtualização

refere-se a abstração lógica de recursos computacionais físicos. Também elucida que uma das mais comuns abstrações de virtualização existentes é a máquina virtual, que utiliza recursos de uma máquina física para operar um SO virtual. Em uma adição as VM's, é esclarecido que o uso da virtualização pode ser realizado de diferentes maneiras, como virtualização de rede, armazenamento, SO's, memória, entre outros. A segunda parte do capítulo (2.2) apresenta os componentes envolvidos na virtualização, que são: Hyprevisor, máquina virtual e aplicações virtuais (PCI DSS, 2011):

A terceira seção deste guia apresenta os riscos de segurança proporcionados pelo ambiente de virtualização. O primeiro risco elencado (subseção 3.1) é proveniente das vulnerabilidades no ambiente físico em que é aplicada a virtualização. Segundo o guia, os sistemas virtuais estão sujeitos aos mesmos ataques e vulnerabilidades que existem na infraestrutura física. Uma aplicação que tem falhas em sua configuração ou é vulnerável a *exploits* ainda terá as mesmas falhas e vulnerabilidades quando for instalada uma implementação virtual. O item 3.2 destaca que o *hypervisor* cria uma nova superfície de ataque, pois proporciona um único ponto de acesso a todo ambiente virtual, e se está comprometido, todas as VM's hospedadas naquele *hypervisor* também estarão. Um dos pontos de vulnerabilidade destacados pode ser uma simples má configuração do *hypervisor*, um erro muito comum que pode ter grande influência na segurança de todas VM's gerenciadas pelo *hypervisor*. Mais algumas vulnerabilidades destacadas no documento são: isolamento fraco, controle de acesso a recursos físicos, endurecimento da segurança e *patches*, potenciais falhas nestes pontos podem ser explorados e assim proporcionar um risco a segurança das VM's. É crítico que o acesso ao *hypervisor* esteja restrito somente aos administradores da rede, e tarefas como revisões e monitoramentos devem ser realizadas constantemente como se estivessem sendo operadas em um ambiente físico.

O item 3.3 trata do aumento da complexidade dos sistemas proporcionados pela virtualização. As configurações virtuais podem ser classificadas como sistemas e redes (PCI DSS, 2011). Por exemplo: VM's podem transmitir dados umas as outras por meio do *hypervisor*, como também sobre uma conexão de rede virtual e/ou por meio de aplicações de segurança

de redes virtuais, como um *firewall*. Enquanto as configurações virtuais oferecem diversos benefícios, estas abstrações adicionais de *software* causam um aumento na complexidade total do sistema, exigindo um cuidado maior em seu gerenciamento. Estas camadas, segundo o documento, exigem um cuidado adicional de segurança com uma política de segurança igualmente complexa para garantir a segurança do sistema.

O item 3.4 apresenta os riscos decorrentes da presença de mais de uma função por sistema físico. Uma preocupação destacada pelo documento neste quesito é, se uma função de um ambiente virtual particular está comprometida ela então poderá comprometer outras funções no mesmo ambiente físico. Explicando ainda este risco o documento apresenta um exemplo: uma VM comprometida pode utilizar os mecanismos da camada virtual de comunicação para lançar ataques em outras VM's no mesmo *host* ou até no *hypervisor.* As tecnologias de virtualização devem mitigar alguns riscos através do reforço do processo de separação entre diferentes funções. Até então, o risco associado com a alocação de múltiplas funções ou componentes em um único sistema físico ainda deve ser considerado.

A subseção 3.5 descreve os riscos decorrentes da mistura de VM's com diferentes níveis de confiança em um *host* em particular. O documento destaca que este é um dos desafios quando se planeja a virtualização de um sistema, identificar as configurações apropriadas para a variedade de tecnologias de virtualização. O risco de hospedar VM's de diferentes níveis de confiança no mesmo *host* precisa ser cuidadosamente avaliado. No contexto virtual, uma VM com baixo nível de confiança tipicamente tem um nível baixo de segurança se comparada ama VM de alto nível de confiança, portanto estas VM's com baixo nível de confiança podem ser mais fáceis de estarem comprometidas se misturadas com VM's de níveis superiores de confiança.

No item 3.6, o risco descrito pelo documento é a falta de separação de tarefas para as VM's. É descrito como um desafio definir papéis granulares (por exemplo, separação de administrador de rede e administrador de servidor), e politicas de acesso em um sistema distribuído e virtualizado. O risco de falhar em definir propriamente estes papéis e as políticas é significante, pois o acesso

impróprio ao *hypervisor* pode potencialmente fornecer amplo acesso a componentes importantes da infraestrutura.

O item 3.7 trata das VM's em estado de espera, o guia explica que em várias plataformas de virtualização as VM's podem existir em estados ativos e dormentes (em espera). VM's que não estão ativas ainda podem abrigar dados importantes como credenciais de autenticação, chaves de cifragem, ou informações críticas de configuração.

A subseção 3.8 do guia discorre sobre as imagens de VM e *snapshots*. Estas imagens e *snapshots* fornecem uma maneira rápida de abordar ou restaurar sistemas virtuais por meio de múltiplos *hosts* em um período curto de tempo. Uma atenção especial deve ser dada a preparação das imagens e *snapshots* das VM's, como estas podem capturar dados importantes presentes no sistema no momento que a imagem foi criada, isto inclui conteúdos ativos em memória (PCI DSS, 2011). Adicionalmente, se as imagens não estiverem propriamente seguras de uma modificação, um atacante pode ganhar acesso e inserir vulnerabilidades ou códigos maliciosos na imagem. A imagem comprometida pode ser utilizada por todo o ambiente, resultando em um comprometimento rápido de todos os *hosts*.

A subseção 3.9 descreve os riscos decorrentes da imaturidade das soluções de monitoramento. No mesmo momento que aumenta a necessidade de ferramentas para efetuar *logs* e monitoramento em ambientes virtualizados, é igualmente reconhecido que as ferramentas para efetuar isto nestes ambientes não evoluíram da mesma maneira (PCI DSS, 2011). O documento estabelece que se comparado às ferramentas tradicionais para monitorar uma rede física, as ferramentas para sistemas virtuais podem não fornecer o mesmo nível para monitorar as comunicações ou o fluxo de tráfego entre as VM's em uma rede virtual. De maneira similar, ferramentas especializadas de monitoramento e de geração de *logs* para ambientes virtuais tornam-se necessárias para realizar a captura em um detalhamento necessário para o ambiente virtual.

O item 3.10 detalha o risco de vazamento de informação entre segmentos de rede virtuais. O vazamento de informações no planejamento de

dados resulta em dados existentes fora de locais conhecidos (PCI DSS, 2011), contornando os controles de proteção de dados que seria aplicada. O vazamento de informações no plano de controle ou plano de gerenciamento pode ser explorado para permitir o vazamento de informações no plano de dados, ou para influenciar rotas de rede para passar por controles de segurança de rede.

O capítulo 4 trata das recomendações gerais para garantir a segurança do ambiente virtualizado. A primeira subseção diz respeito a avaliação dos riscos associados com as tecnologias de virtualização, esclarecendo que as entidades devem saber cuidadosamente avaliar os riscos de se virtualizar cada componente antes de selecionar e implementar uma solução de virtualização. Em seguida, trata da restrição de acesso físico aos *hosts*, falando sobre os cuidados necessários para evitar que um atacante ganhe acesso físico a um *host* que hospeda múltiplas VM's. Implementar a defesa em profundidade[41] é segue a lista de recomendações, de maneira similar a um ambiente físico, a defesa em profundida é uma prática comum para assegurar os dados e outros ativos. Outra recomendação descrita no documento é isolar as funções de segurança, descrevendo que as funções de segurança implementadas na VM devem seguir o mesmo processo de separação existente no ambiente físico. É especialmente recomendado que este requisito seja implementado em ambientes virtualizados, pois esta medida dificulta significantemente as chances de um atacante comprometer os componentes de um sistema virtual. As recomendações seguem para uma avaliação adequada do *hypervisor*, os cuidados necessários nas configurações desse componente vital no ambiente virtual, utilizar ferramentas adequadas de monitoramento e geração de *logs*, entre outras recomendações.

Fechando o guia, a seção 4.4 descreve medidas para avaliar os riscos no ambiente virtualizado. O primeiro é definir o ambiente, identificar todos componentes, incluindo *hypervisors*, *hosts*, redes, consoles de gerenciamento entre outros, o seguindo as medidas, deve-se identificar potenciais ameaças e mapear as vulnerabilidades existentes no sistema.

---

[41] Consiste em utilizar diversos controles de segurança complementares ao invés de um só.

3.2.4 COMPARATIVO NIST, RED HAT E PCI

Para efetuar o comparativo dos guias de segurança em virtualização é necessário observar os pontos descritos em seus documentos para constatar as diferenças. A primeira diferença observada é que não houve um padrão bem definido para a descrição dos pontos de segurança pelos guias, como ocorrido no comparativo **3.1.4**, dos guias de segurança em computação em nuvem. Os guias descrevem os principais problemas de segurança relacionados à virtualização, entretanto, apenas o guia da (RED HAT, 2013) os descreveram focando as soluções em seus produtos. Os guias do NIST (SCARFONE, SOUPPAYA, HOFFMAN, 2011) e (PCI DSS, 2011), por serem guias realizados por organizações fazem menos referência aos fins de suas organizações e descrevem os pontos de segurança de uma maneira mais abrangente. Para realizar este comparativo é descrito na **Tabela 24** os pontos abordados pelos guias, facilitando a observação dos pontos em comum e das diferenças entre os guias.

Tabela 24 – Comparativo NIST, RED HAT e PCI

Pontos Abordados	NIST	RED HAT	PCI
1 – Segurança do *hypervisor.* (Monitoramento/logs/ Configuração/ restrições de acesso administrativo)	Abordado	Abordado	Abordado
2 – Segurança da infraestrutura virtual	Abordado	Abordado	Abordado
3 - Mistura de VM's de diferentes níveis de confiança no mesmo *host*	N.A	N.A	Abordado
4 - Segurança em virtualização de servidores	Abordado	Abordado	Abordado

Pontos Abordados	NIST	RED HAT	PCI
5 - Segurança em virtualização de *desktops*	Abordado	N.A	Abordado
6 - Vazamento de informações entre segmentos virtuais de rede.	Abordado	N.A	Abordado
7 - Imaturidade das funções de monitoramento	N.A	N.A	Abordado
8 – Segurança na geração de Imagens e Snapshots	N.A	N.A	Abordado
9 – Isolamento de Rede	Abordado	Abordado	Abordado
10 – Proteção da Imagem da VM	Abordado	N.A	Abordado
11 – Segurança dos dados da VM	Abordado	Abordado	Abordado
12 - Segurança das API's do hypervisor	N.A	N.A	N.A

Fonte: próprio autor.

A **Tabela 24** apresenta algumas diferenças entre os guias de segurança em virtualização. É possível observar que o documento que aborda mais tópicos, é o da PCI, seguido pelo NIST e o mais básico é o Red Hat. Os pontos críticos em comum abordados pelos guias são os de maior importância na virtualização, como: segurança do *hypervisor*, segurança dos *Guests* e segurança da infraestrutura virtual. O *hypervisor* é destacado nos documentos como um componente fundamental na virtualização, pois tem a função de coordenar as VM's e permitir acesso destas ao *hardware* físico, portanto se este componente estiver comprometido a chance de todas as VM's naquele

host estarem comprometidas é grande. Segundo os três guias, os *Guests* devem estar com uma configuração de semelhante a um SO físico, porque além de compartilharem as mesmas vulnerabilidades, ainda existem riscos provenientes do ambiente virtual, portanto, devido a esta complexidade, deve dispensar-se um cuidado maior para garantir a segurança das VM's. A infraestrutura virtual, outro ponto destacado pelos três guias, envolve garantir a segurança da rede virtual, a qual deve estar com os canais de comunicação seguros. Estes três pontos, são os grandes entraves para a segurança do ambiente virtual, entretanto, para garantir um alto nível de segurança é necessário assegurar diversos outros aspectos, e o guia da (PCI DSS, 2011) descreve com detalhamento diversos pontos críticos, como: imagens e *snapshots*, imaturidade das ferramentas de geração de *logs* e monitoramento e mistura de VM's com diferentes níveis de confiança em um mesmo *host*.

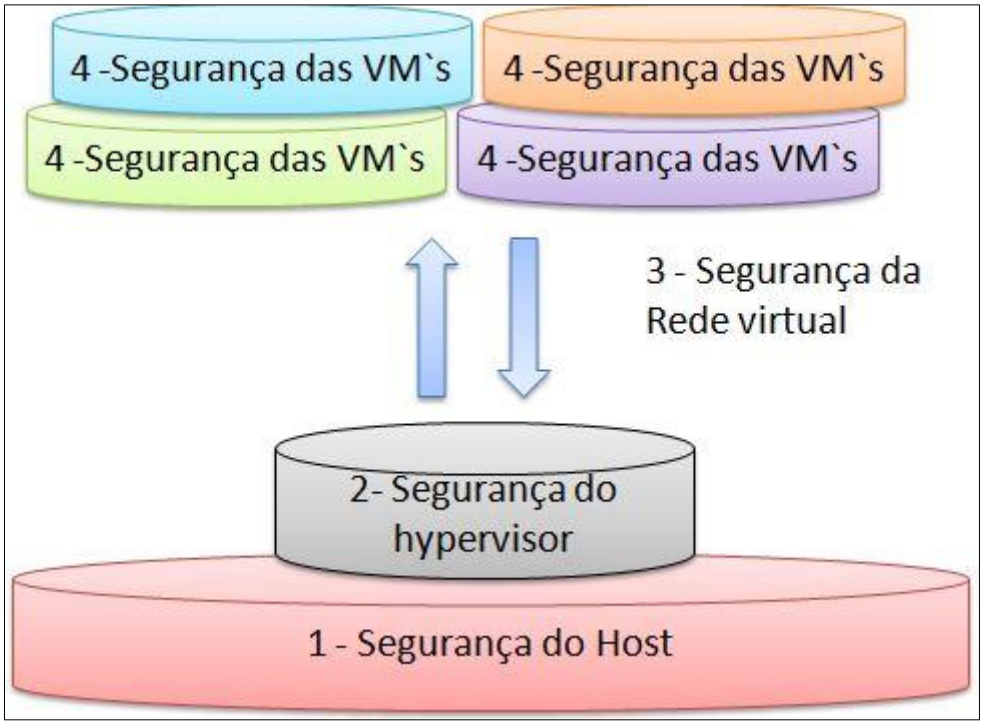
3.2.5 CONSIDERAÇÕES SOBRE O COMPARATIVO

Por meio deste comparativo é possível observar os principais pontos de segurança elencados pelos guias, e a segurança do *hypervisor* como o maior ponto de preocupação estabelecido pelos guias. A ausência de parâmetros fixos como no último comparativo se deve a ausência de padronização dos documentos para estabelecer estes pontos de preocupação, considerando isto os pontos elencados para o comparativo foram os pontos observados nos guias. Considerando esta listagem dos pontos abordados é possível comparar quais pontos foram abordados por todos os documentos e quais pontos foram abordados por somente um documento, permitindo definir quais pontos são mais relevantes para a segurança (no caso de todos os guias abordarem este tema) e qual guia de segurança é mais completo (o guia que abordou mais pontos) ou não.

Neste sentindo, utilizando estes parâmetros de comparação, o guia (PCI DSS, 2011) apresenta-se o mais completo, descrevendo com detalhamento

diversos pontos de segurança, como também recomendações para prevenir estas vulnerabilidades. O guia do NIST (SCARFONE, SOUPPAYA, HOFFMAN, 2011), aparece como o segundo guia mais completo, abordando assuntos englobados para virtualização total, tanto para servidores quanto para *desktops*. O guia da (RED HAT, 2013) foi o guia mais simples em comparação com os demais, pois apresentou somente os principais pontos de segurança e recomendações para prevenção. Como resultado deste comparativo observa-se a **Figura 20**, com os principais pontos críticos para a criação de um ambiente seguro de virtualização.

Figura 20 – Principais Pontos de Segurança nas VM's



Fonte: próprio autor.

A **Figura 20** destaca os pontos críticos de segurança ao se planejar e implementar um ambiente virtual. Primeiramente, os guias deram ênfase à segurança do *host*, que hospeda toda a infraestrutura virtual e sua segurança é vital para o sucesso da virtualização, diversos cuidados foram recomendados, desde monitoramento a restrições de acesso as máquinas. O *hypervisor* é o componente mais relevante na infraestrutura virtual, pois orquestra todas as

VM's, seguido dos canais de comunicação, com recomendações para o uso de cifras e protocolos seguros de transmissão, e a segurança das VM's. A subseção **3.2.5.1** dá ênfase a cifragem dos dados, que está relacionado com a segurança do *hypervisor,* das VM's e canais de comunicação.

3.2.5.1 PONTOS RELEVANTES AO TRABALHO ABORDADOS NO COMPARATIVO

As próximas subseções descrevem os pontos abordados no comparativo que podem ser mais relevantes ao desenvolvimento do trabalho. Estes pontos são isolamento de VM's (**3.2.5.2**) e os *hypervisors* (**3.2.5.3**). Os principais *hypervisors* suportados pelo OpenStack serão abordados detalhadamente nas subseções posteriores **3.3**.

3.2.5.2 ISOLAMENTO

O isolamento é um forte requisito de segurança para os ambientes virtualizados, pois as VM's compartilham o mesmo ambiente físico e sem o isolamento adequado elas podem intencionalmente ou não capturar dados de outras VM's, consumir grande parte do tráfego de rede, realizar ataques a outras VM's entre outras coisas (SCARFONE, SOUPPAYA, HOFFMAN, 2011). É necessário garantir que as VM's executem isoladamente em diversos aspectos como:

- **Isolamento de rede:** o isolamento de rede é necessário para garantir que uma VM não utilize toda a banda de rede virtual, capture dados de outras VM's, entre outros problemas que envolvem os conceitos de segurança de rede virtual (utilizando principalmente recursos de criptografia) e o controle e monitoração do uso dos recursos;

- **Armazenamento:** necessário para que VM's não tenham acesso a dados armazenados por outras VM's. Os dados armazenados por uma VM devem ser cifrados e utilizar mecanismos de autenticação para garantir a segurança dos dados;
- **Processamento de instruções:** o principal responsável por garantir o isolamento das instruções das VM's é o *hypervisor*, pois este gerencia as requisições das VM's permitindo o acesso destas aos recursos físicos do *host*. Segundo (JAEGER, 2007) cada processador possui um conjunto de instruções, com prioridades para determinadas instruções. Considerando isto, o *hypervisor* gerencia as instruções das VM's favorecendo o acesso ao *hardware* das instruções com maior nível de prioridade.

Estes são os aspectos mais importantes de isolamento para garantir a segurança do ambiente virtualizado. Como destacado pelos guias, o *hypervisor* tem papel fundamental para a segurança do ambiente virtual, na subseção **3.2.5.4** são apresentadas as principais soluções de virtualização suportadas pelo OpenStack Grizzly.

3.2.5.3 VISÃO GERAL DOS *HYPERVISORS* SUPORTADOS PELO OPENSTACK

O *hypervisor* é uma plataforma de *software* que realiza o gerenciamento das VM's, permitindo o acesso destas ao *hardware* físico. Grande parte dos aspectos relacionados à segurança do ambiente virtual também é de responsabilidade do *hypervisor*, como o aspecto do isolamento observado na subseção 3.2.5.3. As soluções de virtualização apresentadas nesta subseção são as principais suportadas pelo OpenStack, segundo definição do próprio

site. Os *hypervisors* são divididos em três grupos em nível de compatibilidade com a plataforma (OPENSTACK HYPERVISORS, 2013):

- **Grupo A**: os *drivers* são totalmente suportados. *Hypervisors*: libvirt qemu/KVM em x86;
- **Grupo B**: os *drivers* não foram totalmente testados. *Hypervisors*: XenAPI em x86;
- **Grupo C**: os *drivers* foram minimamente testados ou não foram testados. *Hypervisors*: VMWare, baremetal, docker, Hyper-V, libvirt (LXC, Xen), PowerVM e ARM.

Destes *hypervisors* apresentados pelo OpenStack, os três principais são: qemu/KVM, Xen e VMWare. Segundo (OPENSTACK HYPERVISORS, 2013) a maior parte do desenvolvimento da plataforma foi realizado utilizando os *hypervisors* KVM e Xen, significando que há um maior suporte da comunidade para os problemas relacionados a estes *hypervisors*. As próximas tabelas (**Tabela 25**, **Tabela 26**, **Tabela 27**,**Tabela 28** e **Tabela 29**) elencam as características de cada *hypervisor* suportado pelo OpenStack divididos nas seguintes categorias: ciclo de vida da VM, características de rede, monitoramento, características técnicas e segurança. Legenda: o símbolo ✔ indica que o OpenStack suporta a característica, e ✗ indica que a característica não é suportada. As células com N.T indicam que as características ainda não foram testadas.

.

Tabela 25 – Ciclo de Vida da VM nos *Hypervisors* suportados pelo OpenStack

Ciclo de Vida da VM								
Características	**Xen**	**qemu/KVM**	**VMWare ESXi**	**LXC via llibvirt**	**Hyper-V**	**Baremetal**	**PowerVM**	**Docker**
Criar /Finalizar VM's	✓	✓	✓	✓	✓	✓	✓	✓
Reiniciar VM's	✓	✓	✓	✓	✓	✓	✓	✓
Pausar / Continuar VM's	✓	✓	✗	✓	✓	✓	✗	✓
Ajustar tamanho de VM's	✓	✓	N.T	N.T	✓	N.T	✓	✗
Suspender / Resumir VM's	✓	✓	✓	N.T	✓	N.T	✗	✗

Fonte: (OPENSTACK HYPERVISORS, 2013).

A **Tabela 25** apresenta as operações básicas que o *hypervisor* deve executar sobre as VM's, por este motivo é observado que quase todas as operações básicas são suportadas pelos *hypervisors* no OpenStack. As operações de criação/finalização e reinicialização das VM's é suportada por todos, a operação de pausar e continuar as VM's somente não é realizada pelo VMWare e pelo PowerVM. As operações de ajuste de tamanho e suspensão das VM's não foram testadas para todos *hypervisors,* apenas o Xen qemu/KVM

e Hyper-V realizam todas as características apresentadas na tabela. A **Tabela 26** apresenta as características de rede.

Tabela 26 – Características de Rede nos *Hypervisors* Suportados pelo OpenStack

Rede								
Características	**Xen**	**qemu/KVM**	**VMWare ESXi**	**LXC via llibvirt**	**Hyper-V**	**Baremetal**	**PowerVM**	**Docker**
Rede VLAN	✓	✓	✓	✓	✗	N.T	✗	N.T
Rede Flat	✓	✓	✓	✓	✓	✓	✓	N.T
Roteamento de tráfego entre VLANS	✓	✓	✓	N.T	✗	N.T	N.T	N.T

Fonte: (OPENSTACK HYPERVISORS, 2013).

As redes VLAN são as redes virtuais criadas pelo *hypervisor* para as VM's, o Hyper-V e o PowerVM não suportam a criação de VLAN's, os demais suportam a criação destas VLAN's exceto o Baremetal e o Docker que não foram testados no OpenStack. As redes flat são aquelas redes em que cada nó é conectado diretamente no outro, todos os *hypervisors* exceto o Docker suportam esta característica. E o roteamento de tráfego entre VLANS diz respeito a troca de informações entre as sub-redes criadas, somente o Xen, qemu/KVM e o VMWare suportam esta característica, enquanto que o Hyper-V não suporta e as demais não foram testadas. A **Tabela 27** apresenta as características para o monitoramento das VM's executadas no *hypervisor*.

Tabela 27 – Características de Monitoramento nos *Hypervisors* suportados pelo OpenStack

Monitoramento								
Características	**Xen**	**qemu/KVM**	**VMWare ESXi**	**LXC via llibvirt**	**Hyper-V**	**Baremetal**	**PowerVM**	**Docker**
Saída Serial de Console	✓	✓	✓	✗	✓	✓	N.T	✓
Console SPICE	✗	✓	✗	N.T	✗	✗	✗	✗
Coletar informações dos *Guests*	✓	✓	✓	✓	N.T	N.T	✓	✓
Coletar informações do *Host*	✓	✓	✓	✓	N.T	N.T	✓	✓

Fonte: (OPENSTACK HYPERVISORS, 2013).

A **Tabela 27** apresenta algumas características importantes para realizar o monitoramento das VM's executadas sob o *hypervisor. O* console SPICE (*Simple Protocol for Independent Computing Environments*) é um protocolo que permite a interação com dispositivos virtualizados, possibilitando o acesso remoto desta VM para um *host* físico por meio de um *front-end* para os usuários locais (RED HAT, 2013), dos *hypervisors* testados apenas o qemu/KVM suporta este protocolo. A **Tabela 28** algumas características técnicas dos *hypervisors*.

Tabela 28 – Características Técnicas nos *Hypervisors* suportados pelo OpenStack

Características Técnicas								
Características	**Xen**	**qemu/KVM**	**VMWare ESXi**	**LXC via llibvirt**	**Hyper-V**	**Baremetal**	**PowerVM**	**Docker**
Migração *live* de VM's	✓	✓	✗	N.T	N.T	N.T	N.T	N.T
Snapshots	✓	✓	✓	N.T	N.T	N.T	N.T	N.T
Integração com o Glance	✓	✓	✓	✓	✓	✓	✓	✓
Console VNC	✓	✓	✓	✗	N.T	✗	N.T	✗
Permitem Diagnósticos do Nova	✓	✓	N.T	N.T	N.T	N.T	✗	✗

Fonte: (OPENSTACK HYPERVISORS, 2013)

Algumas características como migração *live* de VM's são interessantes caso haja necessidade de migração da VM de um *host* para outro sem haja nenhuma interrupção no serviço prestado ao cliente, somente o Xen e o qemu/KVM suportam esta característica, enquanto que o VMWare não suporte e os demais não foram testados. O snapshot é a característica que se refere a capacidade do *hypervisor* de capturar o estado de uma VM em um determinado ponto (CSA, 2011), os três principais *hypervisors* qemu/KVM, Xen e VMWare

suportam esta operação enquanto os demais não foram testados. O console VNC permite a conexão com a VM por meio de uma interface gráfica permitindo a administrador a executar diversas operações na VM. Algumas características como a integração com o Glance e a capacidade do Nova de realizar diagnósticos sobre os *hypervisors* são importantes para o OpenStack. O Glance possui amplo suporte a vários formatos de imagens, suportanto assim todas imagens utilizadas pelos *hypervisors*, para o diagnóstico do Nova somente o Xen e o qemu/KVM possuem compatibilidade com o OpenStack. A **Tabela 29** exibe algumas propriedades de segurança suportadas pela plataforma.

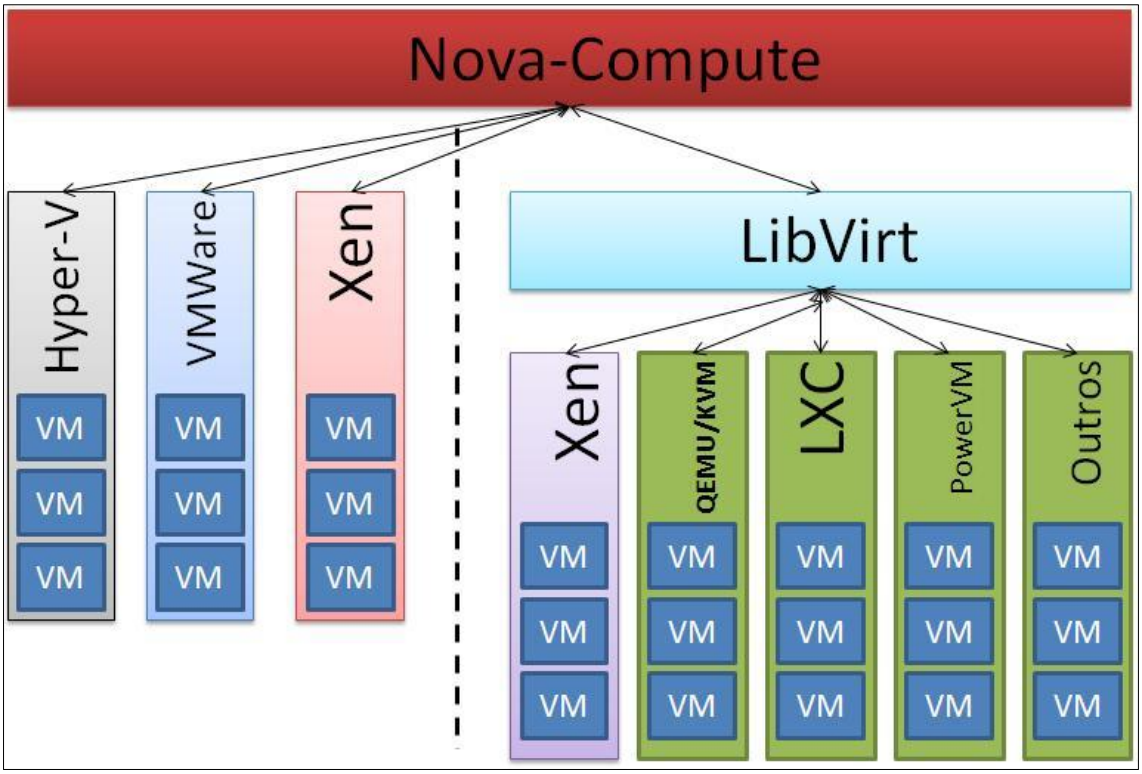
T**abela 29** – Características de Segurança nos *Hypervisors* suportados pelo OpenStack

Segurança								
Características	**Xen**	**qemu/KVM**	**VMWare ESXi**	**LXC via llibvirt**	**Hyper-V**	**Baremetal**	**PowerVM**	**Docker**
Modificar senha do Administrador	✓	✗	✗	✗	✗	✗	✗	✗
Grupos de Segurança	✓	✓	✗	N.T	N.T	N.T	✗	N.T
Regras de *Firewall*	✓	✓	N.T	N.T	N.T	N.T	✗	N.T

Fonte: (OPENSTACK HYPERVISORS, 2013)

Estas características foram definidas e testadas pela própria equipe do OpenStack para cada um dos *hypervisors*. É possível constatar na **Tabela 29** apenas algumas operações básicas de segurança, como a operação de modificação da senha do administrador refere-se à capacidade do *guest* de modificar a senha da conta administrativa em uma instância, somente o *hypervisor* Xen permite esta operação. De maneira geral, observa-se que ainda são necessários muitos testes em outros *hypervisors* e também suporte da plataforma para os *drivers* dos demais *hypervisors*. A **Figura 23** ilustra o suporte destes *hypervisors* no OpenStack.

Figura 23 – Suporte dos Hypervisors no OpenStack



Adaptado de: (RACKSPACE, 2013).

A **Figura *23*** ilustra que o Nova pode gerenciar os *hypervisors* via suporte nativo, neste caso o Nova necessita ter os *drivers* configurados destes *hypervisors* ou pode gerenciá-los via a API Libvirt, que é uma API específica

para suportar a comunicação com diversas ferramentas de virtualização (LIBVIRT, 2013). Por exemplo, o Hyper-V é gerenciado diretamente pelo Nova e o qemu/KVM é gerenciado via Libvirt, enquanto o Xen e o VMware podem ser gerenciados diretamente pelo Nova ou via alguma API como o Libvirt. Alguns dos *hypervisors* que o Libvirt suporta

- Qemu/KVM;
- XEN;
- LXC;
- OpenVZ ;
- VirtualBox ;
- VMWare ESX e GSX;
- Hyper-V;
- IBM PowerVM;
- Parallels;
- Redes virtuais utilizando bridge, NAT, VEPA e VN-LINK; e
- Armazenamento em discos IDE/SCSI/USB entre outros

A ferramenta unifica alguns comandos para executar operações básicas em VM's de diferentes *hypervisors*, como iniciar, pausar, salvar, restaurar, migrar, entre outros comandos. Algumas características de segurança fornecidas pelo Libvirt são:

- Gerenciamento remoto utilizando cifras TLS (Transport Layer Secure) e certificados x509;
- Gerenciamento remoto de autenticação utilizando Kerberos ;e
- Controle de acesso local utilizando um conjunto de políticas.

As características de segurança fornecidas com a utilização da API, como métodos de autenticação e comunicação cifradas funcionam sob um conjunto de políticas. A ferramenta unifica alguns comandos para executar operações básicas em VM's de diferentes *hypervisors*, como iniciar, pausar,

salvar, restaurar, migrar, entre outros comandos. Ainda possui como benefícios a possibilidade de utilizar canais de comunicação cifrados via TLS, certificados x509 para autenticação e possibilidade de autenticação remota utilizando o protocolo Kerberos. Para melhor compreensão das principais soluções de virtualização as próximas subseções apresentam uma breve descrição do qemu/KVM (**3.3.1**), Xen (**3.3.2**) e VMWare (**3.3.3**).

3.3 PRINCIPAIS *HYPERVISORS* SUPORTADOS PELO OPENSTACK GRIZZLY (qemu/KVM, Xen, VMWare)

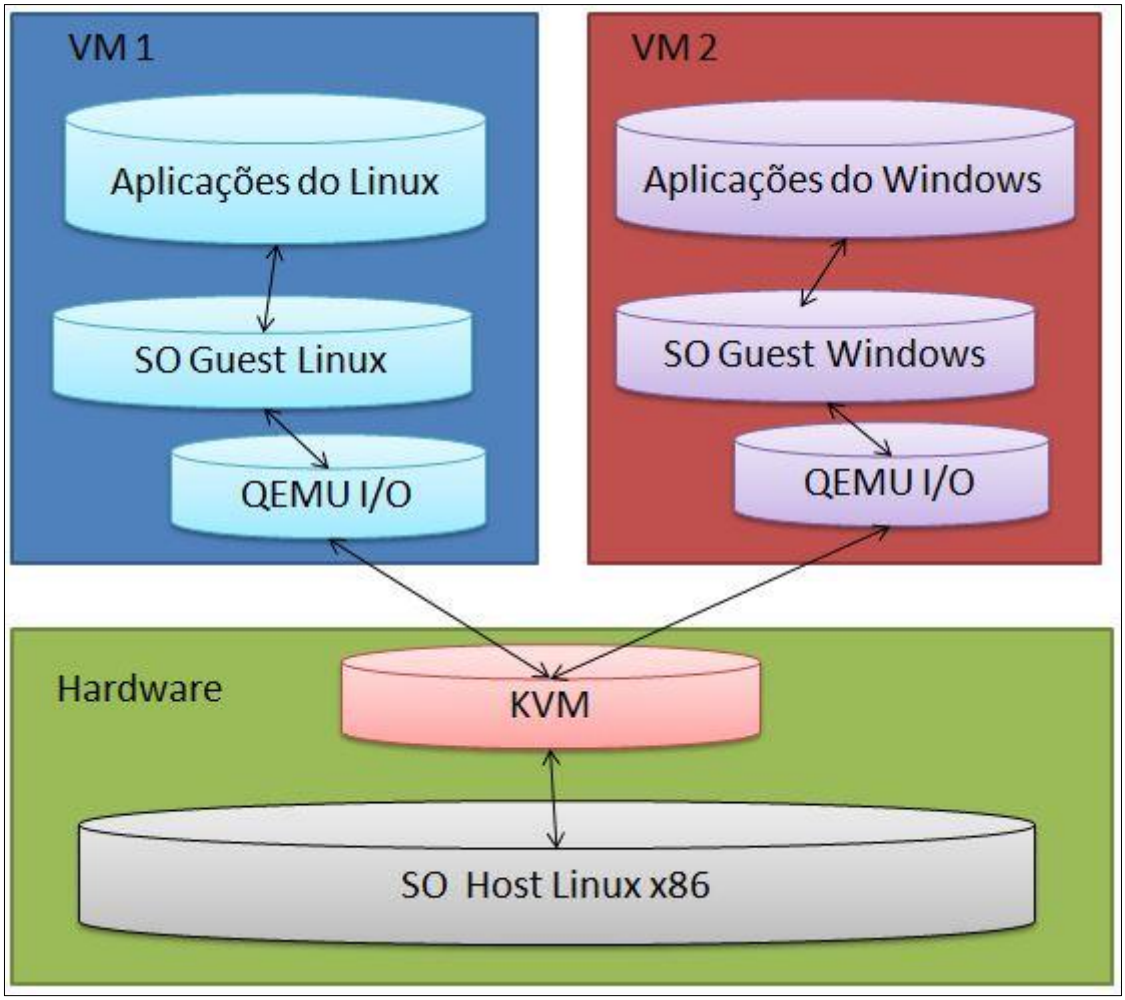
Nesta subseção é realizada uma descrição breve dos principais *hypervisors* suportados pelo OpenStack evidenciando as características relacionadas a segurança de cada um. A subseção **3.3.1** apresenta o qemu/KVM, a **3.3.2** o Xen, **3.3.3** o VMWare e a subseção **3.3.4** uma visão geral dos *bugs* estabelecidos no *LaunchPad* para estes *hypervisors*.

3.3.1 QEMU/KVM EM X86

Criado em 2007 pela empresa Red Hat, o qemu/KVM trata-se de uma solução de virtualização, do tipo 2 (**Apêndice A)** para GNU/Linux baseada em arquiteturas x86 (para Intel VT ou AMD-V). Os desenvolvedores do KVM criaram um método que transformou o próprio *kernel* do GNU/Linux em um *hypervisor*, simplificando a gestão e melhorando o desempenho em ambientes virtualizados (HABIB, 2008). A plataforma suporta processadores de 32 e 64 bits usando Intel VT ou AMD-V, que são tecnologias de virtualização nos quais

SO's para plataformas x86 são executadas sob outro SO x86 hospedeiro. A **Figura 24** ilustra o cenário do qemu/KVM.

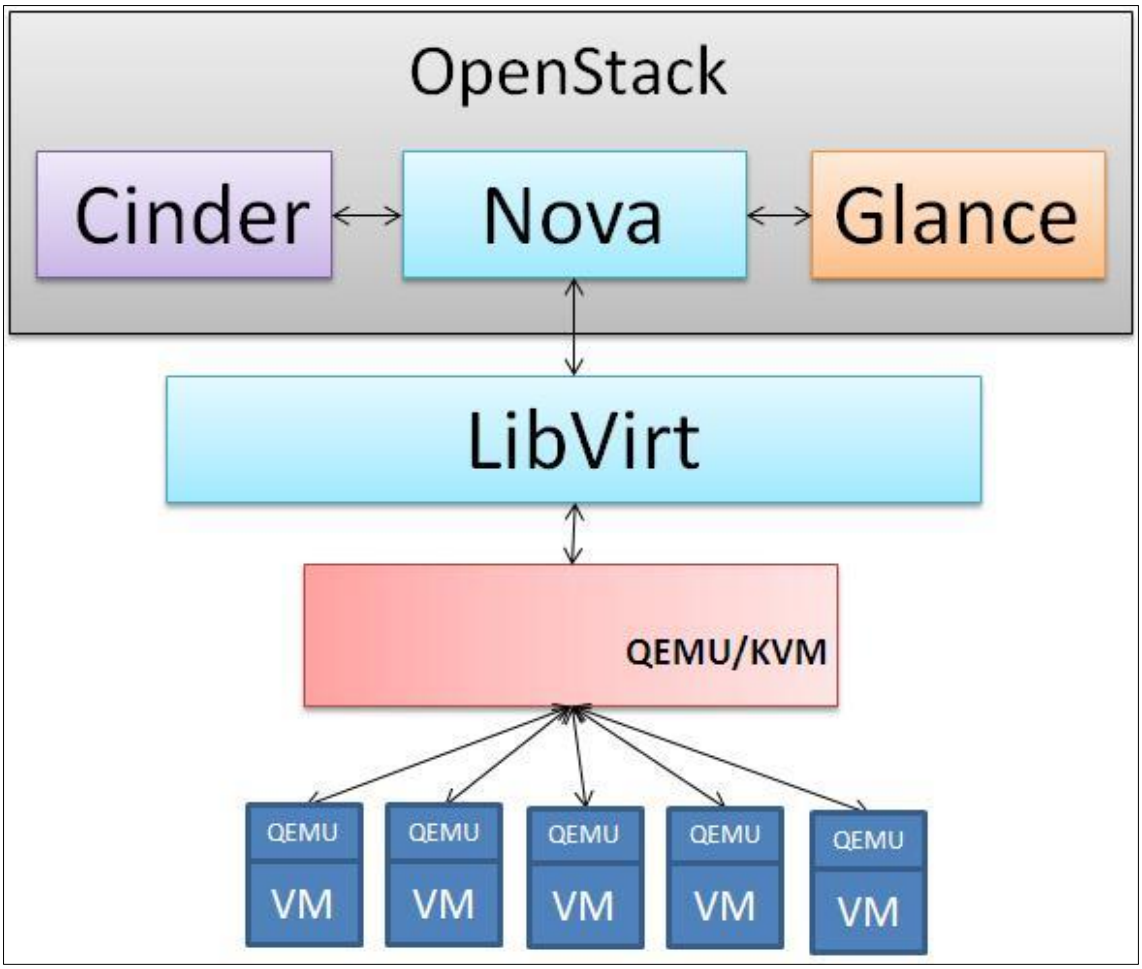
Figura 24 – Arquitetura do qemu/KVM



Adaptado de: (HABBIB, 2008).

A **Figura 24** ilustra a arquitetura do KVM, seu funcionamento consiste na utilização de uma interface localizada nas VM's, para criar um espaço de endereçamento único para a VM, simulando a entrada e saída de dados e mapeando saída de vídeo para o hospedeiro. A **Figura 25** ilustra como o qemu/KVM é utilizado no OpenStack por meio do Libvirt.

Figura 25 - Arquitetura do Qemu/KVM no OpenStack



Adaptado de: (RACKSPACE, 2013).

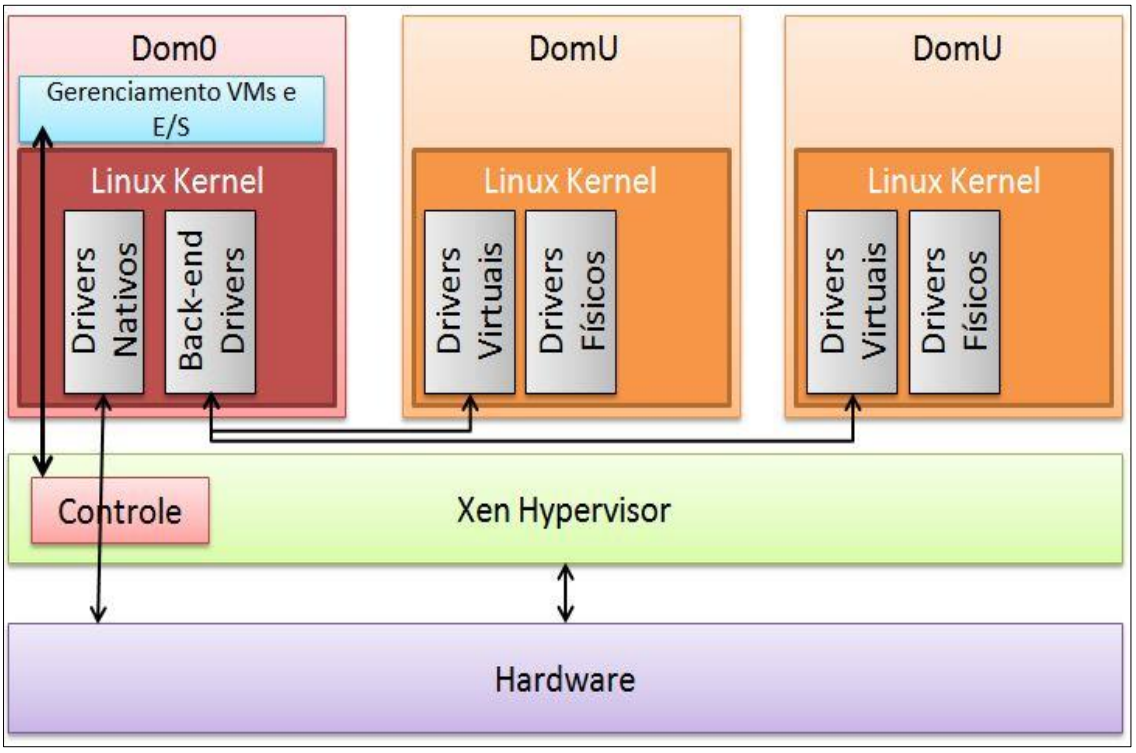
A diferença entre a **Figura 24** e **Figura 25** é que a camada de *software* do OpenStack simula o *hardware* sobre o qual o qemu/KVM executa, comunicando-se com o *hypervisor* por meio da API Libvirt. De forma geral, os processos Linux apresentam dois modos de execução, uma aplicação executa em modo usuário até o momento em que se tornam necessários serviços mais privilegiados, como escrita de dados no disco rígido, enviando assim um pedido para o modo *kernel*, que executa a operação. O KVM trata cada VM como sendo um processo regular do gerenciador de processos do GNU/Linux, incluindo um terceiro modo, o modo *guest*, o qual executa a partir VM (LINUX INSIGHT, 2008).

3.3.2 XEN

O Xen é originário de um projeto de pesquisa da universidade de Cambridge[42], na Inglaterra em 2003 (BARHAM et al., 2003). É um *hypervisor* do tipo 1 (**Apêndice A**), que utiliza o recurso da paravirtualização. Através da paravirtualização, o Xen pode alcançar um alto desempenho mesmo em arquiteturas x86, que têm a reputação de não-cooperação com técnicas de virtualização tradicionais (OPENSTACK, 2013).

Para entender como o Xen implementa a para-virtualização, é importante salientar o conceito de domínio. Os domínios são as máquinas virtuais do Xen. Essas podem ser de dois tipos, privilegiadas (domínio 0 – dom0) e não-privilegiadas (domínio U - domU) (MENEZES, 2008). A **Figura 26** ilustra a arquitetura do Xen.

Figura 26 - Arquitetura do Xen

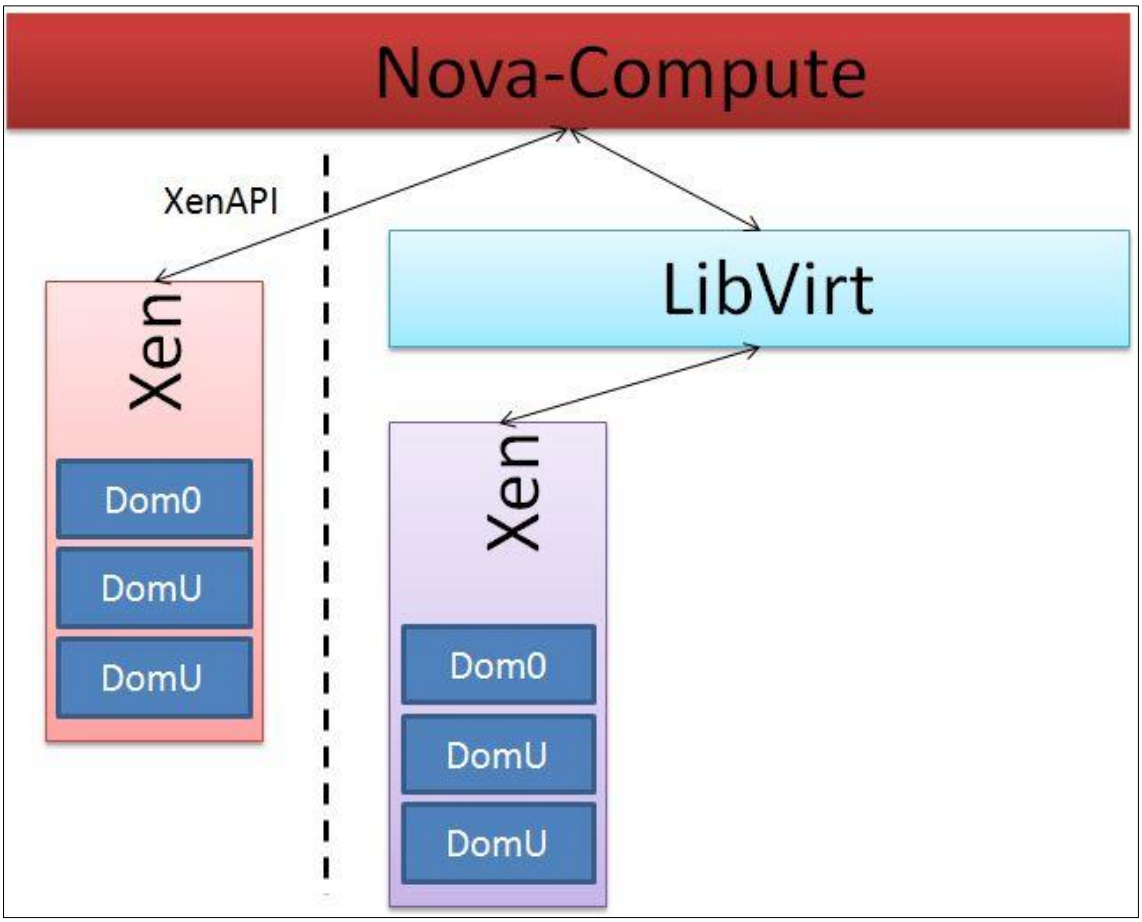


Adaptado de: (MENEZES, 2008).

[42] http://www.cam.ac.uk/

Quando a máquina *host* é iniciada, uma VM do domínio 0, privilegiado, é criada. Esse domínio acessa uma interface de controle no Xen e executa aplicações de gerenciamento e também de entrada e saída (E/S) para as VM's. As VM's dos domínios U só podem ser criadas, iniciadas e desligadas através do domínio 0 (BARHAM et. al., 2003). Na VM do domínio 0, é executado um Linux com *kernel* modificado, que pode acessar os recursos da máquina física, já que possui privilégios especiais, e ainda se comunicar com as outras máquinas virtuais, domínio U. O SO do domínio 0 tem que ser modificado para possuir os *drivers* de dispositivo da máquina física e *drivers* de *back-end* que tratam requisições de acessos à rede e ao disco realizadas pelas máquinas virtuais do domínio U. Em resumo, só a VM do domínio 0 tem acesso direto aos recursos da máquina física, enquanto que as demais VM's tem acesso a uma abstração dos recursos, que para serem acessados, devem passar pela VM do domínio 0. A **Figura 27** representa como o Xen pode ser utilizado no OpenStack

Figura 27 – Arquitetura do Xen no OpenStack



Adaptado de: (RACKSPACE, 2013).

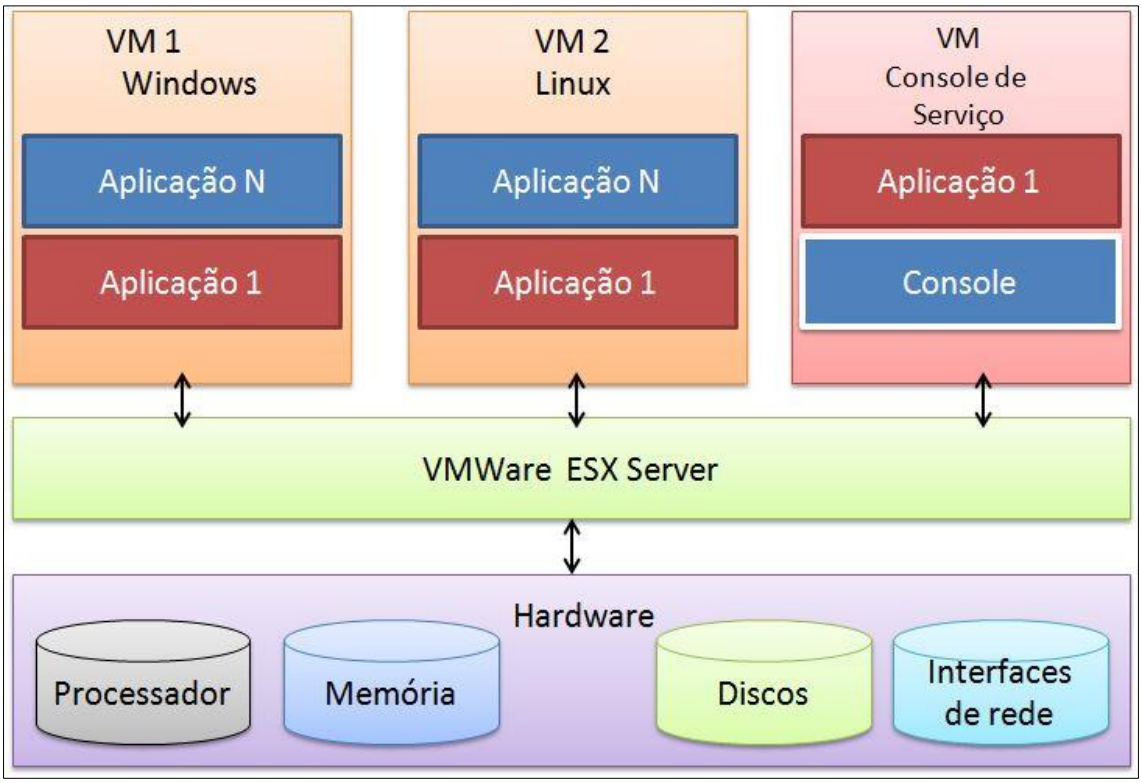
O Xen possui duas formas em que pode ser utilizado no OpenStack, uma é diratemente por meio da interface XenAPI ou pelo Libvirt. Segundo (BERRANGE, 2008) existem algumas vantagens de se usar o Libvirt ao invés da XenAPi, como evita que sua aplicação fique presa em um *hypervisor* em particular, permitindo portar sua aplicação para qualquer *hypervisor* que é suportado pelo Libvirt, acesso remoto assegurado por SSL, certificados X509 entre outros. Porém, o (OPENSTACK, 2013) recomenda que seja utilizado os *drivers* do XenAPI para a utilização do *hypervisor* Xen, pois o XenAPI é uma biblioteca da linguagem Python, que oferece suporte para o *hypervisor*.

3.3.3 VMWARE

O VMWare é uma das mais populares plataformas de virtualização para a arquitetura x86. Nesta versão do OpenStack Grizzly, a VMWare buscou ampliar seu comprometimento com a plataforma, e cedeu por meio da Citrix parte do códigos para a integração dos *drivers* do *Hypervisor* ESX para o Nova (RED HAT, 2013). Ainda segundo a Red Hat estes *drivers* cedidos ainda fornecem uma integração limitada com o Nova, e ainda não possuem grande suporte comparado aos demais *hypervisors*, mas agora o VMWare faz parte da comunidade OpenStack e tem trabalhado para melhorar esta integração.

O VMWare ESX Server é um *hypervisor* tipo 1 que cria conjuntos lógicos de recursos do sistema, de modo que muitas máquinas virtuais possam gerenciar os mesmos recursos físicos. É um SO que funciona como um *hypervisor* e é executado diretamente no hardware do sistema. O ESX Server insere uma camada de virtualização entre o *hardware* do sistema e as VM's, transformando o *hardware* do sistema em um conjunto lógico de recursos de computação que o ESX Server pode alocar dinamicamente a qualquer sistema operacional ou aplicativo (THOLETI, B., 2012). Os SO's *guests* executando em VM's interagem com os recursos virtuais como se fossem recursos físicos. A **Figura 28** exibe a arquitetura do VMWare ESX Server.

Figura 28 – Arquitetura VMWare ESX Server.



Adaptado de: (THOLETI, B., 2012).

A **Figura 28** a mostra um sistema com um ESX Server executando VM's. O ESX Server está executando uma VM com o console de serviço e três máquinas virtuais adicionais. Cada máquina virtual adicional está executando um sistema operacional e aplicativos independentes das outras máquinas virtuais enquanto compartilham os mesmos recursos físicos.

No OpenStack, o Nova possui integração direta com o VMWare ESX por meio dos *drivers* (RACKSPACE, 2013):

- **Vmwareapi.VMWareESXDriver**: originalmente escrito pela Citrix e depois atualizado pela VMWare, permite ao Nova se comunicar diretamente a um *host* ESXi;
- **Vmwareapi**.**VMWareVCDriver:** criado pela VMWare para o *release* Grizzly, permite ao Nova se comunicar com o servidor gerenciador do VMWare gerenciando um *cluster* de *hosts* ESXi's.

A utilização do *hypervisor* pode ser realizada diretamente do Nova com os *drivers* do ESX ou utilizando-se a API do Libvirt, como ilustrado na **Figura 23** e **Figura 27.**

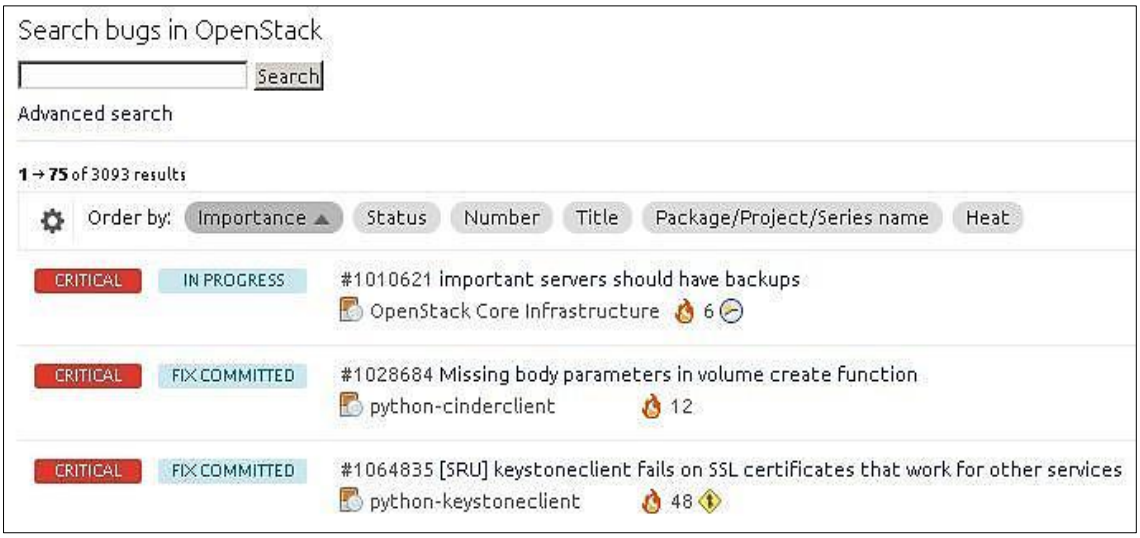
3.3.4 VISÃO GERAL DOS BUGS E VULNERABILIDADES PARA O QEMU/KVM, XEN E VMWARE

Esta subseção tem por objetivo fornecer uma visão geral dos *bugs* relacionados às plataformas de virtualização no OpenStack, quantos foram resolvidos, quantos permanecem em aberto, permitindo com isto avaliar o funcionamento destes *hypervisors* na plataforma. Primeiramente é apresentado o LaunchPad, que é a estrutura utilizada pela plataforma para que a comunidade possa submeter os *bugs*, depois os *bugs* e vulnerabilidades encontrados para as plataformas de virtualização.

3.3.4.1 CRITÉRIOS LAUNCHPAD E CVE (*Common Vulnerabilities and Exposures*)

O LaunchPad permite que a comunidade possa submeter bugs encontrados no OpenStack (em qualquer release ou componente) através do envio de e-mails seguindo um padrão para descrever cada bug. Os bugs podem ser associados a um componente e também possuir uma *tag* identificando o seu tipo, facilitando a localização dos mesmos de acordo com o componente ou seu tipo (e.g., para segurança pode ser realizada uma localização pelo componente Keystone ou a *tag security*).

Figura 29 - Interface do *LaunchPad*.



Fonte: (LAUNCHPAD, 2013).

Os *bugs* listados usando o sistema *Launchpad*, ilustrado pela **Figura 29,** são categorizados de acordo com seis características:

- ***Importance* (Relevância ou Importância)**: lista de espera, indefinida, baixa, média, alta e crítica;
- ***Status***: nova, confirmada, em triagem, em progresso e corrigida;
- ***Number***: número de identificação do *bug*;
- ***Title***: título dado ao *bug*;
- ***Release****:* qual versão o *bug* pertence (Essex, Folsom, Grizzly, versões mais antigas podem não ser suportadas); e
- ***Heat* (Popularidade)**: classificação por quantidade de comentários no *bug*.

Os *bugs* levantados no *Launchpad* serão observados em dois gráficos, um por importância do *bug* e outro pelo status. O OpenStack classifica a importância dos *bugs* utilizando a classificação de quatro níveis para os *bugs* (crítica, alta, média e baixa) mais dois níveis que permitem a definir com precisão a situação do *bug* para a plataforma (indecidido e lista de espera). Os níveis para classificar os *bugs* são (SOFTWARE FUNDAMENTALS, 2013):

- **Crítica**: são *bugs* que afetam funcionalidades ou dados críticos da plataforma, geralmente não possuem soluções alternativas e necessitam ser solucionados rapidamente;
- **Alta:** são *bugs* que prejudicam as funcionalidades críticas do sistema, porém geralmente possuem soluções alternativas. *Bugs* nesta categoria possuem solução, mas estas geralmente são complexas;
- **Média:** estes *bugs* afetam funcionalidades ou dados que não são críticos para o funcionamento do sistema. Geralmente estes *bugs* possuem soluções triviais;
- **Baixa**: *bugs* deste tipo não afetam dados ou funcionalidades importantes para o sistema, não afetando a eficiência ou a produtividade da plataforma, normalmente são considerados inconveniências como erros de gramática, erros de leiaute, entre outros;

Esta classificação dos *bugs* por quatro relevâncias é utilizada por diversas entidades relacionadas a testes de *software* (SOFTWARE FUNDAMENTALS, 2013) (UTEST, 2013) para a classificação dos *bugs*. O OpenStack, para classificar com mais precisão a situação do *bug* adicionou dois tipos (LAUNCHPAD, 2013):

- **Lista de espera**: são *bugs* que podem ser corrigidos na próxima atualização para um determinado componente ou no próximo *release*;
- **Indefinido**: são *bugs* que ainda não foram analisados pela equipe responsável pelo componente em questão.

Além da classificação da relevância dos *bugs*, o OpenStack define o status de cada *bug*, para controlar o tratamento dispensado a cada *bug*. Esta categorização é importante para determinar a situação dos *bugs* por *hypervisors* e também de maneira geral, verificando quantos estão resolvidos,

quantos estão confirmados, entre outros. O status dos *bugs* são determinados de acordo com as seguintes categorias:

- **Novos:** o *bug* foi recentemente criado;
- **Incompletos:** o *bug* está esperando dados de quem o reportou;
- **Confirmados:** o *bug* foi reproduzido e confirmado;
- **Em triagem:** os comentários do *bug* contém uma análise completa de como corrigir adequadamente o problema.
- **Em progresso:** o trabalho para corrigir o *bug* está em progresso
- **Correção enviada:** a correção foi enviada para ser unida ao componente ou *release* atual da plataforma;
- **Correção liberada:** correção foi liberada para o componente proposto, mas para um componente ou *release* antigo da plataforma;
- **Opinião:** o *bug* trata-se de um problema válido, mas não deve ser corrigido;
- **Não será corrigido:** o *bug* é válido, mas não há a intenção de se corrigir o problema.

Para listar e categorizar as vulnerabilidades destes *hypervisors* será utilizada a base de dados do CVE[43] (*Common Vulnerabilities and Exposures*). O CVE é uma lista pública e livre de identificadores padronizados para vulnerabilidades e exposições de sistemas computacionais (CVE MITRE, 2013). A base de dados onde estão os dados é fornecida pelo NIST, chamada de NVD (*National Vulnerability Database*). O CVE disponibiliza algumas métricas para avaliar as vulnerabilidades publicadas, que são:

- **Severidade:** baixa (0-3), média (4-6), alta e média (4-10), alta (7-10);
- **Vetor de acesso:** rede, rede local, apenas local;
- **Complexidade de acesso:** baixa, média e alta;
- **Autenticação:** nenhuma, única, múltipla.

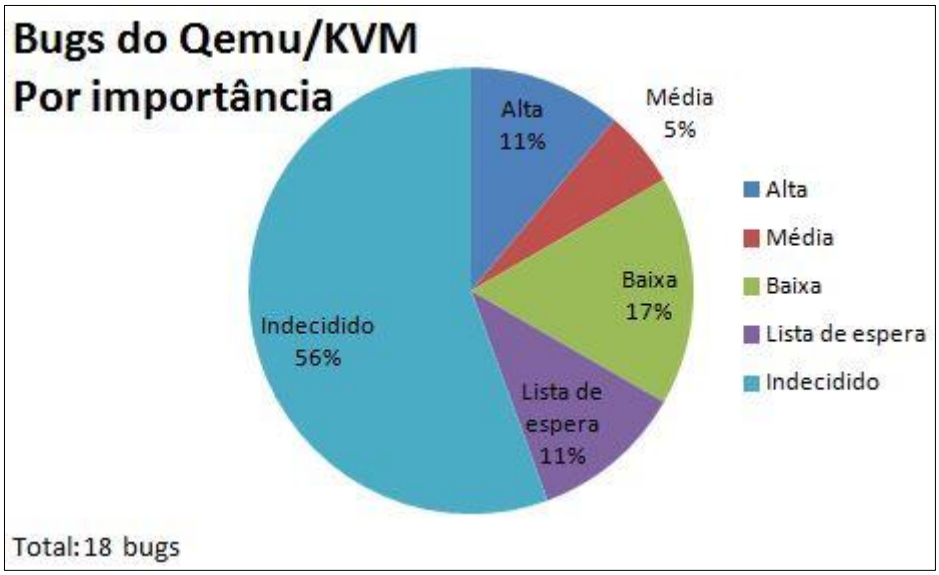
[43] http://www.cve.mitre.org/cve/

As métricas disponibilizadas pelo CVE de vetor de acesso das vulnerabilidades, complexidade de acesso, e necessidade de autenticação são fatores que influenciam na categorização da severidade final de cada vulnerabilidade. Com base nestas métricas é possível avaliar a situação geral da segurança das plataformas de virtualização. As próximas subseções apresentam os gráficos realizados com os dados obtidos para os *hypervisors* Qemu/KVM (**3.3.4.2**), Xen (**3.3.4.3**) e VMWare (**3.3.4.4**) e na última subseção são apresentadas as considerações (**3.3.4.5**).

3.3.4.2 QEMU/KVM

O qemu e o KVM são os principais *hypervisors* suportados pelo OpenStack, consequentemente também o mais explorado pela comunidade que utiliza e reporta *bugs* no LaunchPad. A **Figura 30** apresenta os *bugs* agrupados por importância.

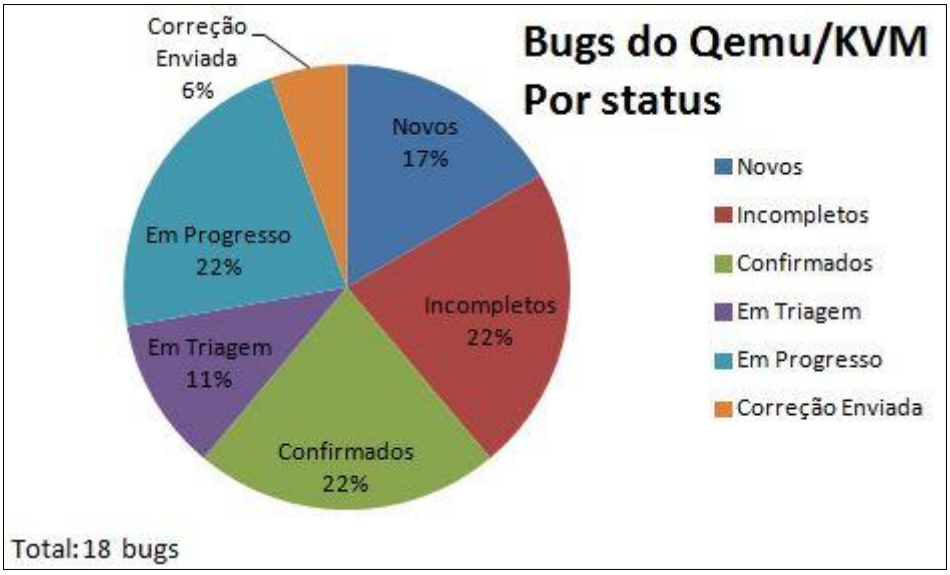
Figura 30 – Bugs do Qemu/KVM por Importância



Fonte: (LAUNCHPAD, 2013).

É possível observar que dos dezoito *bugs* apresentados, 56% ainda estão classificados como indefinidos, indicando que estes são *bugs* novos e que ainda não foram analisados. Com 17% dos *bugs* são de baixa importância, o que é mais do que a soma dos *bugs* classificados com importância média e alta, isto pode indicar que pelo fato de ser o *hypervisor* mais utilizado pela comunidade, as principais falhas já foram resolvidas, restando apenas falhas que não representam grande ameaça a estabilidade do *hypervisor.* A **Figura 31** apresenta os dezoito *bugs* agrupados por status.

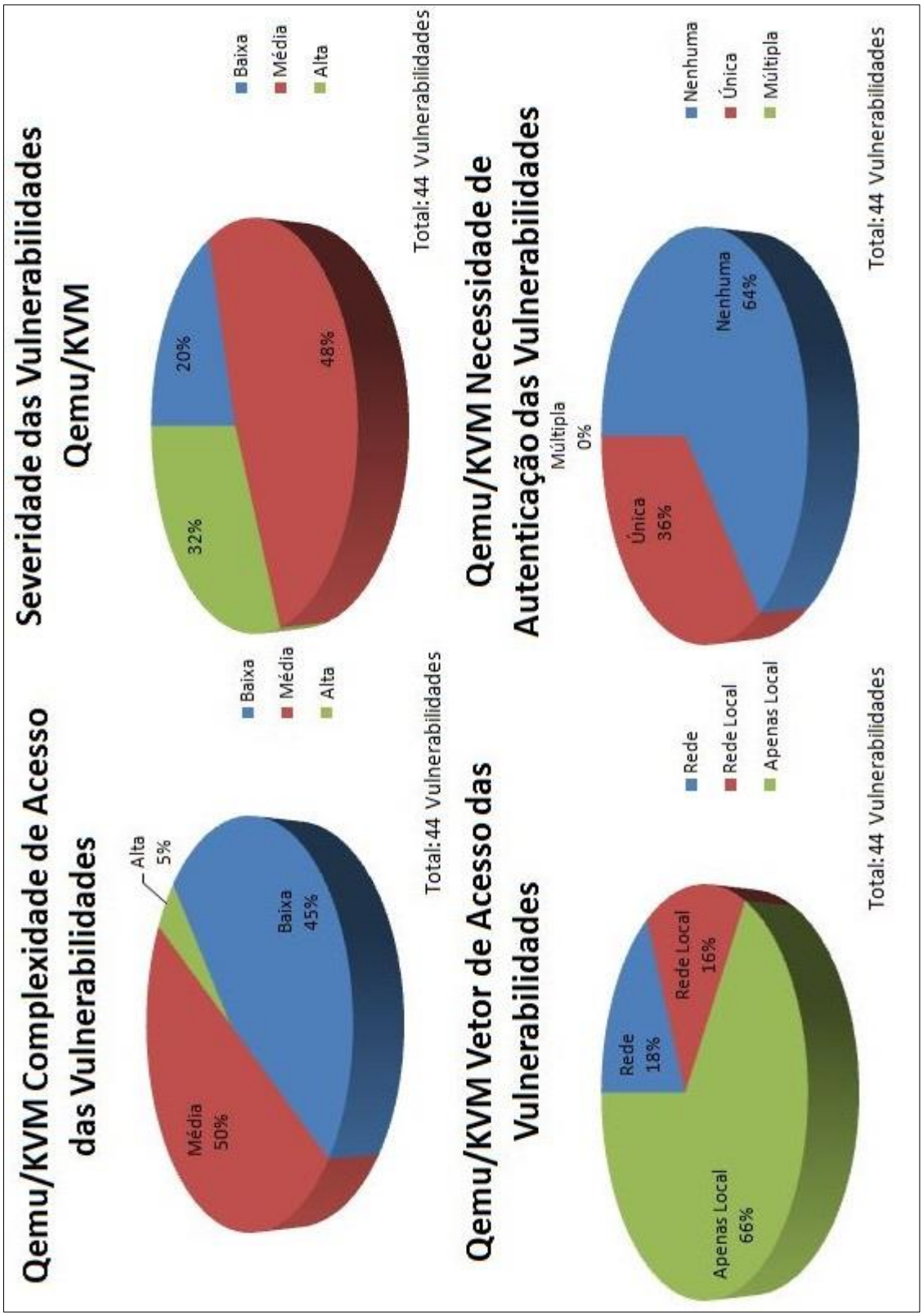
Figura 31 – Bugs do Qemu/KVM por Status



Fonte: (LAUNCHPAD, 2013).

O gráfico da **Figura 31** confirma o que foi apresentado no gráfico anterior, como grande parte dos *bugs* foram reportados recentemente, observa-se que existem 17% novos *bugs* e 22% incompletos esperando mais dados por quem reportou o *bug.* Ainda é possível observar que pelo fato da maioria das falhas serem novas, apenas 6% foram corrigidos e 22% ainda estão em processo de correção, indicando que a grande maioria está ou entrará em processo de análise pelos responsáveis pelo suporte dos *hypervisors* no OpenStack. As vulnerabilidades estão listadas nos gráficos presentes na **Figura 32**.

Figura 32 – Vulnerabilidades Qemu/KVM



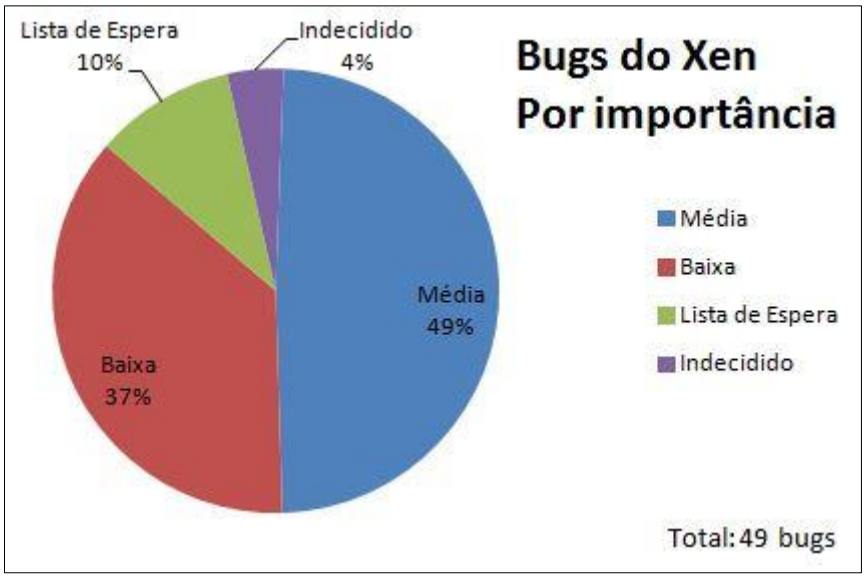
Fonte: (CVE MITRE, 2013).

A **Figura 32** apresenta quatro gráficos obtidos a partir dos dados encontrados no CVE, o gráfico de severidade segue um conceito parecido com o gráfico apresentado na **Figura 30**, categorizando as vulnerabilidades em alta, média e baixa. O resultado mostrou que 48% das vulnerabilidades são classificadas como média não expondo funcionalidades críticas do sistema, seguida de 32% de vulnerabilidades categorizadas como alta, que representam um grande risco ao *hypervisor*. O gráfico de vetor de acesso indica o meio em que as vulnerabilidades podem ser reproduzidas, o ideal é que as vulnerabilidades possam ser reproduzidas localmente, não expondo o *hypervisor* pela rede. O gráfico apresentou 66% das vulnerabilidades com vetor de acesso apenas local. A complexidade de acesso indica a dificuldade de se reproduzir esta vulnerabilidade, ou seja, se ela exige grandes conhecimentos técnicos ou não. Neste gráfico, quanto mais baixa a classificação pior, pois indica que as vulnerabilidades são mais fáceis de serem reproduzidas, para o qemu/KVM 45% das vulnerabilidades foram classificadas como baixa e 50% como média, o que pode ser interpretado como um resultado ruim. O gráfico de necessidade de autenticação tem como objetivo dizer se para reproduzir estas vulnerabilidades é necessário, ou não estar autenticado. O gráfico apresentou em 64% das vulnerabilidades que não é preciso estar autenticado para realiza-las, o que aumenta de modo geral a severidade das vulnerabilidades.

3.3.4.3 XEN

O Xen é um *hypervisor* que está definido no segundo grupo de *hypervisors* que o OpenStack oferece suporte, isto quer dizer que a plataforma oferece suporte aos *drivers* porém as funcionalidades não foram completamente testadas. A **Figura 33** exibe os *bugs* agrupados por importância.

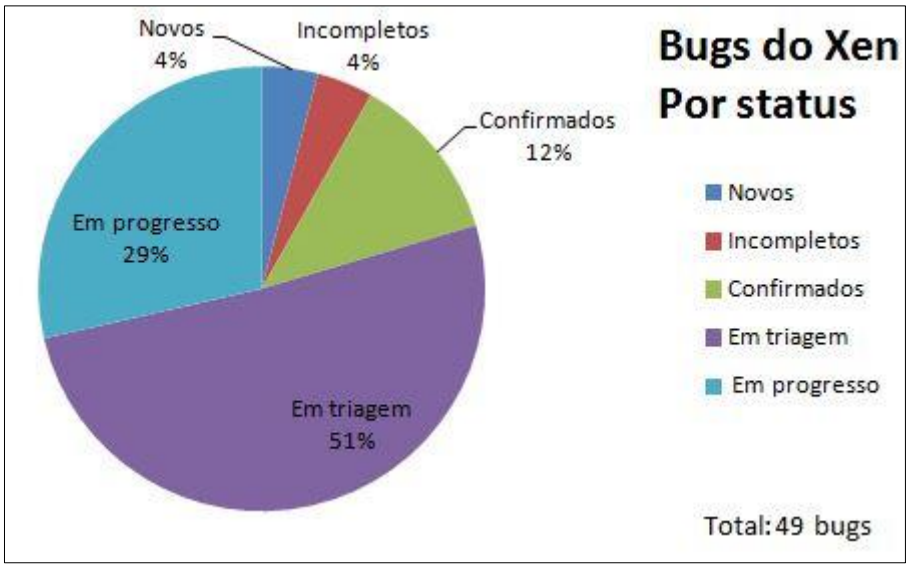
Figura 33 – Bugs do Xen por Importância



Fonte: (LAUNCHPAD, 2013).

O Xen apresenta um número relativamente alto de *bugs* se comparado ao qemu/KVM, tal fato ocorre por que a comunidade por meio do uso da plataforma acaba descobrindo e reportando os erros encontrados. É possível notar que do total de *bugs* nenhum destes é classificado como erros críticos ou altos, mas quase metade dos *bugs* classificados como médios, significando que as funcionalidades críticas do *hypervisor* estão seguras e os *bugs* encontrados não afetam estas funcionalidades e possuem soluções triviais. A **Figura 34** apresenta os *bugs* do Xen agrupados por status.

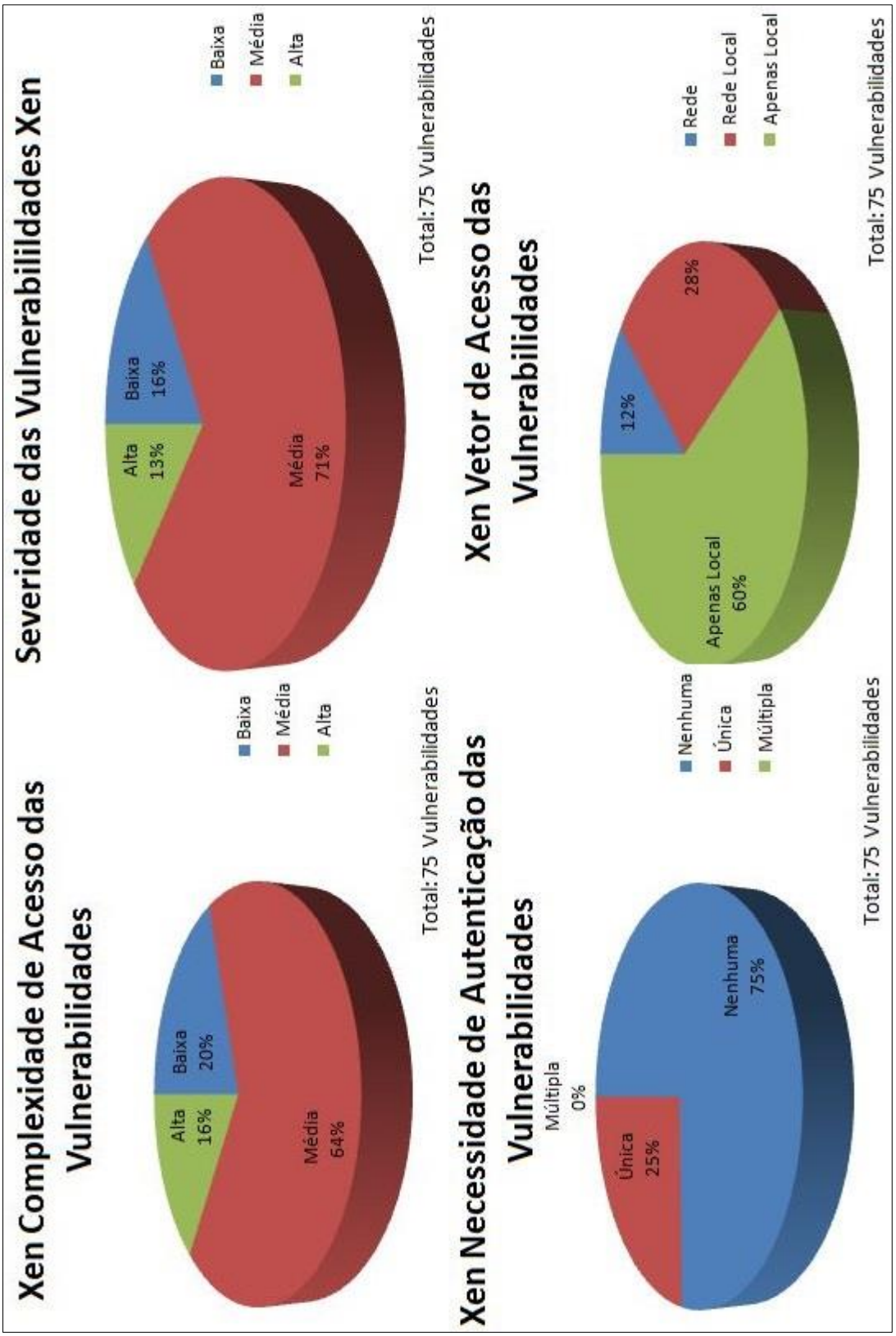
Figura 34 – Bugs do Xen por Status



Fonte: (LAUNCHPAD, 2013).

A **Figura 34** apresenta os *bugs* do Xen classificados por status. A metade 51% dos 49 *bugs* reportados ainda está em processo de triagem pelos analistas, e quase 30% já estão sendo corrigidos. É possível inferir que a maioria dos *bugs* listados são recentes, mas não totalmente novos ou recém submetidos, eles estão em processo de análise pelos responsáveis pelo Xen no OpenStack. As vulnerabilidades estão listadas nos gráficos presentes na **Figura 35**.

Figura 35 – Vulnerabilidades Xen



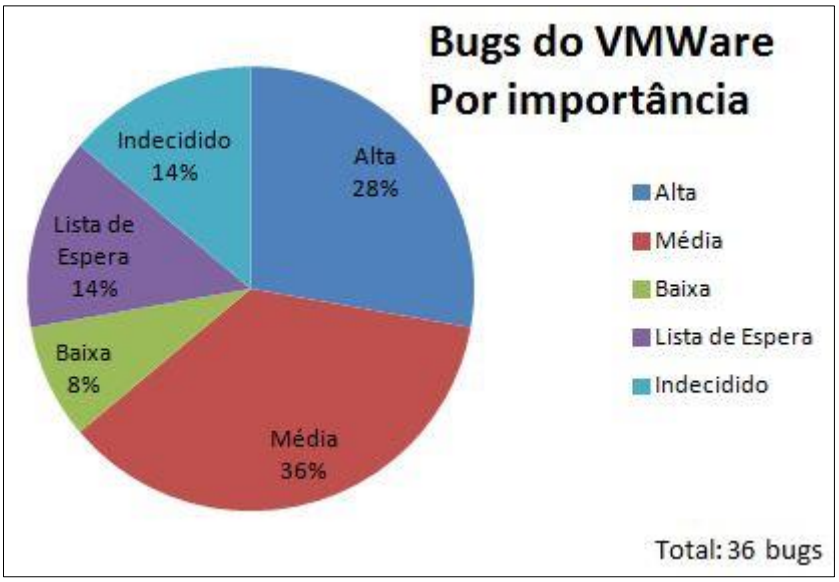
Fonte: (CVE MITRE, 2013).

O gráfico de severidades apresenta a grande maioria das vulnerabilidades com severidade média, que não apresentam grandes riscos para os dados do *hypervisor*, nem impossibilitam o seu uso, mas que também não podem ser desconsideradas. Destas vulnerabilidades, 60% podem ser reproduzidas localmente, muitas vezes uma vulnerabilidade pode ser classificada com uma alta severidade, mas o seu vetor de acesso pode ter influência na nota final da severidade, e o vetor de acesso local faz com que esta severidade seja diminuída. A complexidade de acesso destas vulnerabilidades também em sua maioria (64%) foi categorizada como média e 74% delas não necessitam de autenticação para serem reproduzidas.

3.3.4.4 VMWARE

O VMWare ESX Server está no grupo C definido pelo OpenStack, ou seja o suporte as funcionalidades do *hypervisor* ainda não é suportado completamente pela plataforma, pois os *drivers* ainda não foram completamente cedidos pela VMWare e testados pela comunidade de usuários da plataforma. A **Figura 36** apresenta os *bugs* do VMWare agrupados por importância.

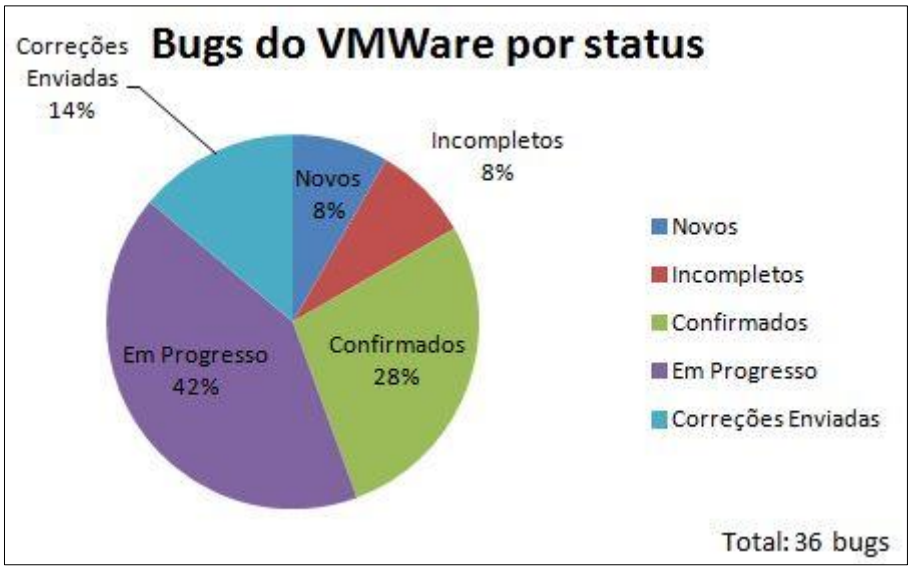
Figura 36 – Bugs do VMWare por Importância



Fonte: (LAUNCHPAD, 2013).

O VMWare apresenta um total de trinta e seis *bugs* na plataforma, não é um número tão alto se comparado ao Xen mas é compreensível pois o VMWare não é um *hypervisor* tão utilizado como o Xen, portanto a descoberta de *bugs* pela comunidade é tende a ser um pouco maior para o Xen. O VMWare não apresenta *bugs* críticos para a plataforma, entretanto apresenta 36% dos bugs de média e 28% de alta importância, denotando que a maior parte dos *bugs* listados já foram analisados e classificados. A **Figura 37** apresenta os *bugs* do VMWare classificados por status.

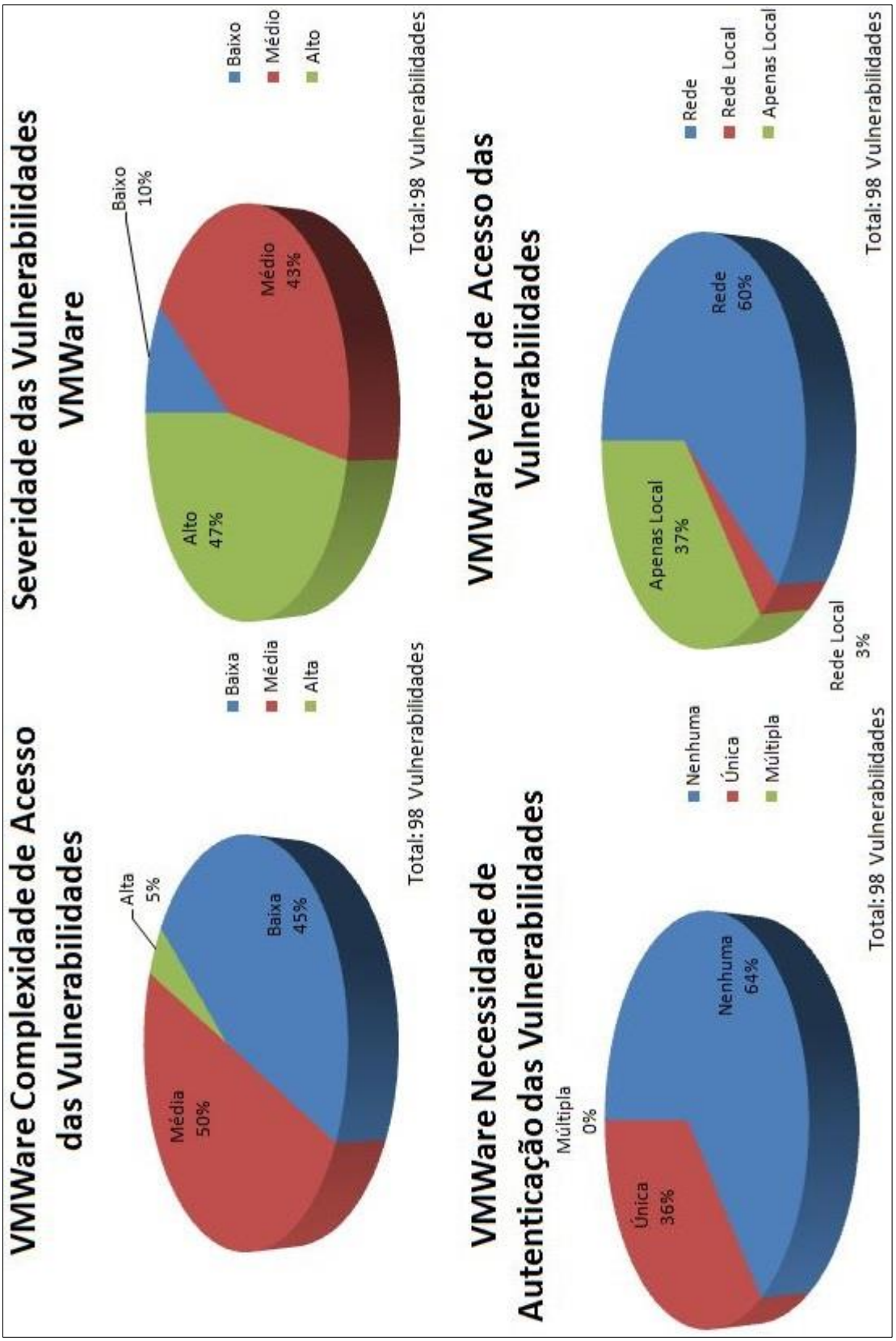
Figura 37 – Bugs do VMWare por Status



Fonte: (LAUNCHPAD, 2013).

Como observado na **Figura 37** os *bugs* listados não são recentes, ou seja, já passaram por um processo de análise e em sua maioria estão em processo de elaboração da correção (42%), os *bugs* confirmados representam quase 38% e a porcentagem de correções enviadas também e alta 14% se comparada aos demais (Xen e Qemu/KVM). Apenas 8% dos *bugs* listados são novos e 8% possuem descrição incompleta por parte do usuário que reportou o problema. As vulnerabilidades estão listadas nos gráficos presentes na **Figura 38**.

Figura 38 – Vulnerabilidades VMWare



Fonte: (CVE MITRE, 2013).

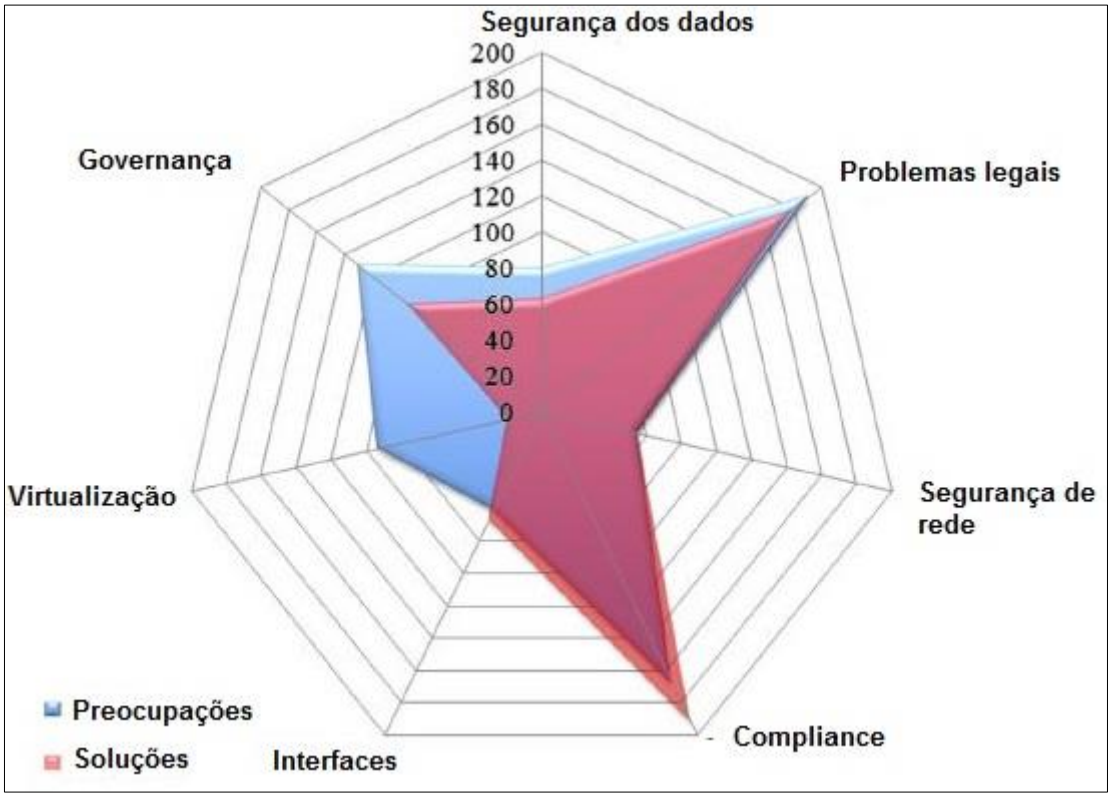
O VMWare é o *hypervisor* que apresentou vulnerabilidades com o maior nível de severidade, 47% das vulnerabilidades listadas no CVE eram classificadas como alta, isso se deve a grande parte destas vulnerabilidades estarem expostas para a rede, resultado que é apresentado no gráfico de vetor de acesso. A complexidade para a reprodução destas vulnerabilidades também é um fator que aumenta a severidade das vulnerabilidades, e para o VMWare 50% possuem complexidade média e 45% baixa. E para finalizar, a necessidade de autenticação é o fator final que contribui para a alta severidade das vulnerabilidades no VMWare, 64% das vulnerabilidades listadas não necessitam que o usuário esteja autenticado.

3.3.4.5 CONSIDERAÇÕES SOBRE OS BUGS E VULNERABILIDADES

Esta subseção teve como objetivo apresentar a situação dos principais *hypervisors* utilizados no OpenStack sob o prisma de seus *bugs* e vulnerabilidades de segurança. Para realizar esta categorização foi utilizada a base de dados e métricas do LaunchPad e CVE. O LaunchPad lista os *bugs* relacionados a cada *hypervisor* de modo geral, ou seja, estes não estão relacionados diretamente com falhas de segurança, então foi utilizada a base de dados e métricas do CVE que lista as vulnerabilidades pertinentes a cada *hypervisor*.

De modo geral, foi possível observar que os três *hypervisors* listados possuem um alto número de *bugs* e vulnerabilidades, o que confirma o que foi afirmado por (GONZALEZ et. al, 2012) na **Figura 38**, que existe um número bem maior de preocupações de segurança relacionadas a virtualização do que soluções para estas preocupações.

Figura 38 – Citações em Trabalhos de Preocupações x Soluções de Segurança na Computação em Nuvem



Fonte: (GONZALEZ et. al., 2012).

A **Figura 38**, adaptada do artigo de (GONZALEZ et. al, 2012) expõe o número de citações para preocupações e soluções em sete áreas relacionadas a computação em nuvem, dentre elas a virtualização. Como observado na **Figura 38** e também durante esta subseção, a virtualização é uma área que possui um grande número de preocupações e poucas soluções, o que pode justificar o alto número de *bugs* e vulnerabilidade apresentados nesta subseção.

Os *hypervisors* qemu/KVM foram os que apresentaram melhor resultado tanto na base de dados do LaunchPad como no CVE, por melhor resultado deve-se entender menos *bugs* e vulnerabilidades categorizados como alta importância. A maioria dos *bugs* listados no LaunchPad para o qemu/KVM eram novos, e destes, a minoria foi classificada como de alta importância, e no CVE a maioria das vulnerabilidades foi classificada com severidade média. Já o

pior resultado, tanto na análise do LaunchPad como no CVE, ficou a cargo do VMWare (ESX Server), apresentando mais *bugs* categorizados com severidade alta e média e no CVE a maioria das vulnerabilidades categorizadas como alta. O Xen também não apresentou bons resultados, mas ainda sim, resultados melhores do que o VMWare, apresentando grande maioria de vulnerabilidades com severidade média e vetor de acesso local, o que diminui o fator de risco destas vulnerabilidades.

É importante observar que nenhum dos três *hypervisors* analisados apresentou “bons” resultados no CVE, ou seja, em nenhum deles a maioria de suas vulnerabilidades foram classificadas com severidade baixa. Tal fato confirma a necessidade de um reforço de segurança nesta área que possui fator fundamental na computação em nuvem. Esta não é uma área relativamente nova, mas que está sendo bastante exigida com o franco crescimento da computação em nuvem, e como exposto por (GONZALEZ et. al. 2012), (GARTNER, 2011), e diversos outros autores, é uma área que necessita de mais estudos a fim de avançar as soluções para as preocupações existentes nesta área. A próxima subseção propõe um *framework* para categorizar estas vulnerabilidades de virtualização com base nos estudos realizados neste trabalho.

3.4 PROPOSTA DE UM FRAMEWORK PARA CLASSIFICAÇÃO DE VULNERABILIDADES RELACIONADAS Ã VIRTUALIZAÇÃO

Uma necessidade identificada ao se lidar com vulnerabilidades de virtualização é a categorização destas vulnerabilidades de acordo com a funcionalidade que a gerou e a entidade que possui esta funcionalidade. Ou seja, ao se deparar com uma vulnerabilidade, identificar qual a funcionalidade e entidade responsáveis por esta vulnerabilidade. Portanto, classificar estas vulnerabilidades partindo da entidade, e de suas funcionalidades é fundamental para mapear os principais pontos inseguros em um ambiente virtual.

Para identificar os componentes envolvidos, suas funcionalidades e vulnerabilidades, emprega-se o comparativo entre os guias de segurança em virtualização realizado em **3.2** e diversas outras obras relacionadas à segurança da virtualização e computação em nuvem, como (GONZALEZ et al., 2012) que apresenta uma análise quantitativa do estado da segurança da computação em nuvem, em que um dos pontos abordados é a virtualização. A obra de (VIMAL V., 2009) que relata diversas vulnerabilidades da virtualização e (WILLIAMS e CROSS, 2010) da IBM que também tratam das vulnerabilidades do ambiente de virtualização correlacionando com algumas vulnerabilidades encontradas no CVE. Como exemplo (GONZALEZ et.al, 2012) apresenta uma proposta de categorização para as vulnerabilidades na virtualização dividindo-as em cinco categorias como ilustrado na **Figura 38**.

Figura 38 - Taxionomia de Vulnerabilidades na Virtualização

Fonte: (GONZALEZ et. al, 2012).

Nesta classificação proposta por (GONZALEZ et. al, 2012) houve uma distinção por categorias de vulnerabilidades, não considerando as entidadades as quais elas pertencem. Por exemplo, é possível observar que o vazamento de dados pode ser proveniente de uma falha de isolamento, lógico em nível de armazenamento, rede, processamento, ou físico. O mesmo ocorre com o

ataque entre VM's que pode ser categorizado como falha do isolamento de rede.

Considerando isto, é necessário evidenciar as entidades envolvidas no processo de virtualização de recursos para categorizar suas funcionalidades e vulnerabilidades adequadamente. Primeiramente, com base nos guias utilizados no comparativo é necessário considerar os dois tipos de arquitetura utilizados na virtualização (**Apêndice A**). Os componentes são:

- ***Host* Físico**: provê recursos para todo o ambiente virtual;
- ***Hypervisor*:** responsável por orquestrar as VM's e permitir acesso destas ao *host* físico;
- **Máquinas Virtuais:** abstração de SO que deve possuir, no mínimo, o mesmo nível de segurança de que um SO não virtualizado;
- **Sistema Operacional (Depende do tipo de virtualização):** para uma arquitetura de virtualização do tipo 2, deve-se assegurar o SO que hospeda o ambiente virtual.

Estas são as quatro principais entidades relacionadas a virtualização, esta proposta visa elencar as áreas que necessitam segurança para cada uma destas entidades, elencando os problemas específicos a cada uma delas. No entanto, esta tarefa é algo que demanda um aprofundamento maior para levantar e classificar os problemas relacionados de cada uma, o que foge um pouco do objetivo deste trabalho. Portanto, esta proposta é um trabalho futuro que se é apresentado como um projeto neste trabalho para sugestões dos membros da banca.

3.5 CONSIDERAÇÕES PARCIAIS

Este capítulo teve como objetivo fornecer um embasamento teórico mais especializado em relação aos objetivos do trabalho, abordando aspectos de segurança em computação em nuvem com a apresentação e comparação dos guias de segurança, e também a apresentação e comparação dos guias de segurança em virtualização. Com isto, o estudo da segurança dos *hypervisors* utilizados pelo OpenStack é facilitado, permitindo o conhecimento dos princípios mais básicos de um *hypervisor* até aspectos de segurança relacionados. E para finalizar, o comparativo dos *bugs* e vulnerabilidades dos três principais *hypervisors* utilizados pelo OpenStack tem como objetivo informar a situação de segurança atual de cada um deles, comparando e discutindo os resultados para averiguar a necessidade de estudo e resolução de problemas nesta área.

O capítulo **4** aborda aspectos mais específicos de segurança na plataforma OpenStack, aprofundando com uma abordagem mais prática utilizando o *hypervisors* qemu/KVM, abordando os problemas especificados no trabalho de conclusão I como também realizando operações básicas para efeito de testes de segurança da plataforma.

4 ASPECTOS DE SEGURANÇA NO OPENSTACK GRIZZLY

Após o estudo das documentações dos guias de segurança em computação em nuvem e segurança em virtualização, foi iniciado o estudo prático do OpenStack com a implementação de uma nuvem privada para testes. A instalação da plataforma é descrita na subseção **4.1**, descrevendo os passos necessários e inclusive a topologia utilizada. A topologia utilizada segue a implentação *multi-node* do DevStack[44], que se é uma instalação do OpenStack por *shell script* voltada para desenvolvedores. Este capítulo aborda na subseção **4.2** os problemas elencados no TCC-I, revisando e explicando soluções que foram dadas aos mesmos. Em seguida apresenta configurações de segurança no hypervisor Qemu para garantir o isolamento das máquinas virtuais, com a solução sVirt.

4.1 INSTALAÇÃO DO OPENSTACK COM O DEVSTACK MULTI-NODE

O **primeiro passo** é a instalação nas máquinas dos SO's recomendados e documentados para a utilização do DevStack, eles são:

- Ubuntu 12.04[45].
- Fedora 18[46]; e
- CentOS/RHEL[47].

O SO escolhido para a instalação da plataforma foi o Ubuntu 12.04, justamente por esta ser uma plataforma mais popular e possuir maior acervo de informações para diversos procedimentos. Ainda é possível utilizar os SO's: OpenSUSE e Debian, entretanto os *scripts* do DevStack não configuram a plataforma nestes sistemas operacionais.

---

[44] http://devstack.org/
[45] http://www.ubuntu.com/download
[46] http://fedoraproject.org/get-fedora
[47] http://www.centos.org/

O **segundo passo** para a instalação da plataforma é instalação do *git*[48] e do *yum*[49] para cada um dos nós. O *git* é um sistema de controle de versão utilizado pelo OpenStack e o *yum* é um gerenciador automático para instalação, atualização e remoção de pacotes no GNU/Linux.

Após a instalação do *git* e do *yum*, é necessário realizar o *download* dos *scripts* que realizam a configuração automática do OpenStack no Ubuntu, este passo também deve ser realizado para todos os nós. A **Figura 39** apresenta a execução deste passo:

Figura 39 – Instalação Arquivos DevStack

```
bruno@bruno-W150HNM-W170HN:~$ git clone https://github.com/openstack-dev/devstack.git
Cloning into 'devstack'...
remote: Counting objects: 13154, done.
remote: Compressing objects: 100% (5811/5811), done.
remote: Total 13154 (delta 9196), reused 10930 (delta 7240)
Receiving objects: 100% (13154/13154), 2.38 MiB | 297 KiB/s, done.
Resolving deltas: 100% (9196/9196), done.
bruno@bruno-W150HNM-W170HN:~$
```

Fonte: Próprio Autor.

Em seguida, o **terceiro passo**, e o mais importante é o preparo dos arquivos de configuração utilizados pelo *script* de instalação do DevStack, o arquivo “localrc”. O localrc do nó *Cluster Controller* foi realizado com a seguinte configuração:

[48] https://github.com/
[49] http://yum.baseurl.org/

Figura 40 – Arquivo localrc do Nó do Cluster Controller

```
1 HOST_IP=192.168.0.100
2 FLAT_INTERFACE=eth0
3 FIXED_RANGE=10.4.128.0/20
4 FIXED_NETWORK_SIZE=4096
5 FLOATING_RANGE=192.168.42.128/25
6 MULTI_HOST=1
7 LOGFILE=/opt/stack/logs/stack.sh
8 ADMIN_PASSWORD=labstack
9 MYSQL_PASSWORD=supersecret
10 RABBIT_PASSWORD=supersecrete
11 SERVICE_PASSWORD=supersecrete
12 SERVICE_TOKEN=xyzpdqlazydog
```

Adaptado de: (DEVSTACK, 2013).

O arquivo de configuração localrc define algumas opções da instalação que o DevStack realizará nos *scripts*, estas informações são importantes para o funcionamento da instalação. Nas cinco primeiras linhas do arquivo são definidas configurações de rede para a instalação, como endereço do *host*, interface, entre outras. A linha seis define que a execução dos *scripts* ocorrerá em modo multi-nós, um importante parâmetro cujo esquecimento resulta em fracasso da instalação. A linha sete define o local em que será armazenado o *log* da instalação e a linha 8 em até a 11 define as senhas dos serviços, com a *token* de serviço sendo definida na linha 12, estas opções podem ser configuradas de acordo com o usuário. Em seguida é realizada a configuração do(s) nó(s) *Compute*, nesta instalação foi utilizado somente um nó, a **Figura 41** apresenta o arquivo de configuração localrc.

Figura 41 – Arquivo localrc do Nó Compute

```
1 HOST_IP=192.168.0.101 # change this per compute node
2 FLAT_INTERFACE=eth0
3 FIXED_RANGE=10.4.128.0/20
4 FIXED_NETWORK_SIZE=4096
5 FLOATING_RANGE=192.168.42.128/25
6 MULTI_HOST=1
7 LOGFILE=/opt/stack/logs/stack.sh.log
8 ADMIN_PASSWORD=labstack
9 MYSQL_PASSWORD=supersecret
10 RABBIT_PASSWORD=supersecrete
11 SERVICE_PASSWORD=supersecrete
12 SERVICE_TOKEN=xyzpdqlazydog
13 DATABASE_TYPE=mysql
14 SERVICE_HOST=192.168.0.100
15 MYSQL_HOST=192.168.0.100
16 RABBIT_HOST=192.168.0.100
17 GLANCE_HOSTPORT=192.168.0.100
18 ENABLED_SERVICES=n-cpu,n-net,n-api,c-sch,c-api,c-vol
```

Adaptado de: (DEVSTACK, 2013).

A configuração do localrc do *Compute Node* é bem semelhante ao do *Cluster Controller Node*, com a adição das linhas 14 em diante. Da linha 14 à 17 são definidos os serviços que estão hospedados no *Controller*, e na linha 18 é realizada a liberação de serviços para o nó.

Com os arquivos devidamente configurados no *Controller Node* e *Compute Node*, então é necessário executar o **quarto passo**, que é a execução do *script* que realiza a instalação da plataforma, este *script* deve ser executado em ambos os nós. A **Figura 42** apresenta a execução deste passo.

Figura 42 – Execução do Script Stack.sh

```
bruno@bruno-W150HNM-W170HN:~$ cd devstack/
bruno@bruno-W150HNM-W170HN:~/devstack$ ./stack.sh
```

Fonte: Próprio Autor.

O *script* executará instalando todos os serviços e componentes necessários para a plataforma e ao final da execução apresenta uma

mensagem de sucesso apresentando os *links* para a utilização da plataforma via *dashboard*, o Horizon. A **Figura 43** apresenta esta mensagem.

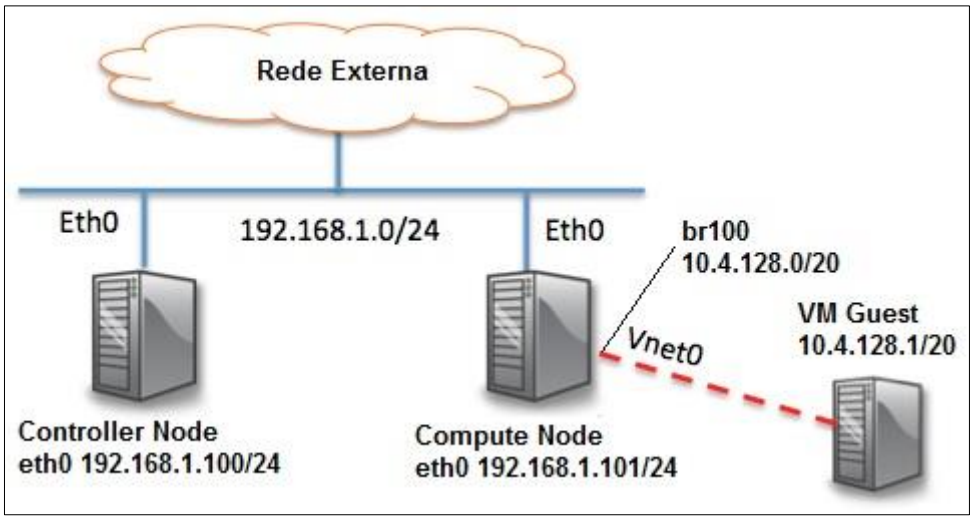
Figura 43 – Instalação Bem Sucedida

```
Horizon is now available at http://192.168.0.100/
Keystone is serving at http://192.168.0.100:5000/v2.0/
Examples on using novaclient command line is in exercise.sh
The default users are: admin and demo
The password: SENHA_DEFINIDA_NO_LOCALRC
This is your host ip: 192.168.0.100
bruno@bruno-W150HNM-W170HN:~/devstack$
```

Fonte: Próprio Autor.

Com isto é possível acessar a plataforma via o componente Horizon ou pelas API's tanto no *Controller Node* quanto no *Compute Node*. A topologia de rede utilizada para a implementação da nuvem foi a de multi-nós proposta pelo DevStack. O DevStack é uma instalação por *script* do OpenStack mantida por desenvolvedores para elaboração de protótipos e também depurações da plataforma (SALISBURY, 2013). A instalação de multi-nós do DevStack utiliza dois nós para a execução da plataforma, um nó denominado *Cluster Controller* e o nó *Compute*. A **Figura 44** apresenta a topologia da rede.

Figura 44 – Topologia da Rede



Fonte: Próprio Autor.

A **Figura 44** apresenta a topologia da rede utilizada na instalação do OpenStack com os *scripts* do DevStack. A instalação é realizada em uma rede *flat*, ou seja, é uma rede em que todos os componentes estão ligados diretamente um ao outro. O próprio (DEVSTACK, 2013) alerta que esta configuração não é interessante para se utilizar em uma nuvem de produção, isto é, esta é uma configuração apropriada para a realização de testes na plataforma. Nesta configuração o *Compute Node* é o *gateway* das VM's que pode realizar um NAT (*Network Address Translation)* para que VM's tenham acesso à internet.

4.2 REVISÃO DOS PROBLEMAS ELENCADOS NO TCC-I

As questões elencadas para o trabalho de conclusão II em ordem de prioridade são:

1. Verificar como é o procedimento de backup, recuperação e exclusão dos dados dos usuários, possivelmente integrar ao projeto um algoritmo que seja apropriado para a exclusão segura dos dados dos usuários;

2. O OpenStack não realiza a cifragem dos dados antes de armazená-los na nuvem, uma opção para garantir a segurança dos dados armazenados seria a utilização de um mecanismo de cifragem de dados, visando a segurança dos dados armazenados na nuvem;
3. Analisar de que maneira é realizado o armazenamento das senhas no serviço de autenticação do Swift, o swauth. Se necessário, implementar um mecanismo de armazenamento de senhas segundo recomendações dos guias de segurança (NIST, CSA e ENISA);
4. Propor uma arquitetura com o controle de acesso e políticas centralizadas no Keystone, facilitando a manutenção da segurança na nuvem;
5. (Opcional) Proposta de uma arquitetura para tratar a questão do uso indevido das tokens por componentes utilizados pelos usuários.
6. Outro aspecto a ser considerado é a questão da segurança das imagens da VM, que apesar de não ser uma questão listada na primeira parte deste trabalho é um problema a ser considerado. O intuito desta questão é verificar qual o suporte que a plataforma oferece para fornecer segurança para as imagens da VM. Uma análise mais completa deste aspecto é realizada na seção **4.3**.

4.2.1 BACKUP, RECUPERAÇÃO E EXCLUSÃO

A (CSA, 2011) descreve que o *backup* de dados é um mecanismo que permite prevenir a perda de dados, destruição e a sobrescrita indesejada, e alerta aos usuários da nuvem sobre esta questão. Segundo (SLIPETSKYY, 2011) no *release* Essex, não havia a possibilidade de realizar o *backup* e a recuperação dos dados, mas para o *release* Folsom este recurso foi adicionado. No Folsom e Grizzly é permitido o *backup* e a recuperação dos dados através de algumas ferramentas como:

- **Lvm2:** manipula diretamente os volumes;
- **Kpartx:** descobre a tabela de partição dentro do volume;

- **Tar:** cria um *backup* do tamanho mínimo;
- **Sha1sum:** calcula o *checksum* do *backup* para atestar sua consistência.

O benefício deste método[50] é que ele reduz o tamanho do arquivo de *backup*, por exemplo, se existe um volume do Nova com 100GB e somente 4GB estão sendo utilizados, este método cria um volume de *backup* apenas com os dados que estão sendo utilizados, economizando espaço no armazenamento da cópia de segurança.

Na tese de (SLIPETSKYY, 2011) é descrito a exclusão dos dados para o *release* Essex, e no Grizzly não foi encontrado nenhum *changelog* que notifica alguma mudança sobre o processo de exclusão dos dados do Essex para o Folsom e Grizzly, portanto assume-se que é o mesmo procedimento. Segundo (SLIPETSKYY, 2011), quando um usuário decide excluir seus dados, o OpenStack cria um arquivo de tamanho zero com a extensão *.ts (*tombstone*) e um *timestamp* com a data e hora como o nome do arquivo. Após, é executado um algoritmo que sobrescreve os *timestamps* mais antigos, excluindo-os.

O procedimento utilizado por (SLIPETSKYY, 2011) foi reproduzido no Grizzly no intuito de tentar recuperar os arquivos excluídos da mesma maneira que no *release* Essex. O primeiro passo, observado na **Figura 45** é a criação de um volume (container), e a realização do *upload* de um objeto, um arquivo de texto contendo a frase: “Este é um arquivo com texto secreto”

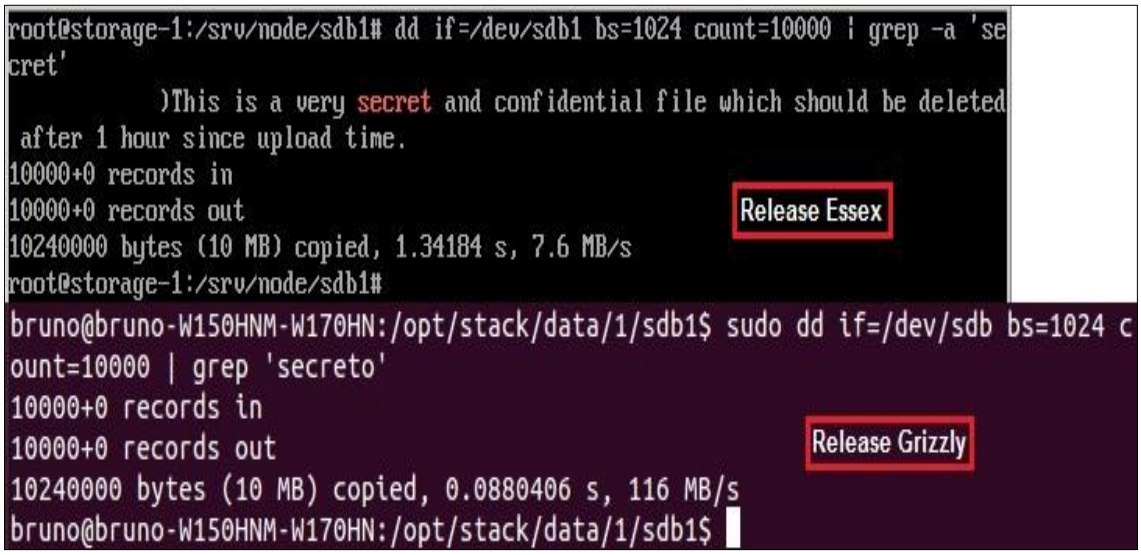
[50] http://docs.openstack.org/folsom/openstack-compute/admin/content/backup-nova-volume-disks.html

Figura 45 – Upload de Arquivo para Procedimento de Exclusão

Fonte: Próprio Autor.

Após a criação do container com o objeto é realizada a exclusão do arquivo para verificar se é possível recuperar o arquivo, ou partes dele. O procedimento utilizado por (SLIPETSKYY, 2011) é uma combinação dos comandos de GNU/Linux “dd” e “grep”. O dd realiza uma cópia *bit* a *bit*, e o comando *grep* realiza a filtragem do que se deseja copiar, os dois comandos combinados realizam o procedimento de recuperação de um arquivo excluído. O repositório de armazenamento dos objetos no *release* Essex era o /srv/node/sdb1, no *release* Grizzly, para a versão do *devstack*, o repositório está localizado em /opt/stack/data/1/sdb1. A **Figura 46** apresenta um comparativo do procedimento realizado por (SLIPETSKYY, 2011) no *release* Essex, e o mesmo procedimento realizado no Grizzly.

Figura 46 – Teste de Recuperação de Dados



Fonte: (SLIPETSKYY, 2011).

O resultado obtido com esse teste é que o problema da recuperação de dados, na documentação do Swift para o Folsom e Grizzly não há nenhum relato sobre como o problema foi corrigido. Ao detectar a falha, Slipetskyy relatou o problema no LaunchPad, e a equipe responsável pela segurança da plataforma corrigiu a falha sem fornecer detalhes. Não é possível observar o algoritmo que o OpenStack utiliza para realizar a exclusão dos arquivos. No entanto a **Figura 47** ilustra uma *trigger* que é disparada no sdb1 toda vez que um objeto ou um container é excluído.

Figura 47 – Trigger para Exclusão de Objetos e Containers

```
1   -- Describe OBJECT_DELETE
2   CREATE TRIGGER object_delete AFTER DELETE ON object
3           BEGIN
4             UPDATE container_stat
5             SET object_count = object_count - (1 - old.deleted),
6                bytes_used = bytes_used - old.size,
7                hash = chexor(hash, old.name, old.created_at);
8           END
9
```

```
1   -- Describe CONTAINER_DELETE
2   CREATE TRIGGER container_delete AFTER DELETE ON container
3           BEGIN
4             UPDATE account_stat
5             SET container_count = container_count - (1 - old.deleted),
6                object_count = object_count - old.object_count,
7                bytes_used = bytes_used - old.bytes_used,
8                hash = chexor(hash, old.name,
9                         old.put_timestamp || '-' ||
10                         old.delete_timestamp || '-' ||
11                         old.object_count || '-' || old.bytes_used);
12          END
13
```

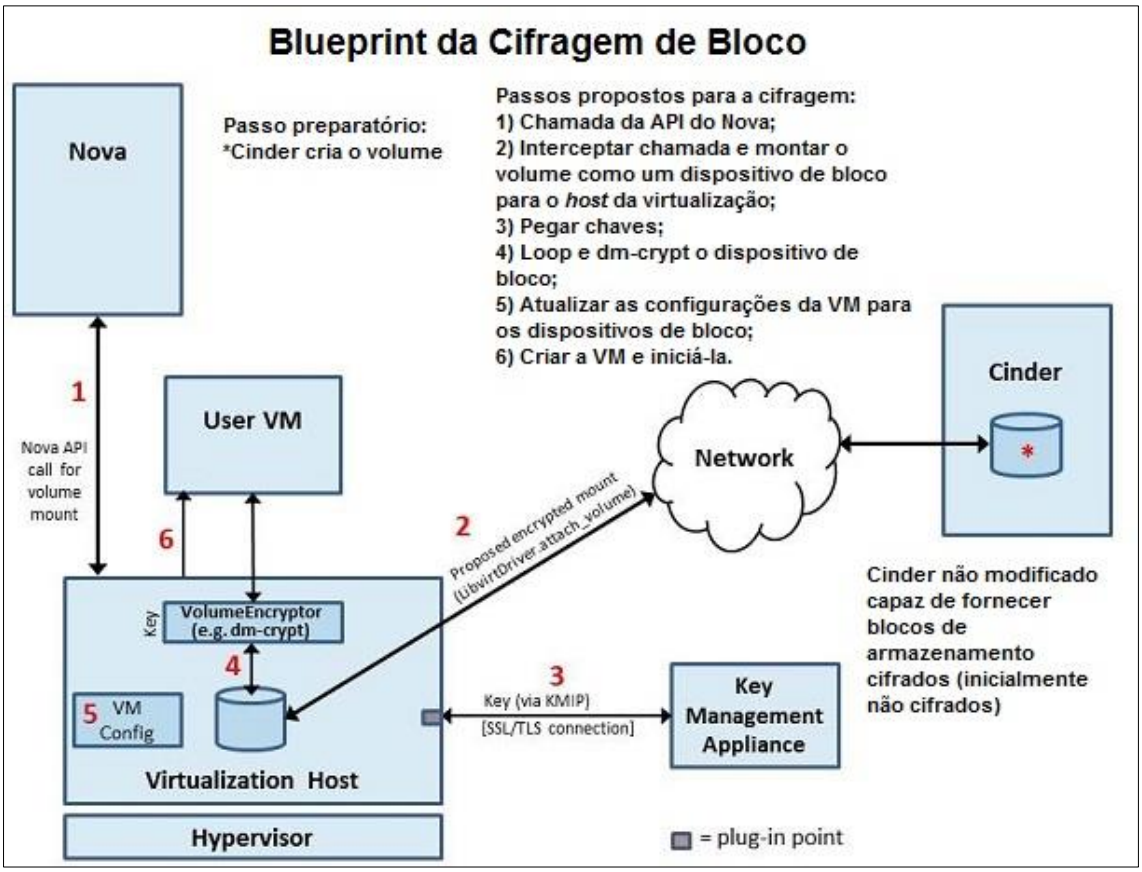
Fonte: (OPENSTACK, 2013).

A *trigger* atualiza o estado do container pela contagem de objetos e pelos *bytes* utilizados no container. Tanto para o container quanto para o objeto é realizada a atualização dos objetos e *bytes* utilizados. Em seguida o OpenStack cria um arquivo de tamanho zero com a extensão *.ts (*tombstone*) e um *timestamp* que sobrescreve o arquivo antigo, que é justamente o procedimento descrito por Slipetskyy.

4.2.2 CIFRAGEM DOS DADOS

O OpenStack por padrão não realiza a cifragem dos dados armazenados na nuvem, entretanto, há um *blueprint* (MIRANTIS, 2013), que se trata de um projeto de implementação para uma próxima versão do Swift, que disponibiliza mecanismos para realizar a cifragem de volumes e de objetos. O projeto está disponibilizado, entretanto ainda é um protótipo que está disponibilizado no próximo *release* da plataforma, o Havana. O Havana por padrão seguirá não cifrando os dados armazenados na nuvem, recomendando que os usuários realizem a cifragem dos próprios dados antes de armazená-los na nuvem. Entretanto o Havana já permite integração com mecanismos que realizam a cifragem automática dos dados armazenados na nuvem como visto em (SECURITY GUIDE OPENSTACK, 2013). As opções para cifragem dos dados estão disponibilizadas para os objetos do Swift, volumes do Cinder e dados de rede (utilizando IPSec – IP *Security Protocol*). O *blueprint* (BENJAMIN et. al., 2013) da cifragem de blocos pode ser observado na **Figura 48**.

Figura 48 – Blueprint da Cifragem de Blocos



Fonte: (BENJAMIN et. al., 2013).

O objetivo deste *blueprint* é fornecer a cifragem dos dados da VM antes que estes sejam escritos em disco, permitindo a privacidade dos dados enquanto estiverem armazenados no disco. (1) O processo para a cifragem inicia quando um usuário seleciona um volume cifrado no painel (Horizon). (2) O próximo passo do processo é a chamada da API do Nova com o comando “attach_volume”. As conexões são realizadas por SSL e o responsável por gerenciar as chaves é um componente chamado Key Management Appliance. O Nova vai mapear o dispositivo como um dispositivo de bloco no *host* de virtualização, depois que o volume é montado no *host* de virtualização, o Nova vai preparar a cifragem utilizando um dispositivo chamado no *blueprint* de VolumeEncryptor, que é responsável por preparar a cifragem para um volume. O VolumeEncryptor é invocado para cifrar os volumes, mas para isto é necessário utilizar as chaves apropriadas para cifrar o *drive*. (3) Então é necessário, no terceiro passo, utilizar um gerenciador de chaves para fornecer

as chaves apropriadas. (4) Então, no Nova são realizadas e recuperadas (5) as configurações da VM e disponibilizada (6) para o usuário.

Portanto, o problema da cifragem dos dados armazenados no OpenStack foi resolvido, entretanto o próprio OpenStack estabelece que este é um serviço que deve ser configurado pelo próprio usuário (SECURITY GUIDE OPENSTACK, 2013). Estes serviços estão como módulos opcionais que ainda não foram formalmente aceitos para serem adicionados definitivamente a plataforma, por este motivo estão disponibilizados como serviços adicionais.

4.2.3 ARMAZENAMENTO DE SENHAS

O armazenamento de senhas não é um problema exclusivo da computação em nuvem, mas de vários sistemas de informação que utilizam autenticação de senhas (CIGOJ, 2012). As práticas de segurança (RYCK et al., 2011) recomendam que o armazenamento de senhas não deve ser em texto às claras e o acesso ao local de armazenamento deve ser controlado. No *release* Essex, foi verificado por (SLIPETSKYY, 2011) e (CIGOJ, 2012) que existiam dois *middlewares* de autenticação para o Swift, o *devauth* [51] e o *swauth* [52]. Verificou-se que o *devauth* armazenava senhas em texto às claras, sendo que qualquer usuário no sistema tinha direito de acesso ao arquivo (armazenado em */etc/swift/auth.db*), e por apresentar diversos problemas de escalabilidade este meio de autenticação não era amplamente utilizado por desenvolvedores. Já no *swauth*, somente o dono do arquivo tinha permissão para acessá-lo, mas a senha também é armazenada em texto às claras.

Já no Folsom, o *middleware* de autenticação *devauth* foi descontinuado por apresentar diversos problemas e o *swauth* separado do serviço Swift. O swauth está disponível para o Grizzly e a principal mudança no *swauth* foi a utilização de um algoritmo de *hash* nas senhas armazenadas (SHA1 – *Secure*

[51] https://pypi.python.org/pypi/DevAuth/0.1
[52] https://github.com/gholt/swauth

Hash Algorithm[53]) fazendo com que as senhas sejam guardadas de maneira mais segura na nuvem. Entretanto a plataforma também oferece suporte ao SHA2 (OPENSTACK, 2013c). A **Figura 49** apresenta o trecho de código em que é utilizado o algoritmo de *hash* SHA1.

Figura 49 – Algoritmo de Hash SHA1

```
314             password = detail['auth'].split(':')[-1]
315             msg = base64.urlsafe_b64decode(unquote(token))
316             s = base64.encodestring(hmac.new(password,
317                                              msg, sha1).digest()).strip()
```

Fonte: (GITHUB, 2013).

A **Figura 49** contém o trecho de código que utiliza o algoritmo de *hash* SHA1 para cifrar o *password* do usuário e a mensagem, que é a *token*, armazenando-os em uma *string* S. Com isto, teoricamente, é garantida a segurança da senha do usuário. No entanto o resumo criptográfico do algoritmo SHA1 não é seguro, e já foi quebrado e inclusive o (ITL NIST, 2006) não aconselha o uso dessa criptografia para assinar arquivos críticos. Ou seja, desde 2006 este algoritmo de *hash* já não é mais indicado pelo NIST como um algoritmo ideal para manter a segurança de arquivos críticos. O algoritmo recomendado pelo NIST atualmente é o SHA-2, e segundo o próprio NIST em uma publicação mais recente (DANG QUYNH, 2012) o SHA-3 já está em processo de desenvolvimento, portanto é necessário para a segurança da plataforma uma atualização no uso dos algoritmos de *hash*.

O problema do armazenamento de senhas em texto as claras foi resolvido, primeiramente com a utilização do algoritmo SHA1 no Grizzly, e oferecendo suporte ao uso do SHA2 no Havana. Entretanto a solução com SHA1 para o Grizzly oferece um baixo nível de segurança, já que o algoritmo já foi quebrado (ITIL NIST, 2006). É interessante para a segurança da plataforma atualizar os componentes, se possível migrar para o *release* Havana.

[53] http://tools.ietf.org/html/rfc3174

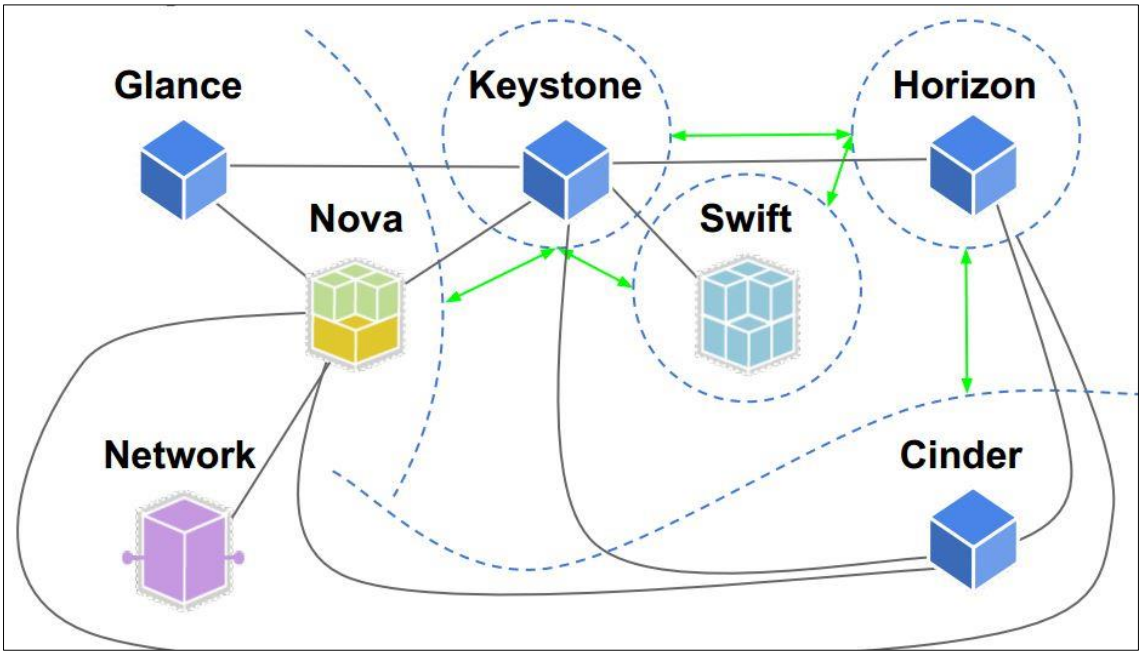
4.2.4 ARQUITETURA CENTRALIZADA DE CONTROLE DE ACESSO

A função do Keystone é atribuir o que os administradores devem fazer e quais são suas ações, entretanto quando estes papéis estão definidos diretamente no código (ou seja estão *hardcoded*), isto pode se tornar um problema para futuras configurações no componente. A questão do *hardcoding* foi apontada por (ABDUL, 2012) e demonstra porque é necessário alterar as configurações por (SLIPETSKYY, 2011). Apresentando que o papel de *reseller admin* no Swift possuía permissões inadequadas, podendo realizar qualquer coisa com qualquer conta, contanto que esteja de posse da URL da conta. Este problema foi submetido por (SLIPETSKYY, 2011) no LaunchPad e posteriormente corrigido[54] para o *release* Folsom, por meio do lançamento de um *patch* que corrige o problema para o componente.

No Grizzly, a segurança da plataforma é subdividida em domínios de segurança, alguns componentes necessitam de um tipo de segurança especializado, sendo mais adequado para suas características, como o Horizon por exemplo que é um componente Web. A **Figura 50** apresenta estes domínios de segurança definidos pelo OSSG (OpenStack Security Group) para o *release* Grizzly.

[54] https://bugs.launchpad.net/swift/+bug/747618

Figura 50 – Domínios de Segurança do OpenStack



Fonte: (OSSG, 2013).

A **Figura 50** apresenta nas linhas pontilhadas os domínios de segurança da plataforma, são quatro domínios de segurança:

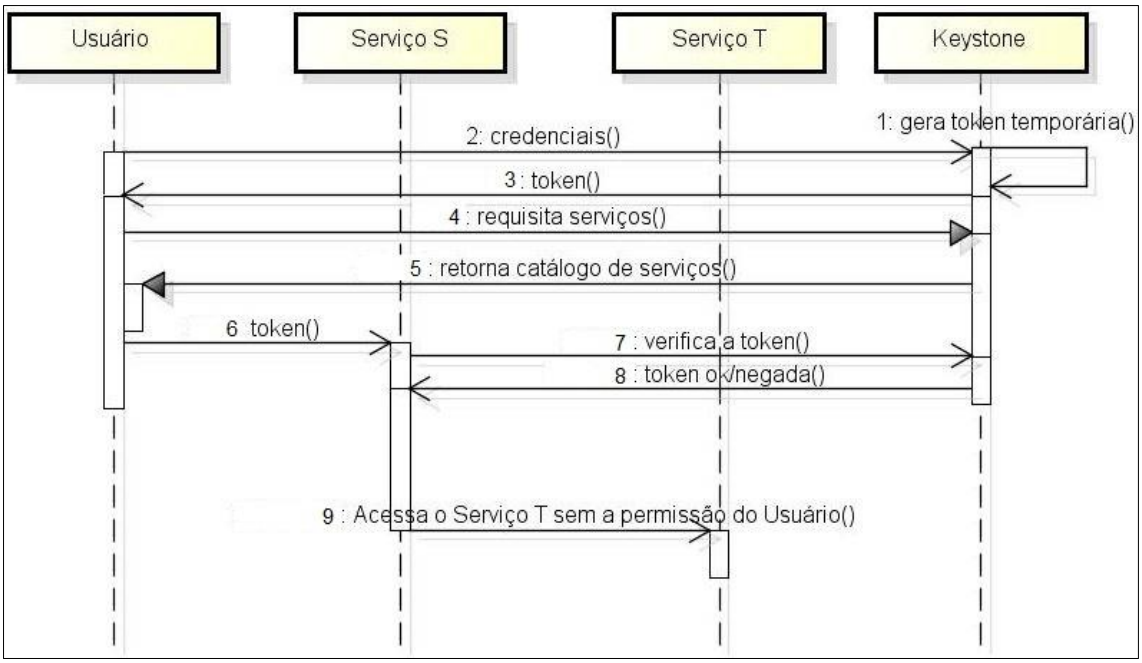
1. **Keystone:** por ser o componente central de segurança na nuvem e responsável pela autenticação das credenciais na nuvem, o Keystone está localizado em um domínio de segurança isolado;
2. **Swift:** é o componente responsável por armazenar objetos na nuvem, sua segurança foi tema de teses de metrado como de (SLIPETSKYY, 2011) e (CIGOJ, 2012), sendo necessário ter um tratamento de segurança diferenciado dos demais;
3. **Horizon:** é a principal interface de acesso do usuário com a nuvem, por ser um componente Web, possui características de segurança que devem ser tratadas de maneira especial;
4. **Demais componentes da nuvem (Nova, Glance, Cinder, Neutron):** não menos importantes que os demais, mas por necessitarem de requisitos de segurança semelhantes, estes componentes estão localizados no mesmo domínio de segurança.

A **Figura 50** também representa nas linhas contínuas o mapa dos dados na plataforma, em que é possível notar dois componentes centrais nesta representação: o Keystone, que em que toda requisição de acesso a um componente deve passar por ele, e o Nova, que é o componente central de processamento na nuvem, responsável por prover ou desprover os recursos computacionais sob demanda. Ainda na **Figura 50** há a representação de setas, que indicam as principais conexões de troca de dados entre os domínios de segurança. Portanto, o controle de acesso é realizado exclusivamente pelo Keystone, de maneira centralizada.

4.2.5 ARQUITETURA PARA O USO INDEVIDO DE TOKENS

O escopo de *tokens* em nível de *tenant*/projeto não é suficiente para isolar serviços e prevenir que outros serviços de utilizem esta *token* para acessar outros serviços. Uma vez que o usuário passa a *token* para um serviço (um componente qualquer do OpenStack), o usuário perde o controle de como a *token* é utilizada pelo serviço. Os serviços podem utilizar as *tokens* fornecidas por um usuário para acessar outros serviços e informações confidenciais do usuário sem o mesmo tomar conhecimento. Ainda é explicitado por (KISIELEWICZ, 2012), que este pode ser um comportamento desejado caso um serviço tenha que acessar outro, mas que isto deve ser explicitamente realizado com a permissão do usuário. A **Figura 51** apresenta este problema.

Figura 51 – Problema de Isolamento de Serviços.



A **Figura 51** ilustra o problema do isolamento de serviços relatado por (KISIELEWICZ, 2012), primeiramente realiza-se o procedimento normal de autenticação no Keystone (1), em que o usuário passa suas credenciais para validação no Keystone (2), que gera a *token* temporária e a retorna ao usuário (3). Este realiza a requisição dos serviços que deseja acessar (4), no caso somente o *Serviço S*. O Keystone retorna ao usuário o catálogo de serviços aos quais tem direito de acesso ao *Serviço S* (5), e então o usuário acessa o *Serviço S* passando a sua *token* (6). A falha de isolamento acontece quando o *Serviço S* realiza um acesso não autorizado pelo usuário a um outro (7) serviço qualquer (e.g., *Serviço T*), podendo, desta maneira, acessar conteúdos que sejam confidenciais do usuário ou realizar algum procedimento sem a devida autorização do usuário.

A solução para o problema foi proposta em um *blueprint* (KISIELEWICZ, 2012), na qual é proposto algumas mudanças arquiteturais no Keystone para tratar este tipo de problema. A solução consiste em (KISIELEWICZ, 2012):

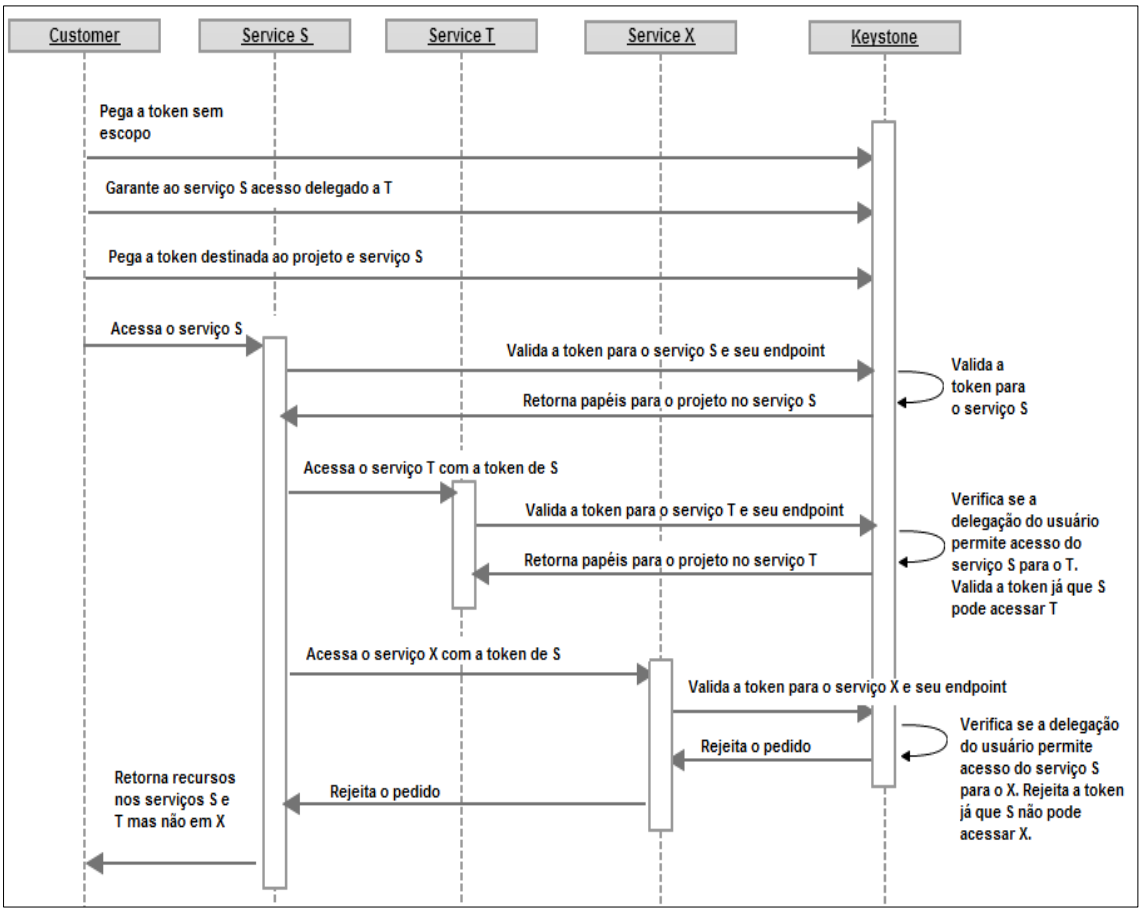
1. As *tokens* necessitam ter um escopo a nível de serviço e opcionalmente a nível de um *endpoint*. Os serviços devem aceitar apenas *tokens* com o

escopo para o ID relacionado ao serviço. Os serviços também devem conferir se o *endpoint* para o qual a *token* está destinada confere com o *endpoint* que do serviço;

2. O usuário deve estar explicitamente admitido a conceder permissões para o serviço que *token* está destinada no uso de um subconjunto de papéis em outros serviços no mesmo projeto. Ou seja, o usuário pode conceder diversas permissões somente para o mesmo projeto (ou *tenant*, a nomenclatura foi alterada do *release* Folsom para o Grizzly);
3. Cada serviço do OpenStack que aceita uma *token*, deve possuir uma conta no KeyStone. Esta conta deve possuir pelo menos uma chave criptográfica, simétrica ou assimétrica, ou um certificado X509 válido.

A API de serviços Keystone necessita ser estendida para permitir que os usuários possam conceder e revogar papéis que possuem em um serviço para ser usado por outro serviço. A API deve permitir a especificação de restrições de tempo opcionais para delegação de papéis. A **Figura 52** apresenta um diagrama de sequência do esquema proposto por (KISIELEWICZ, 2012) utilizando esta solução proposta para *tokens* não assinadas.

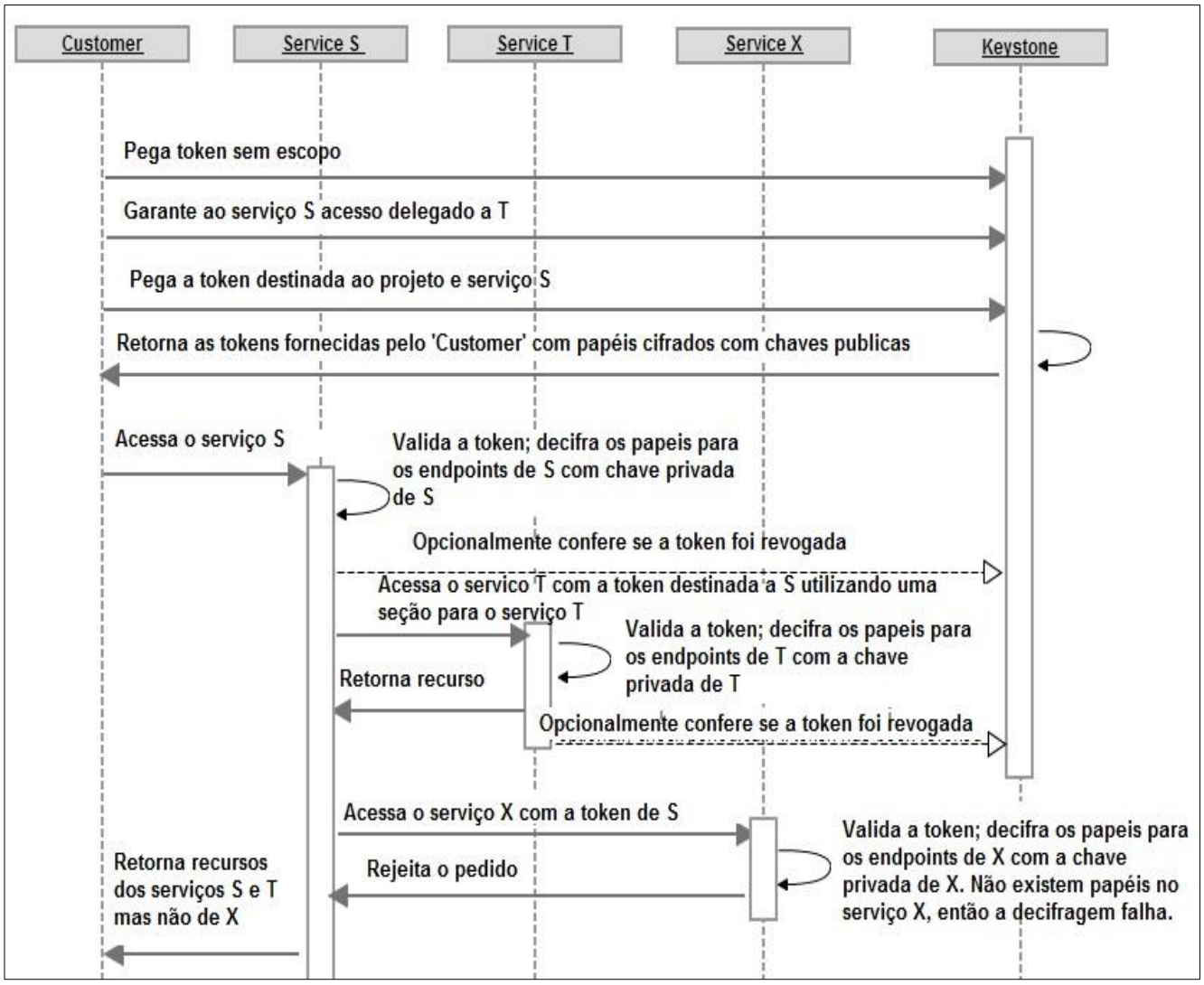
Figura 52 – Separação de Serviços para Tokens não Assinadas



Adaptado de: (KISIELEWICZ, 2012).

A **Figura 52** apresenta a tentativa de um serviço realizar um acesso não autorizado ao componente X, nesta arquitetura o usuário define previamente quais serviços uma determinada *token* tem direito de acesso, nesta representação somente aos serviços S e T. Portanto, quando o serviço S tenta realizar acesso ao serviço X à validação desta *token* é rejeitada. A **Figura 52** apresenta a arquitetura para *tokens* assinadas.

Figura 53 - Separação de Serviços para Tokens Assinadas



Adaptado de: (KISIELEWICZ, 2012).

Como na **Figura 52**, um usuário delega papéis para o serviço T no serviço S. Nenhum papel para o serviço X é delegado para S. Isto é realizado antes que m usuário tente utilizar qualquer recurso em S. Então, o usuário adquire a *token* destinada para o serviço S. Quando a *token* assinada é requisitada, o Keystone cria uma *token* de várias partes, com partes cifradas para serviços separados e assinadas pelo Keystone. Cada parte da *token* vai incluir os *endpoints* dos serviços e papéis naquele serviço. Toda vez que um serviço é acessado é realizada a validação da *token* e os papéis e *endpoints* para aquele serviços são decifrados utilizando a chave privada de cada um dos

serviços. Quando um serviço malicioso S tenta utilizar um recurso no serviço X e passa a sua *token* ou partes dela para X, o serviço X irá rejeitar, pois a *token* destinada ao serviço S não contém nenhum papel ou *endpoint* destinado ao serviço X.

O problema foi proposto no final de 2012 e resolvido para o *release* Grizzly, como pode ser observado em (YONG, 2012) para o suporte de PKI para autenticação e delegação para os serviços. Como observado nas informações da implementação da solução, era um problema de prioridade alta, que foi rapidamente aprovado e implementado, inclusive com *patch* para correção do problema no Folsom.

4.2.6 CONSIDERAÇÕES SOBRE OS PROBLEMAS

Como observado no capítulo **1**, a segurança é um fator fundamental no ambiente de nuvem, as vulnerabilidades ou falhas de segurança depois de reportadas são solucionadas rapidamente pela equipe de segurança do OpenStack (OSSG). Por consequência deste rápido desenvolvimento da plataforma, que possui um ciclo de 6 meses para o lançamento seus *releases*, a plataforma evoluiu em diversos aspectos, principalmente na parte de segurança. Os problemas levantados na primeira parte deste trabalho (TCC-I), ainda trabalhando no Folsom, foram solucionados, com suas soluções já disponibilizadas para o *release* Grizzly ou para o Havana (caso da cifragem de volumes).

Considerando isto, foi proposto pelo orientador deste trabalho, investigar a segurança das imagens das VM's na plataforma, verificando as condições de isolamento proporcionadas pelo *hypervisor*, a segurança dos dados em repouso, e analisando o tráfego desde a autenticação no Keystone até a disponibilização da imagem para o usuário. A análise deste problema é disponibilizada nas subseções **4.3** com a criação da VM, **4.4** com a análise do tráfego e **4.5** com o suporte de segurança oferecido pelo *hypervisor*.

4.3 CRIAÇÃO DE UMA VM VIA API

Para realizar a investigação da segurança das imagens da VM primeiramente é necessário realizar a criação da VM requisitando uma imagem para o Glance, esta requisição gera a necessidade de autenticação da *token*, em que é possível analisar este procedimento e verificar os dados relacionados à imagem. A tarefa de gerenciamento das VM's também é uma atividade fundamental na nuvem, portanto esta subseção também detalha os passos necessários para criar uma VM com ênfase nas opções de segurança da mesma.

A utilização das API's é fundamental para um usuário que deseja utilizar uma nuvem em produção, portanto este procedimento será realizado utilizando as API's da plataforma, para exemplificar o uso das API's. Com isto, é vital saber como é o processo de funcionamento destas API's para autenticação no Keystone, e nos testes foi verificado que esta autenticação ocorre da seguinte forma na CLI (*Command Line Interface*) do Ubuntu:

- COMPONENTE USER PASS TENANT URL_AUTH COMANDO

Por exemplo*: nova –os-username=admin –os-password=labstack –os-tenant-name=admin –os-auth-url=htttp://192.168.0.100/v2.0 \ list*

Ou seja, para todo comando na API que se deseja executar é necessário realizar a autenticação do usuário, para facilitar este trabalho o DevStack disponibiliza variáveis de ambiente em que é possível configurar estes parâmetros facilitando a execução de comandos na API, isto é apresentado na **Figura 54.**

Figura 54 – Facilitar Autenticação nas API's

```
1 //Facilitar a autenticação das API's
2 export OS_USERNAME=admin
3 export OS_PASSWORD=labstack
4 export OS_TENANT_NAME=admin
5 export OS_AUTH_URL=http://192.168.0.100:5000/v2.0
```

Fonte: Próprio Autor.

Com a execução destes comandos a utilização das API's é facilitada, bastando utilizar a forma COMPONENTE COMANDO. Com isto realizado, a tarefa de criação das VM's via API também é agilizada, a **Figura 55** apresenta um pseudo-algoritmo com as tarefas necessárias para a criação da VM.

Figura 55 – Pseudo-algoritmo para Criação da VM

```
1 //Criação de uma VM
2
3 //Habilitar ICMP e SSH para as VM's
4 1o passo: verificar os grupos de seguranca
5 //'default' é o padrao
6 -> nova secgroup-list
7
8 2o passo: adicionar regra para habilitar ssh
9 -> nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
10
11 3o passo: adicionar regra para habilitar o icmp (ping)
12 -> nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
13
14 4o passo adicionar um keypair, se não houver então criar.
15 //Criação do par de chaves
16 -> ssh-keygen -t rsa -f minhaChave.key
17     -> senha/confirmar
18 //Adicionar a chave gerada
19 -> nova keypair-add minhaChave > minhaChave.pem
20 //Confirmar
21 -> nova keipair-list
22
23 5o passo: verificar os flavors que sao os tipos das vms
24 -> nova flavor-list
25
26 6o passo: verificar as imagens disponiveis
27 -> nova image-list
28
29 7o passo: iniciar a instancia
30 -> nova boot --flavor ID_DO_FLAVOR --image ID_DA_IMAGEM
31         --key_name minhaChave --security-group default NOMEDAVM
32 //Confirmar a criacao
33 -> nova list
```

Fonte: (OPENSTACK 2013b).

A **Figura 55** criada com ajuda de um guia de configuração do Nova para o *release* Grizzly apresenta alguns passos para a criação da VM, então estes passos foram testados diversas vezes, sumarizados e organizados na **Figura 56** de modo que funcione como um “manual” para criação de uma VM via API. O primeiro passo consiste na habilitação do protocolo ICMP (*Internet Control Message Protocol*) e SSH, isto pode ser habilitado nos grupos de segurança que são feitos para criar escopos de rede para as VM’s. A **Figura 56** apresenta a execução do primeiro, segundo e terceiro passos.

Figura 56 – Adição de Regras no Grupo de Segurança

```
bruno@bruno-W150HNM-W170HN:~$ nova secgroup-list
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_reg
  import gobject._gobject
+----+---------+-------------+
| Id | Name    | Description |        1
+----+---------+-------------+
| 1  | default | default     |
+----+---------+-------------+
bruno@bruno-W150HNM-W170HN:~$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_reg
  import gobject._gobject
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |        2
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
bruno@bruno-W150HNM-W170HN:~$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_reg
  import gobject._gobject
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |        3
+-------------+-----------+---------+-----------+--------------+
bruno@bruno-W150HNM-W170HN:~$
```

Fonte: Próprio Autor.

A **Figura 56** sumariza os três primeiros passos para a criação da VM. Com estas regras definidas é necessário criar o par de chaves, pública e privada, para garantir a segurança da VM e também um posterior acesso a ela via SSH. A **Figura 57** apresenta a execução do quarto passo.

Figura 57 – Criação de Chaves

```
bruno@bruno-W150HNM-W170HN:~$ ssh-keygen -t rsa -f minhaChave.key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in minhaChave.key.
Your public key has been saved in minhaChave.key.pub.
The key fingerprint is:
cf:47:b6:89:dc:a2:9c:a8:5c:1b:ed:9c:52:93:cc:b1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|        .        |
|       oS+  o    |
|       .E+ = o   |
|      o...* =    |
|   . ..B + o     |
|    o.o.B        |
+-----------------+
bruno@bruno-W150HNM-W170HN:~$ nova keypair-add minhaChave > minhaChave.pem
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_
  import gobject._gobject
bruno@bruno-W150HNM-W170HN:~$ nova keypair-list
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_
  import gobject._gobject
+------------+-------------------------------------------------+
| Name       | Fingerprint                                     |
+------------+-------------------------------------------------+
| minhaChave | 45:1f:56:50:03:90:ef:66:71:de:b8:25:99:0f:e3:a3 |
| test_key   | ef:25:07:d5:49:a8:62:2c:be:ac:bd:bc:fc:93:8b:a6 |
+------------+-------------------------------------------------+
bruno@bruno-W150HNM-W170HN:~$
```

a) b) c) 4

Fonte: Próprio Autor.

As chaves criadas na **Figura 57** fornecem maior segurança para acesso das VM's. É importante observar que após a execução de um comando da API é emitido um *Warning* que é causado pela falta de uma biblioteca do Python no momento da execução dos testes, isto não afetou o resultado final da execução de nenhum comando da API. Tendo realizado a criação das regras de segurança para o grupo e as chaves pública e privada, então é necessário verificar os tipos de imagem e verificar se estas imagens estão disponíveis no Nova, a **Figura 58** apresenta a execução destes passos (5 e 6).

Figura 58 – Verificação das Imagens no Nova

```
bruno@bruno-W150HNM-W170HN:~$ nova flavor-list
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_register_static: as
  import gobject._gobject
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 15 | Teste     | 500       | 2    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 42 | m1.nano   | 64        | 0    | 0         |      | 1     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 84 | m1.micro  | 128       | 0    | 0         |      | 1     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
bruno@bruno-W150HNM-W170HN:~$ nova image-list
/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_register_static: as
  import gobject._gobject
+--------------------------------------+--------------------------------+--------+--------+
| ID                                   | Name                           | Status | Server |
+--------------------------------------+--------------------------------+--------+--------+
| 1957770b-dfe8-4b60-a2dc-f842d175c716 | cirros-0.3.1-x86_64-uec        | ACTIVE |        |
| d103f69a-65d3-4e79-b159-56e0cbf919d4 | cirros-0.3.1-x86_64-uec-kernel | ACTIVE |        |
| 53cb4be8-f3b3-48d7-b6b8-0b582667575b | cirros-0.3.1-x86_64-uec-ramdisk | ACTIVE |        |
+--------------------------------------+--------------------------------+--------+--------+
bruno@bruno-W150HNM-W170HN:~$
```

5

6

Fonte: Próprio Autor.

Verificados os tipos de imagens disponíveis e quais imagens estão ativas, então todos os parâmetros estão checados para lançar a VM, o comando utilizado, que pode ser verificado no passo 7 da **Figura 55** é executado na **Figura 59**.

Figura 59 – Criação da VM

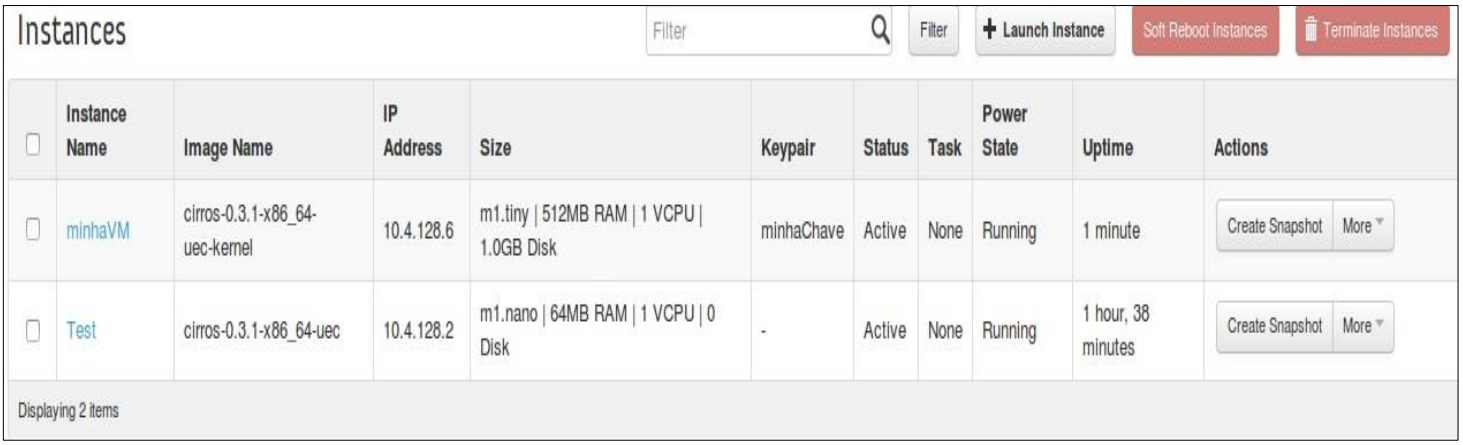
```
bruno@bruno-W150HNM-W170HN:~$ nova boot --flavor 1
--image d103f69a-65d3-4e79-b159-56e0cbf919d4
--key_name minhaChave --security_group default minhaVM

/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_r
  import gobject._gobject
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                               |
| OS-EXT-AZ:availability_zone          | nova                                 |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000004                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-STS:task_state                | scheduling                           |
| OS-EXT-STS:vm_state                  | building                             |
| OS-SRV-USG:launched_at               | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| adminPass                            | 5cMZoCdqJM2A                         |
| config_drive                         |                                      |
| created                              | 2013-10-26T22:05:07Z                 |
| flavor                               | m1.tiny                              |
| hostId                               |                                      |
| id                                   | b3ca098d-215e-41c8-979b-d82586dca69b |
| image                                | cirros-0.3.1-x86_64-uec-kernel       |
| key_name                             | minhaChave                           |
| metadata                             | {}                                   |
| name                                 | minhaVM                              |
| os-extended-volumes:volumes_attached | []                                   |
| progress                             | 0                                    |
| security_groups                      | [{u'name': u'default'}]              |
| status                               | BUILD                                |
| tenant_id                            | 54ed3d63bd5642668f09cddebd2144ad     |
| updated                              | 2013-10-26T22:05:07Z                 |
| user_id                              | 2c9dccaadd7c4d8fbd019ac445cce7eb     |
+--------------------------------------+--------------------------------------+
bruno@bruno-W150HNM-W170HN:~$
```

7

Fonte: Próprio Autor.

A **Figura 59** apresenta o comando que executa a criação da VM, apresentando logo em seguida as propriedades da VM e os seus valores, com informações técnicas que não são passadas ao usuário pelo *horizon*. Em seguida, para verificar se a criação da VM funcionou como esperado basta executar o comando “nova list” que retorna uma lista com as instâncias criadas, ou visitar o *dashboard*, neste caso hospedado no endereço do nó *Controller* 192.168.0.100, a **Figura 60** apresenta esta confirmação no Horizon.

Figura 60 – Instância Criada no Horizon



Instances

Filter | Filter | + Launch Instance | Soft Reboot Instances | Terminate Instances

Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
minhaVM	cirros-0.3.1-x86_64-uec-kernel	10.4.128.6	m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk	minhaChave	Active	None	Running	1 minute	Create Snapshot | More
Test	cirros-0.3.1-x86_64-uec	10.4.128.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	-	Active	None	Running	1 hour, 38 minutes	Create Snapshot | More

Displaying 2 items

Fonte: Próprio Autor.

A **Figura 60** confirma a criação da VM via API no OpenStack, ainda por meio das API's é possível associar um IP flutuante à máquina criada, no entanto neste exemplo ele foi realizado via o Horizon. Para verificar se os IP's flutuantes associados as VM's é necessário executar o comando nova 'floating-ip-list', apresentado na **Figura 61**.

Figura 61 – Verificação dos IP's Flutuantes

```
bruno@bruno-W150HNM-W170HN:~$ nova floating-ip-list
/usr/lib/python2.7/dist-packages/gobject/constants.py:
  import gobject._gobject
+----------------+-----------+------------+--------+
| Ip             | Server Id | Fixed Ip   | Pool   |
+----------------+-----------+------------+--------+
| 192.168.42.129 |           | 10.4.128.6 | public |
+----------------+-----------+------------+--------+
bruno@bruno-W150HNM-W170HN:~$
```

Fonte: Próprio Autor.

Portanto, agora a VM criada “minhaVM” está com o IP flutuante 192.168.42.129 associado a ela, como também o par de chaves criado “minhaChave”, permitindo o acesso via SSH na máquina.

Para maior segurança da VM pode se associar a ela um par de chaves pública e privada, proporcionando uma segurança maior na autenticação para o acesso da VM. Os grupos de segurança permitem associar regras de rede para um grupo de VM’s ligadas a um projeto ou *tenant*, proporcionando maneiras de impor restrições ao acesso das VM’s, é importante lembrar que por padrão os grupos para as VM’s vêm sem nenhuma regra, cabendo ao usuário adicionar estas regras paras as máquinas. Também é importante observar que toda VM criada deve estar associada a um grupo de segurança, mesmo que seja o padrão “*default*” que inicialmente não possui nenhuma regra definida.

4.4 MONITORAMENTO DA CRIAÇÃO DA VM COM O WIRESHARK

Para a realização do monitoramento do tráfego durante a criação da VM foi utilizada a ferramenta Wireshark[55]. Este monitoramento tem o objetivo de analisar os dados da autenticação no Keystone e a requisição da imagem para criação da VM no Glance. Foi verificado que por padrão a plataforma utiliza o protocolo HTTP para a comunicação interna entre os componentes, que não é um protocolo seguro para o tráfego dos dados das credenciais do usuário e da imagem requisitada no Glance.

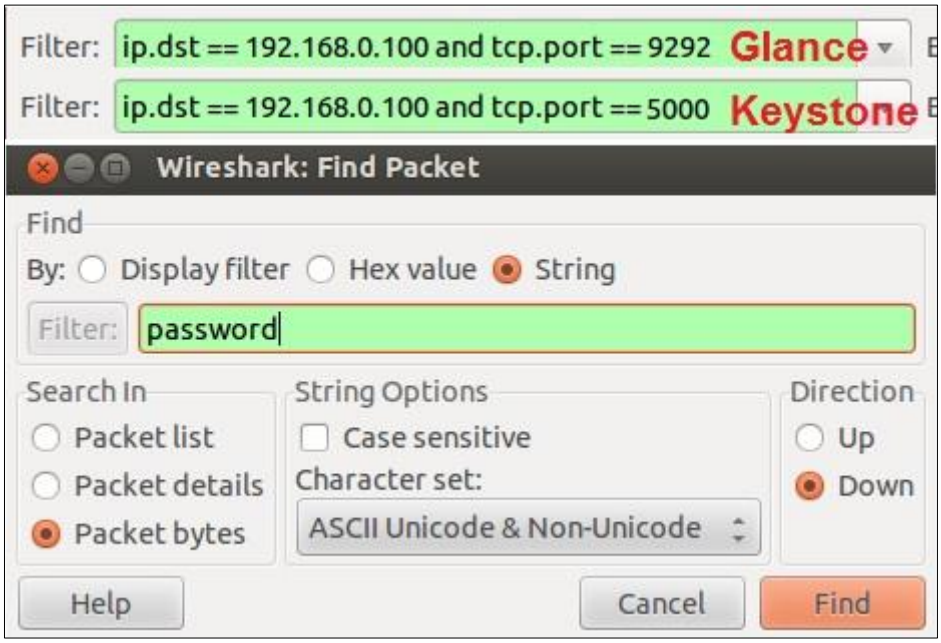
Após a instalação da ferramenta, é necessário observar as interfaces de rede que o Wireshark monitora e selecionar a interface eth0, a interface utilizada para a comunicação entre os nós *Controller* e *Compute* nesta configuração. A interface br100 das máquinas virtuais também pode ser

[55] http://www.wireshark.org/

monitorada com a abertura via linha de comando de outra instância do Wireshark.

A monitoração foi realizada nos componentes Keystone e Glance, que estão hospedados no *Cluster Controller* com portas :5000 e :9292, respectivamente. A princípio, o único filtro aplicado no Wireshark foi pelo ip e porta, listando somente os pacotes destinados aos componentes, a **Figura 62** apresenta este filtro.

Figura 62 – Filtros do Wireshark



Fonte: Próprio Autor.

Com os filtros da **Figura 62** prontos bastaram realizar procuras no conteúdo dos pacotes filtrados. Estas procuras consistiram simplesmente pela palavra-chave "*password*" e também pelo próprio *password* configurado no início da instalação, que era "semsenha". Os resultados obtidos podem ser observados na **Figura 63** e **Figura 64**.

Figura 63 – Pacote de Autenticação Keystone - Glance

```
88273 2262.095859 192.168.0.101   192.168.0.100   IPA     370 unknown 0x53 [Malformed Packet]
88274 2262.095877 192.168.0.101   192.168.0.100   TCP      66 5000 > 55753 [ACK] Seq=1 Ack=305
88397 2262.735933 192.168.0.101   192.168.0.100   IPA    8258 unknown 0x54 [Malformed Packet]
88398 2262.735984 192.168.0.101   192.168.0.100   TCP      66 55753 > 5000 [ACK] Seq=305 Ack=8
88399 2262.736014 192.168.0.101   192.168.0.100   IPA     532 unknown 0x20 [Malformed Packet]

0070  3a 35 30 30 30 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   :5000..C ontent-L
0080  65 6e 67 74 68 3a 20 31  30 36 0d 0a 43 6f 6e 74   ength: 1 06..Cont
0090  65 6e 74 2d 54 79 70 65  3a 20 61 70 70 6c 69 63   ent-Type : applic
00a0  61 74 69 6f 6e 2f 6a 73  6f 6e 0d 0a 41 63 63 65   ation/js on..Acce
00b0  70 74 2d 45 6e 63 6f 64  69 6e 67 3a 20 67 7a 69   pt-Encod ing: gzi
00c0  70 2c 20 64 65 66 6c 61  74 65 2c 20 63 6f 6d 70   p, defla te, comp
00d0  72 65 73 73 0d 0a 41 63  63 65 70 74 3a 20 2a 2f   ress..Ac cept: */
00e0  2a 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 70   *..User- Agent: p
00f0  79 74 68 6f 6e 2d 6b 65  79 73 74 6f 6e 65 63 6c   ython-ke ystonecl
0100  69 65 6e 74 0d 0a 0d 0a  7b 22 61 75 74 68 22 3a   ient.... {"auth":
0110  20 7b 22 74 65 6e 61 6e  74 4e 61 6d 65 22 3a 20    {"tenan tName":
0120  22 73 65 72 76 69 63 65  22 2c 20 22 70 61 73 73   "service ", "pass
0130  77 6f 72 64 43 72 65 64  65 6e 74 69 61 6c 73 22   wordCred entials"
0140  3a 20 7b 22 75 73 65 72  6e 61 6d 65 22 3a 20 22   : {"user name": "
0150  67 6c 61 6e 63 65 22 2c  20 22 70 61 73 73 77 6f   glance",  "passwo
0160  72 64 22 3a 20 22 73 65  6d 73 65 6e 68 61 22 7d   rd": "semsenha"}
0170  7d 7d                                              }}
```

Fonte: Próprio Autor.

Como observado na **Figura 63**, o OpenStack não utiliza canais seguros de comunicação entre os seus componentes, e com isto é possível capturar as informações de qualquer usuário na rede. Neste caso, o pacote se trata de uma autenticação da *token* do usuário do Glance para o Keystone. O mesmo fato se repete nos pacotes filtrados da comunicação com o Keystone, observado na **Figura 64.**

Figura 64 – Pacote de Autenticação Glance - Keystone

```
77600 2169.779546 192.168.0.101   192.168.0.100   IPA     370 unknown 0x53 [
77601 2169.779567 192.168.0.101   192.168.0.100   TCP      66 5000 > 55593 [
77713 2170.140682 192.168.0.101   192.168.0.100   IPA    8258 unknown 0x54 [

0080  74 65 64 5f 61 74 2c 20  64 65 6c 65 74 65 64 2c   ted_at,  deleted,
0090  20 69 6d 61 67 65 5f 69  64 2c 20 76 61 6c 75 65    image_i d, value
00a0  2c 20 6d 65 74 61 5f 64  61 74 61 29 20 56 41 4c   , meta_d ata) VAL
00b0  55 45 53 20 28 27 32 30  31 33 2d 31 30 2d 32 38   UES ('20 13-10-28
00c0  20 30 33 3a 30 32 3a 30  32 27 2c 20 27 32 30 31    03:02:0 2', '201
00d0  33 2d 31 30 2d 32 38 20  30 33 3a 30 32 3a 30 32   3-10-28  03:02:02
00e0  27 2c 20 4e 55 4c 4c 2c  20 30 2c 20 27 33 65 66   ', NULL,  0, '3ef
00f0  39 64 37 36 38 2d 66 33  35 32 2d 34 30 64 61 2d   9d768-f3 52-40da-
0100  38 39 30 63 2d 31 33 64  39 39 61 65 66 64 33 37   890c-13d 99aefd37
0110  31 27 2c 20 27 73 77 69  66 74 2b 68 74 74 70 3a   1', 'swi ft+http:
0120  2f 2f 73 65 72 76 69 63  65 25 33 41 67 6c 61 6e   //servic e%3Aglan
0130  63 65 3a 73 65 6d 73 65  6e 68 61 40 31 39 32 2e   ce:semse nha@192.
0140  31 36 38 2e 30 2e 31 30  3a 35 30 30 30 2f 76 32   168.0.100:5000/v2
0150  2e 30 2f 67 6c 61 6e 63  65 2f 33 65 66 39 64 37   .0/glanc e/3ef9d7
0160  36 38 2d 66 33 35 32 2d  34 30 64 61 2d 38 39 30   68-f352- 40da-890
0170  63 2d 31 33 64 39 39 61  65 66 64 33 37 31 27 2c   c-13d99a efd371',
0180  20 27 7b 7d 27 29                                   '{}')
```

Fonte: Próprio Autor.

O mesmo resultado pode ser observado na requisição de autenticação do Glance para o Keystone, que envia o *password* “semsenha” para o endereço do Keystone (192.168.0.100:5000/v2.0) no *Cluster Controller Node*. Esta troca de informações para autenticação também pode ser observada em pacotes de diversos componentes, comprovando que a plataforma não utiliza canais seguros de comunicação entre os nós da plataforma. A **Figura 65** apresenta mais alguns resultados da busca.

Figura 65 – Senha em Texto Puro, Wireshark

81898	2232.294662	192.168.0.101	192.168.0.100	HTTP	915	POST /auth/login/ HTTP/1.1
81899	2232.294718	192.168.0.101	192.168.0.100	TCP	66	http > 50964 [ACK] Seq=1 A
82119	2232.932586	192.168.0.101	192.168.0.100	HTTP	2373	HTTP/1.1 302 FOUND
82120	2232.932631	192.168.0.101	192.168.0.100	TCP	66	50964 > http [ACK] Seq=856
82122	2232.989970	192.168.0.101	192.168.0.100	HTTP	2173	GET /project/instances/ HT

```
22 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   "..Conne ction: k
65 65 70 2d 61 6c 69 76  65 0d 0a 43 6f 6e 74 65   eep-aliv e..Conte
6e 74 2d 54 79 70 65 3a  20 61 70 70 6c 69 63 61   nt-Type:  applica
74 69 6f 6e 2f 78 2d 77  77 77 2d 66 6f 72 6d 2d   tion/x-w ww-form-
75 72 6c 65 6e 63 6f 64  65 64 0d 0a 43 6f 6e 74   urlencod ed..Cont
65 6e 74 2d 4c 65 6e 67  74 68 3a 20 31 36 30 0d   ent-Leng th: 160.
0a 0d 0a 63 73 72 66 6d  69 64 64 6c 65 77 61 72   ...csrfm iddlewar
65 74 6f 6b 65 6e 3d 41  6d 55 68 30 73 65 59 58   etoken=A mUh0seYX
6a 65 33 5a 63 48 57 31  68 64 71 6a 4a 44 74 65   je3ZcHW1 hdqjJDte
6b 41 45 63 74 5a 30 26  6e 65 78 74 3d 25 32 46   kAEctZ0& next=%2F
70 72 6f 6a 65 63 74 25  32 46 69 6e 73 74 61 6e   project% 2Finstan
63 65 73 25 32 46 26 72  65 67 69 6f 6e 3d 68 74   ces%2F&r egion=ht
74 70 25 33 41 25 32 46  25 32 46 31 32 37 2e 30   tp%3A%2F %2F127.0
2e 30 2e 31 25 33 41 35  30 30 30 25 32 46 76 32   .0.1%3A5 000%2Fv2
2e 30 26 75 73 65 72 6e  61 6d 65 3d 61 64 6d 69   .0&usern ame=admi
6e 26 70 61 73 73 77 6f  72 64 3d 73 65 6d 73 65   n&passwo rd=semse
6e 68 61                                           nha
```

Horizon

77734	2170.161818	192.168.0.101	192.168.0.100	TCP	66	39583 > openstack-id [ACK
77735	2170.162039	192.168.0.101	192.168.0.100	TCP	420	39583 > openstack-id [PSH
77736	2170.162059	192.168.0.101	192.168.0.100	TCP	66	openstack-id > 39583 [ACK

```
00a0  63 61 74 69 6f 6e 2f 6a  73 6f 6e 0d 0a 41 63 63   cation/j son..Acc
00b0  65 70 74 2d 45 6e 63 6f  64 69 6e 67 3a 20 67 7a   ept-Enco ding: gz
00c0  69 70 2c 20 64 65 66 6c  61 74 65 2c 20 63 6f 6d   ip, defl ate, com
00d0  70 72 65 73 73 0d 0a 41  63 63 65 70 74 3a 20 61   press..A ccept: a
00e0  70 70 6c 69 63 61 74 69  6f 6e 2f 6a 73 6f 6e 0d   pplicati on/json.
00f0  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 70 79 74   .User-Ag ent: pyt
0100  68 6f 6e 2d 72 65 71 75  65 73 74 73 2f 32 2e 30   hon-requ ests/2.0
0110  2e 31 20 43 50 79 74 68  6f 6e 2f 32 2e 37 2e 33   .1 CPyth on/2.7.3
0120  20 4c 69 6e 75 78 2f 33  2e 35 2e 30 2d 32 33 2d    Linux/3 .5.0-23-
0130  67 65 6e 65 72 69 63 0d  0a 0d 0a 7b 22 61 75 74   generic. ...{"aut
0140  68 22 3a 20 7b 22 74 65  6e 61 6e 74 4e 61 6d 65   h": {"te nantName
0150  22 3a 20 22 73 65 72 76  69 63 65 22 2c 20 22 70   ": "serv ice", "p
0160  61 73 73 77 6f 72 64 43  72 65 64 65 6e 74 69 61   asswordC redentia
0170  6c 73 22 3a 20 7b 22 75  73 65 72 6e 61 6d 65 22   ls": {"u sername"
0180  3a 20 22 73 77 69 66 74  22 2c 20 22 70 61 73 73   : "swift ", "pass
0190  77 6f 72 64 22 3a 20 22  73 65 6d 73 65 6e 68 61   word": " semsenha
01a0  22 7d 7d 7d                                        "}}}
```

Swift

78228	2171.353177	127.0.0.1	127.0.0.1	MySQL	390	Request Query
78229	2171.353414	127.0.0.1	127.0.0.1	MySQL	77	Response OK
78230	2171.353832	127.0.0.1	127.0.0.1	MySQL	77	Request Query
78231	2171.389993	127.0.0.1	127.0.0.1	MySQL	77	Response OK
78232	2171.390180	127.0.0.1	127.0.0.1	MySQL	79	Request Query
78233	2171.390246	127.0.0.1	127.0.0.1	MySQL	77	Response OK
78234	2171.393902	127.0.0.1	127.0.0.1	MySQL	79	Request Query

```
80  74 65 64 5f 61 74 2c 20  64 65 6c 65 74 65 64 2c   ted_at,  deleted,
90  20 69 6d 61 67 65 5f 69  64 2c 20 76 61 6c 75 65    image_i d, value
a0  2c 20 6d 65 74 61 5f 64  61 74 61 29 20 56 41 4c   , meta_d ata) VAL
b0  55 45 53 20 28 27 32 30  31 33 2d 31 30 2d 32 38   UES ('20 13-10-28
c0  20 30 33 3a 30 32 3a 30  32 27 2c 20 27 32 30 31    03:02:0 2', '201
d0  33 2d 31 30 2d 32 38 20  30 33 3a 30 32 3a 30 32   3-10-28  03:02:02
e0  27 2c 20 4e 55 4c 4c 2c  20 30 2c 20 27 33 65 66   ', NULL,  0, '3ef
f0  39 64 37 36 38 2d 66 33  35 32 2d 34 30 64 61 2d   9d768-f3 52-40da-
00  38 39 30 63 2d 31 33 64  39 39 61 65 66 64 33 37   890c-13d 99aefd37
10  31 27 2c 20 27 73 77 69  66 74 2b 68 74 74 70 3a   1', 'swi ft+http:
20  2f 2f 73 65 72 76 69 63  65 25 33 41 67 6c 61 6e   //servic e%3Aglan
30  63 65 3a 73 65 6d 73 65  6e 68 61 40 31 39 32 2e   ce:semse nha@192.
40  31 36 38 2e 30 2e 31 30  3a 35 30 30 30 2f 76 32   168.0.10 :5000/v2
50  2e 30 2f 67 6c 61 6e 63  65 2f 33 65 66 39 64 37   .0/glanc e/3ef9d7
60  36 38 2d 66 33 35 32 2d  34 30 64 61 2d 38 39 30   68-f352- 40da-890
```

Glance realizando busca na base de dados MySQL

Fonte: Próprio Autor.

A **Figura 65** apresenta o mesmo ocorrido na filtragem por outros componentes, o que comprova a necessidade de segurança nas comunicações internas da plataforma. Toda requisição e confirmação de autenticação é realizada com o usuário e senha em texto as claras, sendo que qualquer usuário interno malicioso pode capturar credenciais de outros usuários na nuvem. Uma alternativa para cifrar o cabeçalho das imagens no Glance é disponibilizada em sua API, que oferece a possibilidade de cifragem que pode ser observado na **Figura 66**.

Figura 66 – Opção de Segurança na API do Glance

```
# ================= Security Options ==========================

# AES key for encrypting store 'location' metadata, including
# -- if used -- Swift or S3 credentials
# Should be set to a random string of length 16, 24 or 32 bytes
metadata_encryption_key = <16, 24 or 32 char registry metadata key>
```

Fonte: Próprio Autor.

Esta opção de segurança disponibilizada pela API obscurece o acesso as informações da imagem, além da utilização da biblioteca AES para cifrar a localização dos metadados da imagem. Além disso não é fornecido pela plataforma nenhuma outra informação sobre o procedimento, porém com o usuário e a senha circulando sem nenhuma cifragem, um usuário malicioso pode capturar estas credenciais e requisitar qualquer imagem pertencente a estas credenciais. A **Figura 67** apresenta a confirmação da cifragem dos dados da imagem.

Figura 67 – Confirmação da Cifragem da Imagem

```
bruno@bruno-W150HNM-W170HN: ~/devstack
2013-11-03 21:27:22 ++ euca-bundle-image -i /home/bruno
rros-0.3.1-x86_64-uec/cirros-0.3.1-x86_64-blank.img -d
s/images/s3-materials/cirros-0.3.1
2013-11-03 21:27:22 Checking image
2013-11-03 21:27:22 Encrypting image
2013-11-03 21:27:22 Splitting image...
2013-11-03 21:27:22 Part: cirros-0.3.1-x86_64-blank.img
2013-11-03 21:27:22 Generating manifest /home/bruno/dev
erials/cirros-0.3.1/cirros-0.3.1-x86_64-blank.img.manif
2013-11-03 21:27:22 ++ [[ stack == \u\n\s\t\a\c\k ]]
2013-11-03 21:27:22 ++ [[ stack == \c\l\e\a\n ]]
2013-11-03 21:27:22 + [[ -x /home/bruno/devstack/local.
2013-11-03 21:27:22 + service_check
2013-11-03 21:27:22 + local service
2013-11-03 21:27:22 + local failures
2013-11-03 21:27:22 + SCREEN_NAME=stack
2013-11-03 21:27:22 + SERVICE_DIR=/opt/stack/status
2013-11-03 21:27:22 + [[ ! -d /opt/stack/status/stack ]
2013-11-03 21:27:22 ++ ls '/opt/stack/status/stack/*.fa
2013-11-03 21:27:22 + failures=
2013-11-03 21:27:22 + '[' -n '' ']'
2013-11-03 21:27:22 + set +o xtrace
2013-11-03 21:27:22 stack.sh completed in 437 seconds.
```

Fonte: Próprio Autor

A cifragem obscurece as informações presentes no cabeçalho da imagem, entretanto o problema na autenticação com o Keystone persiste. Por meio de uma avaliação dos arquivos de configuração do Keystone (etc/Keystone/keystone.conf), foi possível observar que o OpenStack permite a utilização do protocolo HTTPS com os protocolos criptográficos SSL/TLS, verificando a autenticidade das chaves por meio de certificados digitais. E isto, resolve o problema da comunicação insegura entre os componentes, que não estavam configurados para utilizar isto. A **Figura 68** apresenta estas configurações.

Figura 68 – Configuração da API Keystone

```
305 [ssl]
306 enable = True
307 certfile = /etc/keystone/ssl/certs/keystone.pem
308 keyfile = /etc/keystone/ssl/private/keystonekey.pem
309 ca_certs = /etc/keystone/ssl/certs/ca.pem
310 ca_key = /etc/keystone/ssl/private/cakey.pem
311 key_size = 1024
312 valid_days = 3650
313 cert_required = False
314 cert_subject = /C=US/ST=Unset/L=Unset/O=Unset/CN=localhost
```

Fonte: Próprio Autor,

Portanto, como apresentado na **Figura 68**, a plataforma fornece suporte à conexão segura entre os componentes, bastando realizar a configuração para utilizá-los. Os *scripts* de instalação do DevStack sobrescrevem os arquivos de configuração dos componentes a cada execução do arquivo stack.sh, que é o arquivo de inicialização da plataforma. Uma contribuição deste trabalho é a alteração no script “stack.sh”, indicando que o protocolo a ser utilizado na comunicação entre os componentes por padrão seja o HTTPS, como também indicando no arquivo *localrc* o caminho para os certificados SSL para cada componente, que são copiados para o local indicado como demonstrado na **Figura 68**, e forçando o uso de *tokens* PKI ao invés de *tokens* UUID (Universally Unique IDentifier) por meio do comando: “KEYSTONE_TOKEN_FORMAT=PKI”, para que as chaves públicas sejam validadas pelos certificados (OPENSTACK, 2013g). Esta configuração será repassada ao LaunchPad como uma recomendação, para que o *script* execute por padrão uma configuração mais segura da plataforma. Após a configuração dos *scripts* e a inicialização da plataforma, é necessário executar na API do Keystone o comando “keystone-manage” ssl setup, mostrado na **Figura 69**.

Figura 69 – Geração dos Certificados SSL

```
bruno@bruno-W150HNM-W170HN:~$ keystone-manage ssl_setup
2013-11-04 22:25:44.798 22733 INFO keystone.common.openssl [-] openssl gen
Generating RSA private key, 1024 bit long modulus
.........++++++
.++++++
e is 65537 (0x10001)
2013-11-04 22:25:44.831 22733 INFO keystone.common.openssl [-] openssl req
g /etc/keystone/ssl/certs/openssl.conf -subj /C=US/ST=Unset/L=Unset/O=Unset
2013-11-04 22:25:44.845 22733 INFO keystone.common.openssl [-] openssl ca
ays 3650d -cert /etc/keystone/ssl/certs/ca.pem -keyfile /etc/keystone/ssl/
Using configuration from /etc/keystone/ssl/certs/openssl.conf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :ASN.1 12:'Unset'
localityName          :ASN.1 12:'Unset'
organizationName      :ASN.1 12:'Unset'
commonName            :ASN.1 12:'localhost'
Certificate is to be certified until Nov  3 00:25:44 2023 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
bruno@bruno-W150HNM-W170HN:~$
```

Fonte: Próprio Autor.

A **Figura 69** apresenta o comando na API do Keystone, após compilação da plataforma, que gera os certificados SSL configurados na **Figura 68**. É importante observar que para o DevStack, os certificados utilizados são gerados pelo próprio Keystone, mas que para uma nuvem de produção é recomendado utilizar certificados externos para validar as *tokens* PKI. Por ser uma abordagem pra efeito de testes foi utilizado os certificados gerados pelo Keystone. A **Figura 70** apresenta o resultado da configuração.

Figura 70 – Autenticação no Keystone

Stream Content

Antes

```
POST /v2.0/tokens HTTP/1.1
Host: 192.168.0.100:5000
Content-Length: 106
Content-Type: application/json
Accept-Encoding: gzip, deflate, compress
Accept: */*
User-Agent: python-keystoneclient

{"auth": {"tenantName": "service", "passwordCredentials": {"username": "glance", "password": "semsenha"}}}HTTP/1.1 200 OK
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 8528
Date: Sun, 03 Nov 2013 23:27:10 GMT
```

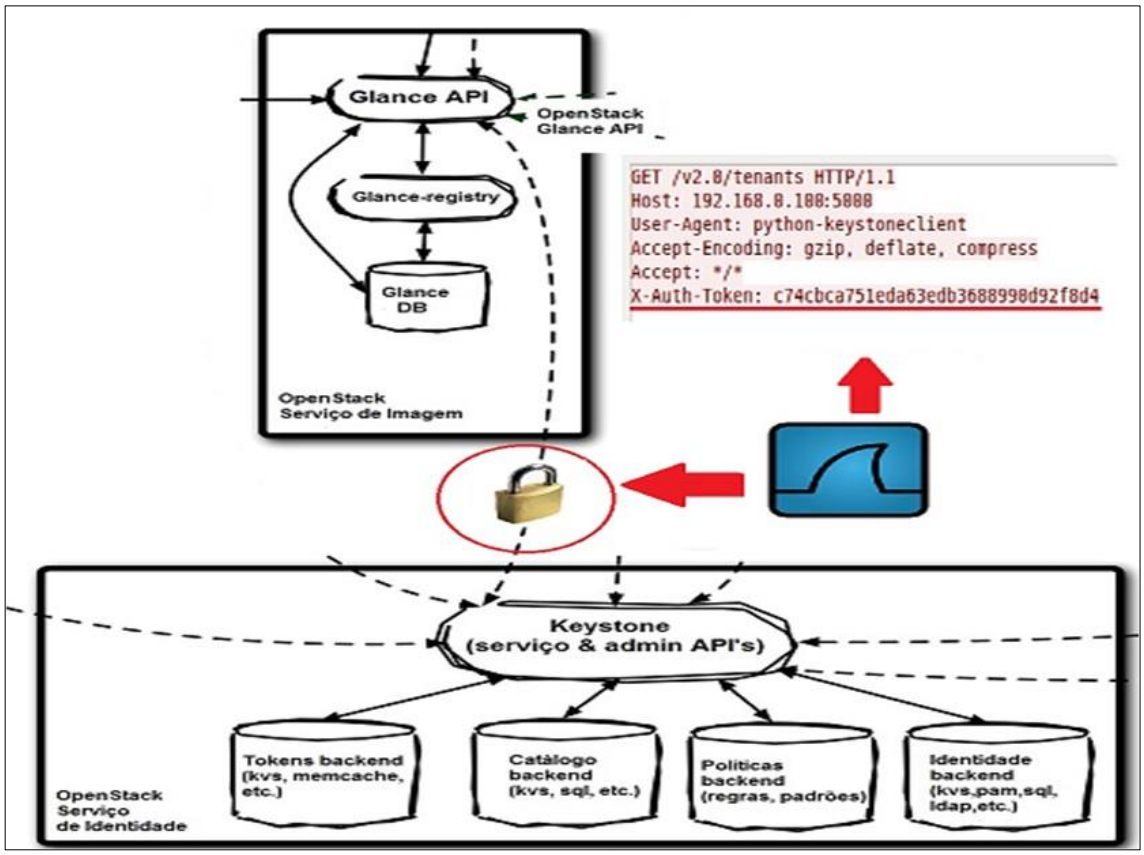
Stream Content

Depois

```
GET /v2.0/tenants HTTP/1.1
Host: 192.168.0.100:5000
User-Agent: python-keystoneclient
Accept-Encoding: gzip, deflate, compress
Accept: */*
X-Auth-Token: c74cbca751eda63edb3688990d92f8d4

HTTP/1.1 200 OK
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 231
Date: Tue, 05 Nov 2013 00:29:11 GMT

{"tenants_links": [], "tenants": [{"description": null, "enabled": true, "id":
"ef2c046522d54dd19b6041608df3ad30", "name": "demo"}, {"description": null, "enabled":
true, "id": "28fe5e8c9ad649f1801e32f78786b6f3", "name": "admin"}]}
```

Portanto, a **Figura 70** apresenta o comparativo da autenticação do Glance no Keystone antes e depois da configuração utilizando SSL e certificados de autoridade. Desta maneira, a comunicação dos componentes com o Keystone é realizada de maneira segura, como observado na **Figura 71**.

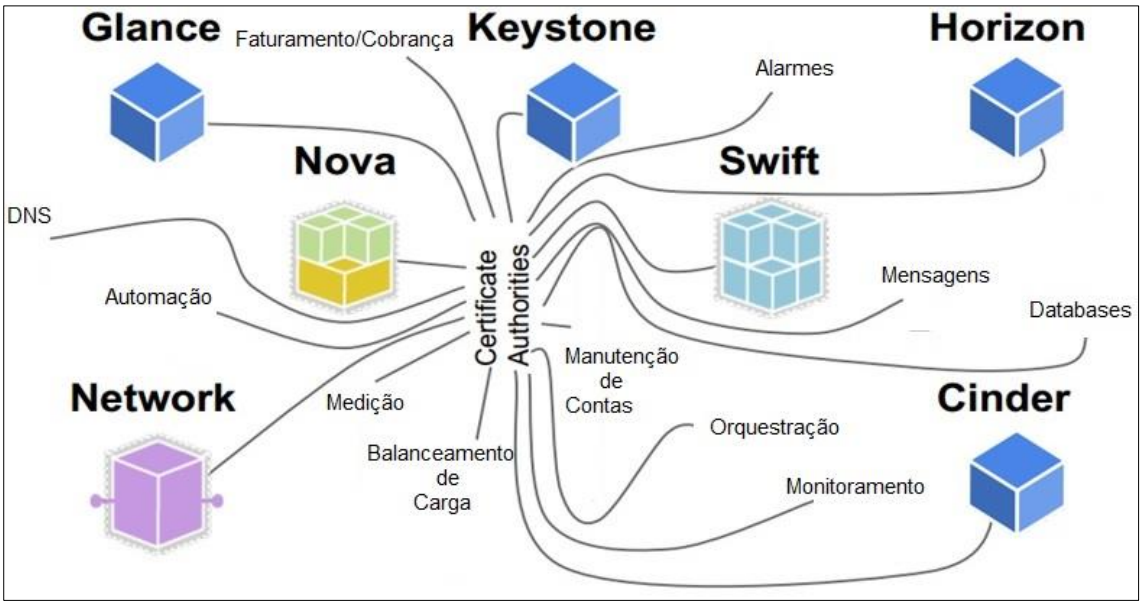
Figura 71 – Comunicação Cifrada com o Keystone



Fonte: Próprio Autor.

A **Figura 71** visa retratar o momento da captura dos dados da comunicação pelo Wireshark, apresentando os da token cifrados. Entretanto não é só a comunicação com o Keystone que utiliza SSL, segundo uma apresentação do OSSG (OSSG, 2013), toda comunicação realizada na plataforma é realizada de maneira segura, com os dados da *token* cifrados, a **Figura 72** mostra o mapa das comunicações SSL na plataforma.

Figura 72 – Comunicação com SSL Certificada



Fonte: (OSSG, 2013).

Portanto, a plataforma no *release* Grizzly disponibiliza ferramentas para que a comunicação interna seja realizada de forma segura, porém é necessário que estas opções sejam configuradas corretamente para que forneçam a confidencialidade necessária dos dados do usuário. Para o DevStack foi necessário alterar os *scripts* de configuração para utilizar SSL e Certificados de Autoridade, em uma nuvem de produção é necessário alterar o arquivo de configuração do Keystone (etc/keystone/keystone.confg), indicando o caminho dos certificados (como apontado na **Figura 68**), e gerar os certificados no Keystone (**Figura 69**).

4.5 ASPECTOS DE SEGURANÇA DO QEMU/KVM.

Esta subseção apresenta aspectos de segurança proporcionados pelo *hypervisor* Qemu e KVM para proporcionar o isolamento e o controle de recursos utilizados pelas VM's, como uma maneira de maximizar a segurança da plataforma. Para isto, é utilizada a ferramenta sVirt responsável pela parte de segurança do SELinux[56], que é suportada pelo OpenStack tanto no Qemu como no KVM. A subseção única **4.5.1** apresenta as capacidades de isolamento da ferramenta para realizar o isolamento de processamento e armazenamento nos recursos de *hardware* controlados pelo *hypervisor*.

4.5.1 ISOLAMENTO E GERENCIAMENTO DAS VM'S

A IBM disponibiliza um guia de segurança (IBM, 2012a) no qual é possível observar diversos aspectos de segurança elencados no comparativo (**3.2.5.1**). O guia de segurança do KVM é dividido em duas partes: segurança do *host* e segurança da VM.

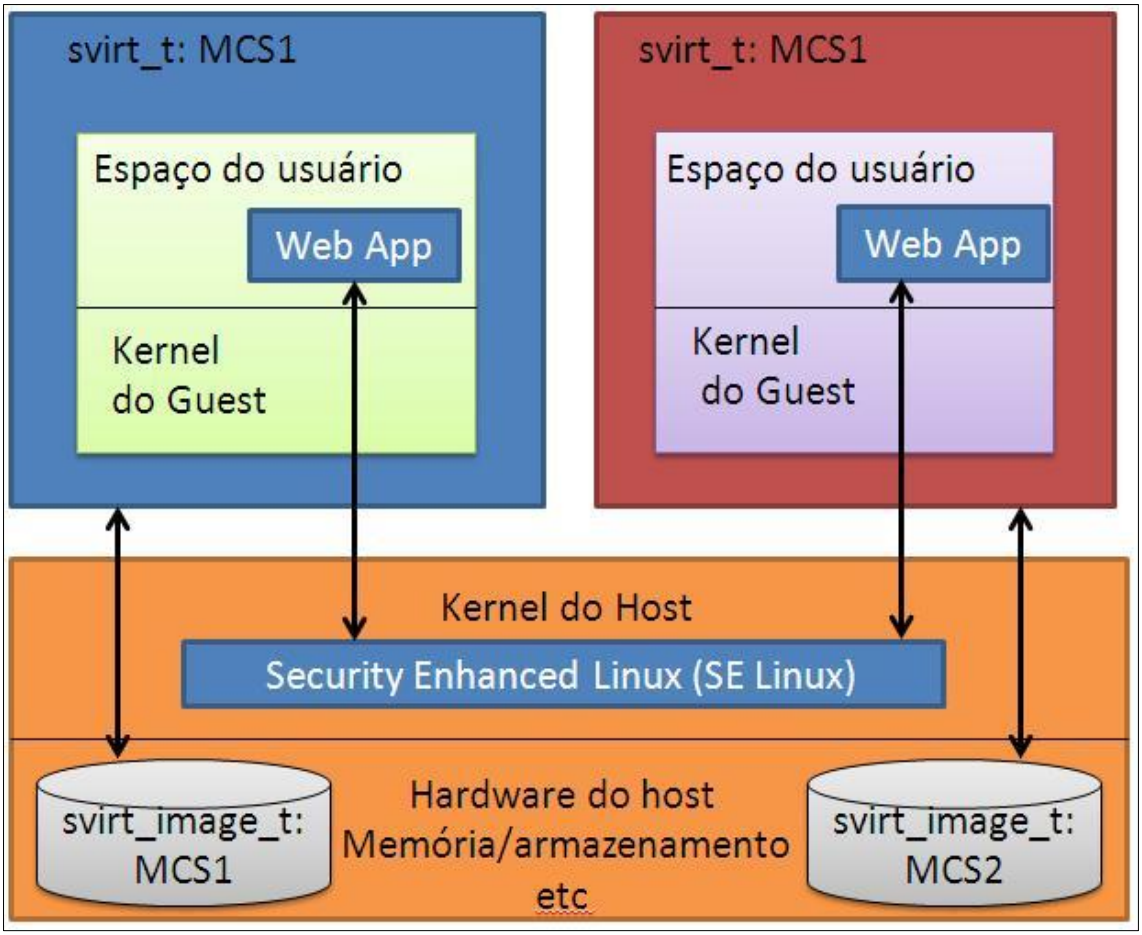
Na segurança do *host* o guia inicia comentando a necessidade de reforçar a segurança física da máquina, denominando isto com um conceito de Base de Computação Confiável ou *Trusted Computing Base* (TCB), que se trata da combinação de *hardware* e *software* em um sistema computacional que reforça uma política de segurança. Em seguida o guia explica o ponto do isolamento de rede do *host*, recomendando que para uma maior segurança é necessário especificar uma interface de rede para o *host* e outra separada para os SO's *guests*, ajudando a isolar o *host* dos SO's *guests* e prevenir possíveis

[56] http://selinuxproject.org/page/Main_Page

usuários maliciosos de posse de uma VM de baixo nível de segurança atacar o *host*.

A segurança das VM's é realizada utilizando o sVirt, que é um serviço incluído no *framework* SELinux para isolar VM's utilizando grupos de controle, prevenindo ataques DoS, fornecendo cifragem da imagem de disco e informações para auditoria (IBM, 2012a). O serviço do sVirt funciona definindo rótulos para os processos e imagens de disco de cada SO, estes rótulos isolam cada VM do restante. A **Figura 73** ilustra o isolamento dos processos das VM's fornecido pelos rótulos dinâmicos criados pelo sVirt.

Figura 73 - Isolamento de Processos Fornecido pelo sVirt.



Adaptado de: (IBM, 2013a).

Este isolamento ilustrado na **Figura 73** previne que usuários maliciosos de um dos SO's *guests* acesse o *host* e ataque os processos e recursos de outras VM's. O SELinux é uma implementação do MAC (*Mandatory Access Control*) que provê uma política de segurança sobre todos os processos e objetos do sistema baseando suas decisões em rótulos (IBM, 2013a).

Quando uma VM inicia é dada a ela um rótulo SELinux pelo sVirt, e de maneira semelhante, é dado ao arquivo de disco virtual desta VM é dado um rótulo correspondente. Desta maneira, com o sVirt atuando no nível do Kernel a VM fica "presa" somente ao disco com o rótulo semelhante ao seu. Por meio do Libvirt, e seu *shell* de comando virsh, é possível observar estes rótulos dinâmicos atribuídos as VM's criadas no OpenStack, a **Figura 74** apresenta o rótulo criado para a VM criada anteriormente "minhaVm" no exemplo de criação da VM.

Figura 74 – Rótulo das VM's Criado pelo sVirt

```
virsh # list
 Id Name                 State
----------------------------------
  1 instance-00000001    running
  2 instance-00000002    running

virsh # dominfo 1
Id:             1
Name:           instance-00000001
UUID:           6b773f73-2c7c-4974-8518-cd9f1cad054c
OS Type:        hvm
State:          running
CPU(s):         1
CPU time:       3.0s
Max memory:     65536 kB
Used memory:    65536 kB
Persistent:     yes
Autostart:      disable
Managed save:   no
Security model: apparmor
Security DOI:   0
Security label: libvirt-6b773f73-2c7c-4974-8518-cd9f1cad054c (enforcing)
```

VM "minhaVM" criada anteriormente

Rótulo

Fonte: Próprio Autor.

Desta maneira, com o sVirt atuando no nível do Kernel a VM fica "presa" somente ao disco com o rótulo semelhante ao seu. O sVirt também

disponibiliza grupos de controle para restringir um conjunto de tarefas (*cgroups*) a um conjunto de recursos, fazendo com que o ambiente se torne seguro a ataques de DoS e possibilidade de monitorar recursos, semelhante ao recurso de *quotas* fornecido pelo OpenStack. O *cgroup* é uma opção a nível de *Kerne* para impor contar e impor limites de recursos de *hardware* (CPU, memórioa, I/O, etc) para um grupo de processos unidos pelo mesmo critério (IBM, 2013a). Cada *guest* utiliza certa quantidade de recursos do sistema, incluindo poder de processamento, memória, disco, e outros recursos. Se um *guest* é malicioso ele deve utilizar a maior parte, senão todos os recursos disponíveis do sistema, com isto é necessário restringir as tarefas a uma quantidade determinada de recursos no servidor (IBM, 2013a).

Como o OpenStack efetua o papel do *hardware* do *host*, a operação equivalente para limitar os recursos disponíveis para uma instância são as quotas para projetos, e flavor's para as VM's. Com o virsh também é possível realizar diversas operações para gerenciar o ciclo de vida das VM's no Qemu, a **Figura 75** apresenta algumas destas operações com o virsh.

Figura 75 – Operações Básicas com as VM's Utilizando Virsh

```
bruno@bruno-W150HNM-W170HN:~$ sudo virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # connect qemu:///system

virsh # list
 Id Name                 State
----------------------------------
  1 instance-00000001    running

virsh # suspend 1
Domain 1 suspended

virsh # list
 Id Name                 State
----------------------------------
  1 instance-00000001    paused

virsh # resume 1
Domain 1 resumed

virsh # reboot 1
Domain 1 is being rebooted

virsh # shutdown 1
Domain 1 is being shutdown

virsh # destroy 1
Domain 1 destroyed

virsh # list
 Id Name                 State
----------------------------------

virsh #
```

Conecta ao *hypervisor* *qemu / kvm*

VM "minhaVM" criada anteriormente

Permite executar diversas operações com as VM's Guests do *hypervisor*

Fonte: Próprio Autor.

Como observado na **Figura 75**, o virsh permite executar tarefas de gerenciamento do ciclo de vida das VM's hospedadas no *hypervisor*, como pausar, despausar, reiniciar, desligar, destruir e iniciar uma VM. Para iniciar uma VM, operação que não é apresentada nesta figura, é necessário possuir um arquivo XML contendo as informações da mesma, para isto, é possível utilizar o comando *$virsh dumpxml IDdaMáquinaVirtual* e será criado um arquivo XML com as informações da VM, e então para criar uma VM basta utilizar o comando *$virsh start /caminho/arquivoXml.xml*. Entretanto, a criação

de uma VM pelo virsh não é indicada para o OpenStack, considerando que uma VM no OpenStack deve estar associada à um projeto, grupo de segurança, entre outras configurações para a VM próprias da plataforma.

O *virsh* é uma ferramenta padrão do Libvirt para a gerência de VM's, ou seja, não é necessário realizar configurações extras para habilitar, basta acessar a API do *virsh* pelo terminal para utilizar a ferramenta. O grande benefício de se gerenciar VM's pelo *virsh* é a agilidade e a simplicidade para executar operações sobre as VM's, enquanto que no Nova o processo é um pouco mais complexo.

4.6 CONSIDERAÇÕES PARCIAIS

Este capítulo teve como objetivo realizar testes práticos na plataforma OpenStack Grizzly utilizando como base os fundamentos de computação em nuvem, e os guias de segurança tanto de nuvem como específicos de virtualização. Para a realização destes testes foi utilizada uma instalação por *script* da plataforma, do DevStack, o que de certa forma limita a possibilidade de personalização da instalação, como por exemplo a utilização de VLANs para segmentar as VM's já que por padrão o DevStack configura as VM's em uma rede flat. Por outro lado, a utilização do DevStack poupou tempo na instalação e configuração da plataforma, o que foi um grande benefício para este trabalho, considerando que o objetivo não era a configuração da plataforma em si, mas avaliar o estado da segurança da plataforma e propor soluções.

A avaliação e comparação dos guias de segurança proporcionam uma visão geral da segurança e os principais aspectos de segurança que devem ser levados em consideração no ambiente de nuvem, com isto, e considerando a solução dos problemas propostos no Trabalho de Conclusão de Curso I (TCC-I), foi proposto em consenso com o orientador deste trabalho, avaliar os problemas de segurança relacionados à virtualização de recursos, observando que esta é uma das áreas que possui muitos problemas e poucas soluções. E então uma nova análise e comparação de guias de segurança específicos para

a virtualização foi realizada, com o objetivo de fornecer maior embasamento teórico para a realização dos testes.

Como o escopo era avaliar a segurança no processo de virtualização na plataforma, foi simulado o processo de criação de uma VM com o monitoramento da comunicação (via Wireshark) dos componentes ocorrendo em paralelo, o objetivo foi avaliar o estado da segurança da comunicação, desde a requisição da imagem pelo *hypervisor* no Nova, passando pelas autenticações no Keystone, até o Glance. Os testes revelaram algo que deve ser alvo de atenção durante a instalação do OpenStack, a configuração de segurança. A não configuração do Keystone para utilizar a comunicação via SSL, fez com que as credenciais do usuário fossem passadas na comunicação em texto às claras, sem qualquer segurança. Entretanto, após uma pesquisa na documentação da plataforma foi avaliado que o OpenStack disponibiliza os mecanismos de SSL e validação das chaves públicas por certificados, bastando somente configurá-los da maneira correta.

Ao final da seção, apresentou-se uma ferramenta utilizada pelo Libvirt para reforçar a segurança nos *hypervisors*, o sVirt. O sVirt possui mecanismos de segurança como grupos de controle e rótulos, que foi identificado nos testes para a VM criada “minhaVM” durante os testes. Além disto, o sVirt permite o gerenciamento direto das VM’s hospedadas no *hypervisor*, com diversos comandos básicos para administrar o ciclo de vida das máquinas.

5. CONSIDERAÇÕES FINAIS

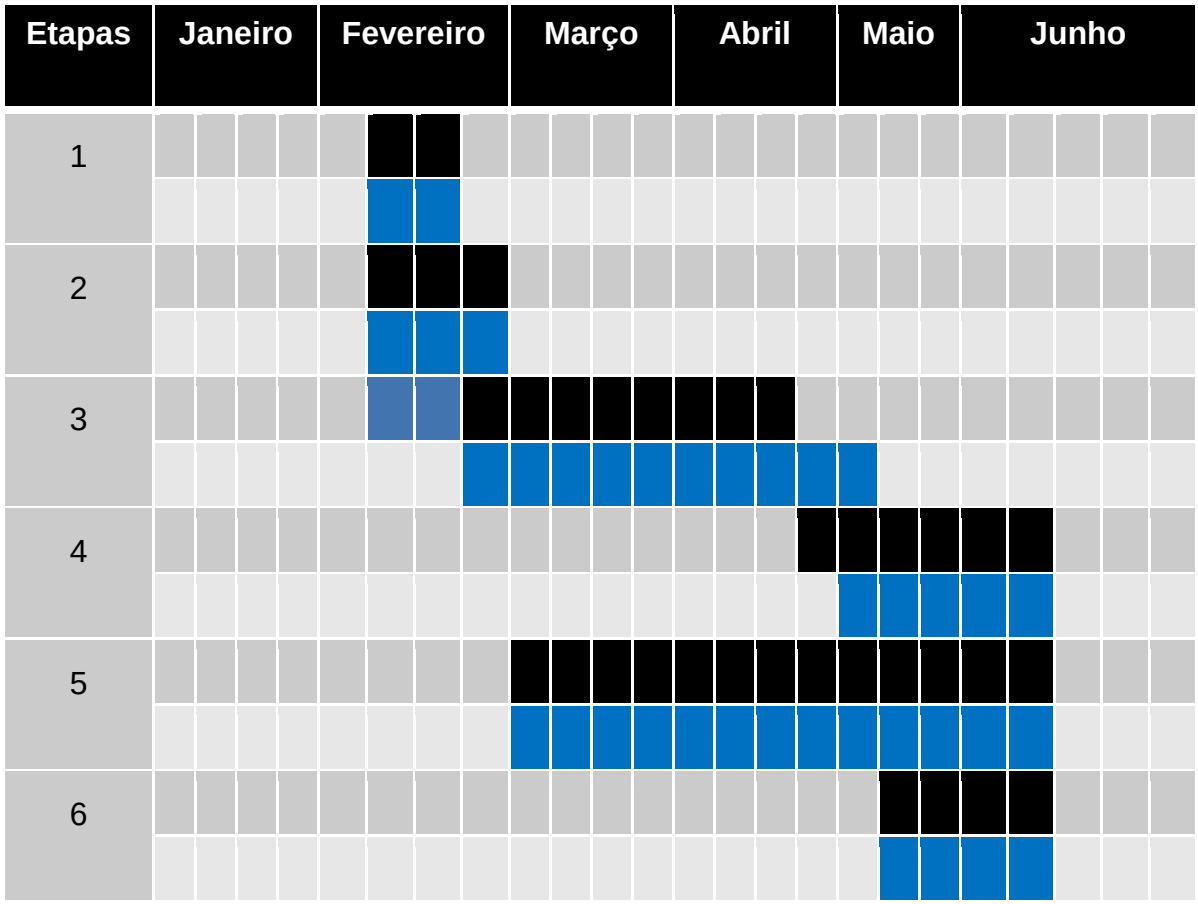
Observando o objetivo geral definido no plano de TCC, que define a contribuição esperada deste trabalho como uma proposta de melhoria e/ou adição de um recurso de segurança para a plataforma, afirma-se que o objetivo geral foi parcialmente atingido. No que diz respeito à melhorias de segurança, a contribuição foi um exemplo prático da necessidade de configurações adequadas de segurança (SSL/Certificados), verificada por meio da análise de tráfego. Diversas contribuições são feitas neste trabalho e possibilitaram maior entendimento do cenário da segurança na nuvem, como o comparativo dos guias de segurança em nuvem e de virtualização, a análise dos *hypervisors* utilizados na plataforma, o levantamento de *bugs* e vulnerabilidades dos *hypervisors*, entre outras. Estas contribuições possibilitam a compreensão da dimensão dos aspectos de segurança para a computação em nuvem, e principalmente fornecem embasamento para a realização de trabalhos mais específicos, o que é uma necessidade para a plataforma se o objetivo é deflagrar e solucionar vulnerabilidades. Atualmente existem diversas organizações trabalhando no desenvolvimento da plataforma, melhorando e corrigindo o OpenStack em diversas áreas, principalmente na área de segurança, com isto uma abordagem pouco específica dos problemas de segurança torna-se pouco efetiva, pois será rapidamente solucionada. Por outro lado, realizar um trabalho específico de segurança exige amplo conhecimento tanto dos conceitos de computação em nuvem como de segurança, para que seja possível aprofundar o conhecimento de maneira adequada e fornecer o embasamento para que isto seja possível foi a principal contribuição deste trabalho, além de reunir de forma padronizada vários documentos dispersos, propiciando uma análise de suas similaridades e diferenciais.

Visando acompanhar o desenvolvimento da plataforma, o trabalho foi desenvolvido inicialmente no TCC-I com pesquisas de vulnerabilidades sobre o *release* Folsom e remanescentes do Essex, posteriormente no TCC-II, foi alterada para o *release* Grizzly, sendo que em alguns casos o *release* Havana

é mencionado (o Havana foi liberado apenas 17/10/2013 o que não forneceu tempo hábil para a análise deste trabalho), e identificando como o desenvolvimento da plataforma é ágil, já ao final deste trabalho os desenvolvedores do OpenStack disponibilizaram o Havana e trabalham no desenvolvimento do *release* Icehouse. Com isto, o rápido desenvolvimento da plataforma foi o principal fator que impediu a resolução de um dos problemas propostos no TCC-I. Contudo, isto permitiu a possibilidade de se explorar outras melhorias na segurança da plataforma, como o aprimoramento do *script* de instalação do DevStack (*stack.sh*), realizando modificações de modo que o *script* habilite opções de segurança por padrão para a instalação do Openstack, como *tokens* PKI, cifragem do cabeçalho das imagens no Glance, protocolo padrão de comunicação HTTPS e utilização de certificados digitais para validar as chaves. Isto foi realizado porque o *script* de instalação sobrescreve os arquivos de configuração da plataforma a cada instalaçao, portanto foi necessário realizar esta modificação no *script*. Neste trabalho também foi explorada a ferramenta *virsh* da biblioteca de ferramentas de gerenciamento de *hypervisors* libvirt, que permite executar tarefas para o gerenciamento do ciclo de vida das VM's na plataforma, como também identificando outro recurso de segurança utilizado pela plataforma para garantir o isolamento das VM's, os rótulos de segurança providos pelo SELinux.

Os cronogramas deste trabalho tanto para o TCC-I quanto o para o TCC-II podem ser observados nas tabelas **Figura 76** e **Figura 77**. Durante a execução do TCC-I, a única modificação do que estava proposto a ser realizado no plano, foi a não realização de testes práticos durante a etapa de análise do Folsom, deixando esta etapa para ser realizada no TCC-II.

Figura 76 – Cronograma Previsto x Realizado TCC-I



Legenda: preto proposto (em cima), azul realizado (em baixo).

As etapas propostas no plano para o TCC-I foram:

1 - Formulação do plano;

2 - Levantamento e fichamento de referências;

3 - Consolidação das referências;

4 – Análise de Segurança;

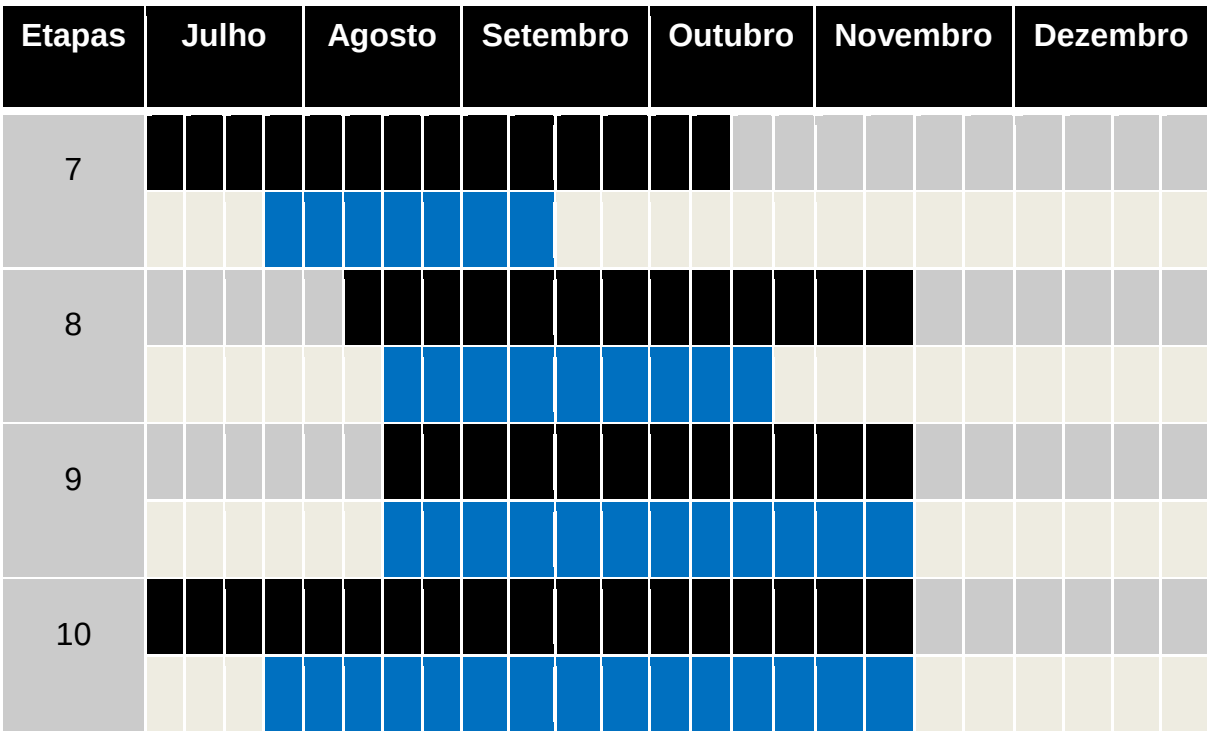
5 - Escrita do TCC-I; e

6 - Apresentação da proposta de melhoria ou adição de recurso de segurança.

A comparação entre os cronogramas permite observar que o trabalho esteve atrasado de acordo com o proposto no planejamento em aproximada mente duas semanas (final de abril e começo de maio). A etapa 3 (embasamento teórico) avançou duas semanas além do previsto, atrasando também o início da etapa 4 (parte de segurança do trabalho), o que resultou na impossibilidade de realizar testes práticos nas questões elencadas na análise.

Para o TCC-II, o objetivo definido no plano foi averiguar as questões elencadas no TCC-I e efetuar melhorias por meio de implementações ou adição de recursos de segurança. O cronograma proposto para o TCC-II pode ser observado na **Figura 77**.

Figura 77 – Cronograma Proposto x Realizado TCC-II



Legenda: preto proposto (em cima), azul realizado (em baixo).

As etapas propostas no plano para o TCC-II foram:

7 - Revisão de conceitos envolvidos na melhoria;

8 - Implementação. Desenvolvimento de melhorias;

9 - Testes. Consiste em verificar se a solução proposta cumpre seu objetivo, testando e avaliando os resultados; e

10 - Escrita do TCC-II.

A comparação entre proposto e realizado permite observar um atraso no início do trabalho em julho, que acarretou em um atraso no restante das atividades. Este atraso não gerou grandes problemas no decorrer da execução do trabalho, já que houve uma intensificação na realização das demais atividades do meio de outubro para o início de novembro acelerando a etapa de escrita das atividades realizadas durante o semestre.

5.1 PRINCIPAIS DIFICULDADES

Como todo sistema de computação em nuvem, a plataforma OpenStack também apresenta alto nível de complexidade, o primeiro passo para lidar com seus aspectos de segurança é a compreensão do funcionamento da plataforma, sua arquitetura e principalmente a relação entre seus componentes. O segundo aspecto importante, e objeto de estudo deste é a compreensão dos aspectos de segurança da plataforma, cuja maior dificuldade é identificar vulnerabilidades em um sistema complexo. A sumarização das principais dificuldades e como elas foram superadas são:

- **Compreensão do funcionamento plataforma**: a primeira grande dificuldade foi analisar separadamente a função de cada componente na nuvem, e posteriormente a relação entre eles, compreendendo assim o funcionamento da plataforma;
- **Como os aspectos de segurança se aplicam na plataforma**: tendo a compreensão da relação entre os componentes, foi necessário observar os principais pontos de segurança estabelecidos nas referências, para tentar identificar vulnerabilidades na plataforma OpenStack Grizzly;

- **Lidando na prática com a plataforma OpenStack**: nesta parte pode se listar os principais desafios como:
    - **Configuração inicial da plataforma:** definir os parâmetros de instalação como configuração de rede, divisão dos serviços nos nós, configuração dos *scripts* do DevStack. Como não existem guias bem documentados para a personalização dos *scripts* do DevStack, nesta parte foram realizadas diversas tentativas até atingir os objetivos estabelecidos;
    - **Utilização das API's:** a interface Web disponibilizada no Horizon é fácil e intuitiva, porém as API's do OpenStack proporcionam muito mais opções e agilidade na execução de operações na plataforma, e é uma tarefa que requer diversas tentativas e pesquisas em guias e manuais do OpenStack para aprender a utilizar as API's;
    - **Compreensão do código**: para observar onde e como são realizadas algumas operações importantes na plataforma (como a exclusão de objetos, armazenamento de senhas), foi necessário analisar o código da plataforma. Esta é uma tarefa que demanda paciência para analisar e compreender o código, que embora documentado é extenso e complexo;
    - **Como aplicar a correção na comunicação interna**: uma vez detectada a necessidade da configuração adequada, aplicar nos *scripts* de instalação do DevStack é uma tarefa que precisa de uma noção básica de *shell script*, para realizar as alterações fazendo com que a plataforma seja configurada de maneira adequada.

5.2 LISTA DE CONTRIBUIÇÕES

Como contribuições realizadas neste trabalho é possível listar:

- Reúne em um único documento vários materiais tanto introdutórios para computação em nuvem, como específicos de segurança que usualmente estão dispersos pela Internet. Além disso, fornecendo um primeiro referencial em português que possui valor no sentido de disponibilizar informações relevantes de segurança para nuvens computacionais para pessoas que não tem domínio da língua inglesa;

- Apresenta diversos comparativos de modo que seja possível obter abordagens distintas sobre um mesmo tema, avaliando, distinguindo e argumentando sobre as diferenças. Como exemplo podem ser citados os comparativos de guias de segurança e de segurança em virtualização, o comparativo de *hypervisors*, arquiteturas de referência, plataformas, e outros;

- Ajustes de segurança do *script* de instalação do DevStack, que realiza a instalação e configuração automática do OpenStack. Foram realizados ajustes de segurança, que viabilizam uma comunicação interna segura entre os componentes da plataforma com o uso de SSL, certificados digitais, e a cifragem dos metadados das imagens no Glance.

5.3 TRABALHOS FUTUROS

Sendo a computação em nuvem uma tecnologia recente e resultante da união de outras tecnologias como virtualização e componentes Web, a falta de material específico para entender como a segurança de alguns componentes se encaixam no contexto da segurança da nuvem é um problema constante. Este fato também incentiva a pesquisa sobre estes assuntos. Alguns temas

abordados durante o trabalho, como a classificação de vulnerabilidades relacionadas a virtualização, despertaram o interesse de pesquisa, estes temas são:

1. **Framework de Classificação de Vulnerabilidades de Virtualização**: como proposto em **3.4**, a necessidade de uma categorização mais especifica das vulnerabilidades provenientes das tecnologias de virtualização gerou o interesse na realização deste *framework*. A intenção é dar continuação ao trabalho feito por (GONZALEZ et. al., 2012), que propôs um *framework* para classificação de problemas de segurança em computação em nuvem, com uma parte destinada aos problemas de virtualização. O objetivo principal é complementar esta classificação, especificando detalhadamente os principais componentes envolvidos na virtualização e vulnerabilidades que são provenientes deles, de modo que seja possível, ao realizar uma análise de segurança de virtualização, categorizar as vulnerabilidades detalhadamente.

2. **Análise e Comparação de Segurança em SLA's:** realizar uma análise e comparação de SLA's específica para a parte de segurança, ou seja, verificar o posicionamento de cada SLA a respeito dos principais aspectos de segurança levantados e realizar um comparativo entre elas. A principal contribuição deste trabalho é proporcionar ao consumidor de nuvem um parâmetro dos principais fatores de segurança que se devem levar em consideração ao se utilizar algum "produto de nuvem", como IaaS, SaaS ou PaaS.

3. **Avaliação e Comparação de Desempenho de Hypervisors:** realizar uma avaliação dos hypervisors via execução de comandos das APIs. Por exemplo, as API's do OpenStack possuem o comando –timing, que permite avaliar o tempo de resposta de uma requisição (o GNU/Linux também proporciona o comando "time"). A finalidade é realizar uma avaliação de desempenho para a realização de atividades básicas com as VM's, rede, etc (a ser levantado), avaliando o desempenho por meio

do comparativo entre os principais *hypervisors* utilizados, como Xen, KVM e VMWare. Este comparativo também pode ser ampliado utilizando-se outras plataformas para realizar as mesmas tarefas, e assim, avaliar também o desempenho de outras plataformas de nuvem.

4. **Auditoria Específica de Segurança na Nuvem**: uma dúvida comum é se uma plataforma realmente realiza os aspectos de segurança definidos na SLA. O objetivo é realizar um levantamento se uma determinada plataforma disponibiliza ferramentas que permitam a realização de auditorias especificas de segurança, ou se permite a integração de alguma ferramenta que possam avaliar sua segurança. Para isto, é necessário mapear e compreender as vulnerabilidades provenientes do ambiente de nuvem, tanto como nos aspectos legais, políticos e organizacionais como nos aspectos técnicos. Outra possibilidade é trabalhar com uma vulnerabilidade específica, já que segurança da computação em nuvem compreende diversos fatores distintos, para possivelmente, desenvolver e integrar uma ferramenta de auditoria para esta vulnerabilidade específica.

Estas são apenas de algumas possibilidades de pesquisa observadas durante a execução deste trabalho, mas pelo fato da computação em nuvem ser um tema relativamente novo, ainda existem diversas áreas que necessitam de um esclarecimento maior. Destas possibilidades de pesquisas observadas, apenas a **3** não é relacionada a segurança, mas também apresenta sua parcela de importância, pois para uma nuvem que será utilizada em larga escala, o desempenho é um fator fundamental para a plataforma, e este comparativo talvez possa realizar uma contribuição nesta área.

REFERÊNCIAS

ABDUL, S. **Open Source Cloud Technologies**. Tutorial de Nuvens de Código Aberto. Aceito em: Symposium on Cloud Computing (SoCC). San Jose (CA), Estados Unidos. Outubro/2012.

ABOLAFYA, N. **Secure Documents Sharing System for Cloud.** Enviroments. Master of Science Thesis. KTH Industrial Engineering and Management. Stockholm, Sweden. 2012.

AMAZON (2013). **Amazon Simple Storage Service (Amazon S3)**. Publicado em: Visão geral do Amazon S3. Disponível em: http://aws.amazon.com/pt/s3/. Acesso em: 13/03/2013.

BADGER, L.; BERNSTEIN, D.; BOHN, R.; VAULX, F.; HOGAN, M.; MAO, J.; MESSINA, J.; MILLS, K.; SOKOL, A.; TONG, J.; WHITESIDE, F.; LEAF, D. **High-Priority Requirements to Further USG Agency Cloud Computing Adoption**. NIST Cloud Computing Program Information Technology Laboratory. 2011.

BARHAM, P.; DRAGOVIC, B.; FRASER, K.; HAND, S.; HARRIS, T.; HO, A.; NEUGEBAUER, R.; PRATT, I.; WARFIELD, A. **Xen and the art of virtualization**. In SOSP '03: Proceedings of the ninteenth ACM symposium on Operating system principles, p. 164-177, New York, NY, USA. ACM. 2003

BEHRENDT, M.; GLASNER, B.; KOPP, P.; DIECKMANN, R.; BREITER, G.; PAPPE, S.; KREGER, H.; ARSANJANI, A. **Introduction and Architecture Overview IBM Cloud Computing Reference Architecture 2.0**. Disponível em: https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc Acesso em: 31/03/2013.

BHANU, P. T. **Hypervisors, virtualização e nuvem: Aprofunde-se no hypervisor VMware ESX Server**. Cloud Computing. Biblioteca técnica IBM. Disponível em: http://www.ibm.com/developerworks/br/cloud/library/cl-hypervisorcompare-vmwareesx/ Acesso em: 10/10/2013.

BORGES, P. H.; NEUMAN, J. de S.; SCHULZE, B.; MURY, R. A. **Computação em Nuvem**. Instituto Federal de Educação, Ciência e Tecnologia do Maranhão. São Luís, Brasil. 2011.

BOGOSSIAN, F.; MACIEL, L., F.; SAMPAIO, R.; COUTO, R. **Computação em Nuvem**. Departamento de Eletrônica e Computação, Escola Politécnica, Universidade Federal do Rio de Janeiro (UFRJ). Disponível em: http://www.gta.ufrj.br/grad/10_1/nuvem/index.html Acesso em: 29/03/2013.

BROOKS, Tyson, T.; CAICEDO, C.; PARK, S. Joon. **Security Vulnerability Analysis in Virtualized Computing Enviroments**. International Journal of Intelligent Computing Research (IJICR), Volume 3, Issues 1 /2.

BURNS, C. (2012) **Top 10 Most Powerful IaaS Companies**. Publicado em: Computer World UK, The Voice of Management. Disponível em: http://www.computerworlduk.com/in-depth/cloud-computing/3351035/top-10-most-powerful-iaas-companies/. Acesso em 10/03/2013.

BUYYA R.;YEO C.S.;VENUGOPAL S.;BROBERG, J.; BRANDIC, I.**Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility**. Publicado em: Future Generation Computer Systems, 2009.

CABRAL, GONÇALO (2012). **Cloud, Afinal o que é e as suas vantagens**. Publicado em: Tutoriais Rumonet. Disponível em: http://artigos.rumonet.pt/alojamento-web/cloud-computing/cloud-afinal-o-que-e-e-as-suas-vantagens/ Acesso em: 05/03/2013

CHAPPEL, DAVID (2008). **A Short Introduction To Cloud Platforms**. Disponível em: http://www.davidchappell.com/CloudPlatforms--Chappell.pdf Acesso em: 05/04/2013.

CHAPPEL, DAVID (2009). **Introducing Windows Azure**. Disponível em: http://www.davidchappell.com/introducing_windows_azure_v1-chappell.pdf Acesso em: 13/04/2013.

CHAGANTI, PRABHAKAR (2011). **Computação em nuvem com o Amazon Web Services**. Publicado em: IBM Developers Works, Biblioteca técnica.

Disponível em: http://www.ibm.com/developerworks/br/cloud/library/ar-cloudaws1/. Acesso em: 13/03/2013.

CHMOUEL B. (2013). **Keystone and PKI Tokens Review**. Disponível em: http://blog.chmouel.com/2013/05/02/keystone-pki-tokens-overview/ Acesso em: 23/05/2013.

CIGOJ, P. **Security Aspects of OpenStack.** Josef Stefan International Postgraduate. Publicado em: School.International Journal of Cloud Computing, v. 1, n. 1, p. 23–36, 2011. Lujubljana, Eslovênia, 2012.

CITRIX (2013). **Citrix CloudStack 3 Brings the Power of Amazon-Style Clouds to Customers of All Sizes.** Disponível em: http://www.citrix.com/news/announcements/feb-2012/citrix-cloudstack-3-brings-the-power-of-amazon-style-clouds-to-customers-of-all-sizes.html. Acesso em 08/04/2013.

CLOUSTACK (2013). **Understanding Apache CloudStack**. Disponível em: http://incubator.apache.org/cloudstack/software.html Acessado em: 14/03/2013.

CLOUDTIMES (2011). **Top PaaS, SaaS and IaaS Cloud Companies by CloudTimes**. Disponível em: http://cloudtimes.org/2011/11/30/top-paas-saas-and-iaas-cloud-companies-by-cloudtimes/ Acesso em: 14/04/2013.

CRM FORECAST (2010). **Salesforce.com Review**. Disponível em: http://www.crmforecast.com/salesforcecom.htm Acesso em 14/04/2013.

CSA (2011). **Security Guidance for Critical Areas of Focus in Cloud Computing V3.0.** Tech. rep., Cloud Security Alliance. Disponível em: http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf Acesso em: 20/01/2013.

CVE MITRE (2013). **Common Vulnerabilities and Exposures List.** Disponível em: http://www.cve.mitre.org/cve/ Acesso em 11/10/2013.

DANG, Q. **Recommendation for Applications Using Approved Hash Algorithms**. NIST Special Publication 800-107. Computer Security Division.

2012. Disponível em: http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf Acesso em: 01/01/2013.

DEVELOPERS GOOGLE (2012). **O que é o Google App Engine?**. Disponível em: https://developers.google.com/appengine/docs/whatisgoogleappengine? Acesso em: 12/04/2013.

DOCS OPENSTACK (2013). **Chapter 13. Cells.** Disponível em: http://docs.openstack.org/trunk/openstack-compute/admin/content/ch_cells.html

Acesso em: 08/09/2013.

DRUPAL (2013) **Comunidade Drupal. O que é o Drupal?**. Disponível em: http://www.drupal.nomundo.net/node/18 Acesso em: 14/04/2013.

FINLEY, KLINT (2012). **Data-as-a-Service: When Cloud and Big Data Converge (Trends 2012)**. Publicado em: Services Angle. Disponível em: http://servicesangle.com/blog/2012/03/27/data-as-a-service-when-cloud-and-big-data-converge-trends-2012/

GARFINKEL, S. L. **An evaluation of amazon's grid computing services: EC2, S3, and SQS** Center for. **Anais**...2007. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.172.2239. Acesso em: 10 abr. 2013

GARG, S. K.; VERSTEEG, S.; BUYYA, R. A Framework For Ranking of Cloud Computing Services. **Future Generation Computer Systems**, v. 29, n. 4, p. 1012-1023, jun. 2013.

GARTNER INC. (2012). **Gartner: Growth in cloud computing to shape 2013 security trends**. Disponível em: http://www.networkworld.com/news/2012/120612-gartner-cloud-security-264873.html . Acesso em: 21/01/2013.

GOLDEN, B. (2011). **SLAs: Todo o Cuidado é Pouco em Computação em Nuvem. Será?**. Publicado em: Portal Computer World. Disponível em: http://computerworld.uol.com.br/tecnologia/2011/11/14/slas-todo-o-cuidado-e-pouco-em-computacao-em-nuvem-sera/ . Acesso em: 29/03/2013.

GONZALEZ, N.; MIERS, C.; REDÍGOLO, F.; SIMPLÍCIO, M.; CARVALHO, T.; NÄSLUND, M.; POURZANDI, M.. A **A Taxonomy Model For Cloud Computing Services***.* Laboratory of Computer Architecture and Networks,University of São Paulo, Science and and Technology Publications*. 2011.*

*GONZALEZ, N.; MIERS, C.; REDÍGOLO, F.; SIMPLÍCIO, M.; CARVALHO, T.; NÄSLUND, M.; POURZANDI, M.. **A Quantitative Analysis of Current Security Concerns and Solutions for Cloud Computing**. Escola Politécnica da Universidade de São Paulo, Ericsson Research Stockholm, Ville Mont-Royal. 2012.*

HABIB, I. **Virtualization with KVM**. Linux Journal, v.2008 n.166, p.8, Fevereiro 2008.

HEADS IN THE CLOUD (2010). **IaaS, Paas, SaaS, BaaS, DaaS, EaaS – I gottaa get the aa fixed on my keyboaard**. Publicado em: Wordpress. Disponível em: http://headsinthecloud.wordpress.com/2010/10/13/iaas-paas-saas-baas%C2%A0daas/ Acesso em: 05/03/2013

IBM. (2013). **IBM and HP Virtualization**. Disponível em: http://www.ibm.com/developerworks/aix/library/au-aixhpvirtualization/ Acesso em: 27/08/2013.

IBM. (2012a) **Kernel Virtual Machine (KVM) KVM security.** Disponível em: http://pic.dhe.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatsecurity_pdf.pdf Acesso em: 28/09/2013.

IBRAHIM, A.; HAMLYN-HARRIS, J.; GRUNDY, J.; ALMORSY, M. **CloudSec: A Security Monitoring Appliance for Virtual Machines in the IaaS Cloud Model***.* Proceedings of the 5th International Conference on Network and System Security (NSS 2011), Milan, Italy. IEEE 2011.

INFOQ. (2012). **Google App Engine traz suporte inicial a Java 7.** Disponível em: http://www.infoq.com/br/news/2012/12/google-app-engine-java7 Acesso em: 12/04/2013.

ITL NIST (2013). **An Update On Cryptographic Standards, Guidelines, and Testing Requirements.** Disponível em: http://www.itl.nist.gov/lab/bulletns/bltnmay06.htm Acesso em: 17/10/2013.

JAEGER. 2007. Virtual Security Machine. Introduction Computer and Network Security. PennState University. Disponível em: http://www.cse.psu.edu/~tjaeger/cse497b-s07/slides/cse497b-lecture-26-virtualmachine.pdf Acesso em: 26/09/2013

KISIELEWICZ. (2012). **Keystone/Service-Isolation-And-Roles-Delegation**. Disponível em: https://wiki.openstack.org/wiki/Keystone/Service-Isolation-And-Roles-Delegation. Acessom em: 23/05/2013.

KOSLOVSKI, G. **Dynamically provisioned virtual infrastructures: specification, allocation and execution**. Tese de doutorado, ENS-Lyon, Université de Lyon. Julho, 2011.

KOSLOVSKI, G.; PRIMET, P.; CHARÃO, A. **Virtual Resources and Interconnection Networks Description Language**. In the Second International Conference on Networks for Grid Applications (GridNets 2008), Lecture Notes of the institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Beijing, China, Springer Berling Heidelberg. 2008.

KUMAR, T. (2012). **Drupal: A Market Leader Due Continuous and Effective Growth**. Disponível em: http://www.valuebound.com/blog/drupal-market-leader-due-continous-and-effective-growth Acesso em: 15/04/2013.

LASZEWSKI, G.; DIAZ, J.; WANG, F.; FOX, G. **Comparison of Multiple Cloud Frameworks**. In 5th International Conference on Cloud Computing. IEEE. 2012.

LAUNCHPAD (2013). **OpenStack**. Disponível em: https://launchpad.net/openstack. Acesso em: 16/05/2013.

LIBVIRT. (2013). **The Virtualization API**. Disponível em: http://libvirt.org/ Acesso em: 28/09/2013.

LINDER, M. S. (2007). **Linguagens de Programação**. Disponível em: http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula5.pdf Acesso em: 20/04/2013.

LINUX INSIGHT (2008). **Finally User-friendly Virtualization for Linux**. Disponível em: http://www.linuxinsight.com/finally-user-friendly-virtualization-for-linux.html. Acesso em: 25/09/2013.

LIU, F.; TONG, J.; MAO, J.; BOHN, R.; MESSINA, J.;BADGER, L.; LEAF, D. **NIST Cloud Computing Reference Architecture**. NIST Special Publication, v. 500, p. 292, 2011.

LUÍS, Antonio Jesus Mendes da Silva. **Cloud Computing Segurança e Privacidade da Informação na Nuvem**. Instituto Superior de Estudos Financeiros e Fiscais. Portugal. Julho, 2012.

MARCON, A.; LAUREANO, M.; SANTIN, A.; MAZIERO, C. **Aspectos de Segurança e Privacidade em Ambientes de Computação em Nuvem.** Programa de Pós-Graduação em Informática - PPGIa, Pontifícia Universidade Católica do Paraná - PUCPR, Instituto Federal de Educação, Ciência e Tecnologia do Paraná - IFPR. Paraná. 2010.

MEIERS, J. (2011). **Medição e Faturamento na Nuvem**, **Métricas de Faturamento Para Recursos de Cálculo em Nuvem**. Biblioteca técnica IBM. Disponível em: http://www.ibm.com/developerworks/br/cloud/library/cl-cloudmetering/ Acesso em: 18/03/2013.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. National Institute of Standards and Technology (NIST), Information Technology Laboratory. Gaithersburg, Maryland – USA. 2011.

MIRANTIS (2013). **Swift Encrypt.** Disponível em: https://github.com/Mirantis/swift-encrypt Acesso em: 01/01/2013.

MSDN (2010). **Sobre o Windows Azure**. Disponível em: http://msdn.microsoft.com/pt-br/library/dd179442.aspx Acesso em: 14/04/2013.

OCCI-WG (2013). **An Open Community Leading Cloud Standarts**. Disponível em: http://occi-wg.org/ Acesso em: 18/04/2013.

OPENCLOUDMANIFESTO (2009). **Open Cloud Manifesto. Dedicated to the Belief that Cloud Should be Open**. Disponível em: http://www.opencloudmanifesto.org/Open%20Cloud%20Manifesto.pdf Acesso em: 23/05/2013.

OPENNEBULA. (2013)**. About the OpenNebula.org Project**. Disponível em: http://opennebula.org/about:about. Acesso em: 15/03/2013.

OPENSTACK (2013). **What Is OpenStack?**. Disponível em: http://www.openstack.org/#tabWhatIs . Acesso em: 21/01/2013.

OPENSTACK (2013a). **OpenStack Cinder Documentation.** Disponível em: https://wiki.openstack.org/wiki/Cinder . Acesso em: 10/05/2013.

OPENSTACK (2013b). **OpenStack Nova's Documentation.** Disponível em:

http://docs.openstack.org/developer/nova/. Acesso em: 10/05/2013.

OPENSTACK (2013c). **OpenStack Glance Documentation.** Disponível em:

http://docs.openstack.org/developer/glance/ .Acesso em: 10/05/2013.

OPENSTACK (2013d). **OpenStack Swift Documentation.** Disponível em:

http://docs.openstack.org/developer/swift/. Acesso em: 10/05/2013.

OPENSTACK (2013e). **Horizon: The OpenStack Dashboard Project.** Disponível em: http://docs.openstack.org/developer/horizon/. Acesso em: 10/05/2013.

OPENSTACK (2013f). **OpenStack Networking Quantum.** Disponível em:

https://wiki.openstack.org/wiki/Quantum. Acesso em: 10/05/2013.

OPENSTACK (2013g). **Keystone: The OpenStack Identity Service.** Disponível em: http://docs.openstack.org/developer/keystone/. Acesso em: 10/05/2013.

OPENSTACK HYPERVISOR. 2013. Hypervisor Support Matrix. Disponível em: https://wiki.openstack.org/wiki/HypervisorSupportMatrix. Acesso em: 26/09/2013

ORACLE. (2012) . **Cloud Reference Architecture**. Disponível em: http://www.oracle.com/technetwork/topics/entarch/oracle-wp-cloud-ref-arch-1883533.pdf. Acesso em: 31/03/2013.

ORLANDO, DAN. (2011). **Modelos de serviço de computação em nuvem, Parte 1: Infraestrutura como serviço**. Disponível em: http://www.ibm.com/developerworks/br/cloud/library/cl-cloudservices1iaas/ Acesso em: 22/03/2013.

PANETTIERI, J. (2013). **OpenStack vs CloudStack: The Latest Score**. Disponível em: http://talkincloud.com/cloud-computing-management/openstack-vs-cloudstack-latest-score Acesso em: 20/04/2013.

PROTOCOLOTI. **SaaS, PaaS, IaaS - As Camadas do Cloud Computing** .Disponível em: http://protocoloti.blogspot.com.br/2012/03/saas-paas-e-iaas-as-camadas-do-cloud.html. Acesso em: 10/02/2013.

PEPPLE, K. (2012). **OpenStack Folsom Architecture**. Disponível em: http://ken.pepple.info/openstack/2012/09/25/openstack-folsom-architecture/ Acesso em: 25/04/2013.

POPEK; GOLDBERG, R. **Formal Requirements for Virtualizable Third Generation Architectures**. In SOSP '73: Proceedings of the fourth ACM symposium on Operating system principles, page 121, New York, NY, USA, ACM. 1973.

QIAO, M. J., BUSQUET R., VASQUEZ, Y. **OpenStack**. Disponível em: http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2012_2/openstack/index.html Acesso em: 27/04/2013.

RACKSPACE. (2013). **Architecting VMware vSphere For OpenStack**. Disponível em: http://www.rackspace.com/blog/architecting-vmware-vsphere-for-openstack/ Acesso em: 28/09/2013.

RITA DE C. C. DE CASTRO; LUIZA DOMINGOS; GLAUCIVÂNIA LUZ; CLAUDIO PEREIRA; MARCELO GOMES. ***Gestão de Vulnerabilidades em Cloud Computing: Um Cenário da Nuvem Pública***. Universidade Estadual do Ceará (UECE). 2012.

REVIEW (2013). **OpenStack**. Disponível em: https://review.openstack.org Acesso em: 17/05/2013.

ROMAN, R.; RODEL, F.; PHUA EU, G.; ZHOU, J. **Analysis of Security Components in Cloud Storage Systems.** Institute for Infocomm Research, Data Storage Institute. Singapore. 2012.

RYCK, P; DESMET, L; PHILIPPAERTS, P; PIESSENS, F. **Benefits, Risks and Recomendations for Information Security**. European Network and Information Security Agency (ENISA) , 2011.

SALESFORCE (2013). **An Introduction to Custom Application Development in the Cloud**. Disponível em: http://www.salesforce.com/us/developer/docs/fundamentals/salesforce_creating_on_demand_apps.pdf Acesso em: 14/04/2013.

SALISBURY, BRENT. **OpenStack Multi-Node DevStack Nova Network Tutorial.** Disponível em: http://networkstatic.net/openstack-multi-node-devstack-nova-network-tutorial Acesso em: 25/10/2013.

SECURITY GUIDE OPENSTACK. (2013). **OpenStack Security Guide Havana**. Disponível em: http://docs.openstack.org/security-guide Acesso em: 01/11/2013.

SLIPETSKYY, R.. **Security Issues In OpenStack**. Master thesis in Security and Mobile Computing, Departament of Telematics, Norwegian University of Science and Technology (NUST). Junho de 2011.

SOFTWARE FUNDAMENTALS (2013). **Defect Severity Bugs.** Disponível em: http://softwaretestingfundamentals.com/defect-severity/. Acesso em: 10/10/2013.

SOLTER, N. **OpenSolaris em Conjunto. A HA depende do Open HA Cluster**. Publicado em: Linux Magazine. Fevereiro de 2010.

UTEST (2013). **Tester Help Topics. Bugs Classification.** Disponível em: http://help.utest.com/testers/participation/submit-reports/classifying-bugs. Acesso em: 10/10/2013.

TOSSULINO, G. (2008). **Drupal é solução robusta e flexível em CMS.** Disponível em: http://webinsider.uol.com.br/2008/05/26/drupal-e-solucao-robusta-e-flexivel-em-cms/ Acesso em: 14/04/2013.

TRENDMICRO (2013). **Cost-effective Virtualization & Cloud Security.** Disponível em: http://www.trendmicro.com/cloud-content/us/pdfs/business/sb_deep-security.pdf Acesso em: 10/09/2013.

VMWare (2013). **Virtualization and Risk: Key Security Considerations for Your Enterprise Architecture**. Disponível em: http://www.vmware.com/files/pdf/partners/security/mcafee-key-security-ent-arch-wp.pdf. Acesso em 10/09/2013.

VIMAL V. (2009). **Virtualization Vulnerabilities and Threats: A Solution White Paper**. Red Cannon Security Inc. Disponível em: http://www.redcannon.com/vDefense/VM_security_wp.pdf Acesso em 17/10/2013.

WEN, X.; GU, G.; LI, Q.; GAO, Y.; ZHANG, X. **Comparison of Open-Source Cloud Management Platforms: OpenStack and OpenNebula**. In 9th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD '12, pages 2457- 2461. 2012.

WILLIAMS B.; CROSS, T. **Virtualization System Security**. IBM Disponível em: http://blogs.iss.net/archive/papers/VirtualizationSecurity.pdf Acesso em: 17/10/2013.

WIRESHARK.(2013). **Wireshark – What's on Your Network?**. Disponível em: http://www.wireshark.org/. Acesso em: 27/10/2013.

APÊNDICE A

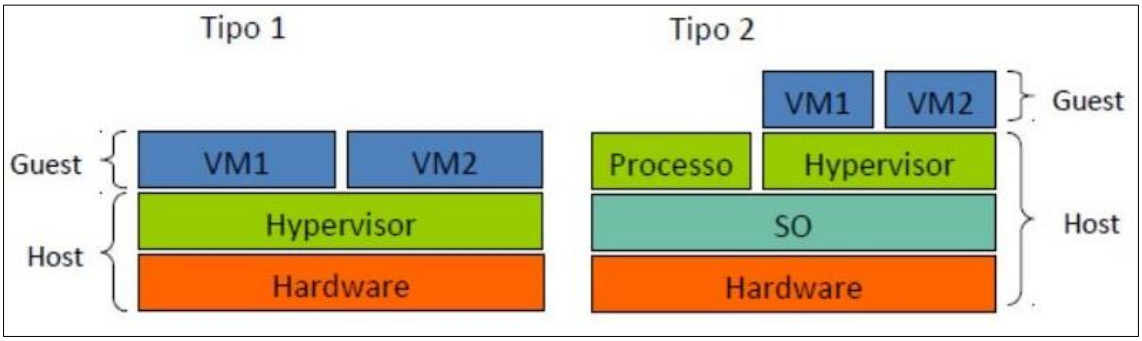
TIPOS DE HYPERVISORS

Há dois tipos de *hypervisors*:

- *Hypervisor* No metal (Baremetal) Tipo 1; e
- *Hypervisor* Hospedado Tipo 2.

Os *hypervisors* tipo 1 funcionam diretamente no hardware do sistema. Os *hypervisors* tipo 2 funcionam em um sistema operacional host que fornece serviços de virtualização, como suporte de dispositivo de entrada e saída e gerenciamento de memória. A **Figura 78** mostra as diferenças entre os hypervisores tipo 1 e tipo 2.

Figura 78 – Tipos de Hypervisors



Fonte: (MIERS, 2013).

APÊNDICE B

CIFRAGEM DE DADOS

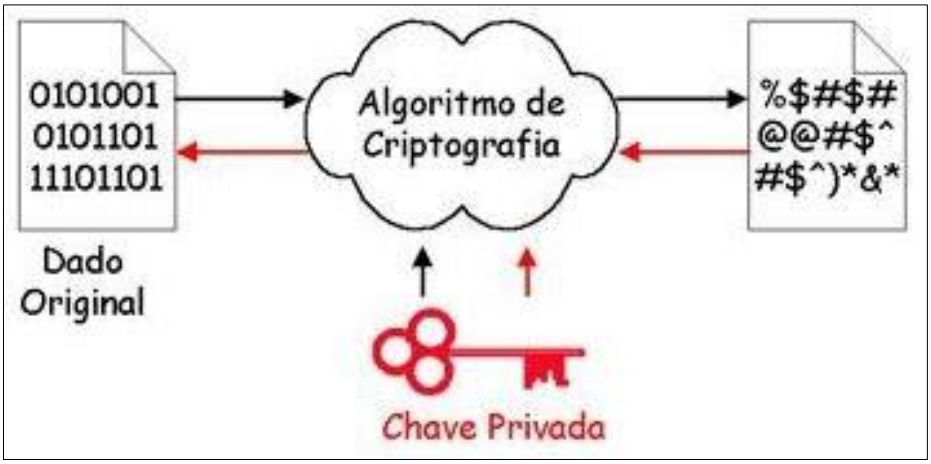
A cifragem é um importante recurso utilizado para garantir a segurança tanto dos dados, canais de comunicação (protocolos SSL, TLS) e imagens das VM's armazenadas em um servidor (JANSEN e GRANCE, 2011). Os três guias do comparativo citam o recurso da cifragem como uma maneira de se garantir a privacidade e a confiança no ambiente virtual, considerando isto é importante reforçar este conceito nesta subseção.

O objetivo das técnicas de cifragem de dados é garantir a confidencialidade e a integridade e existem dois problemas envolvidos neste problema (ABOLAFYA, 2012):

- **Cifragem dos dados**. As tecnologias de cifragem para garantir a confidencidalide e integridade dos dados; e
- **Decifragem**. Os dados seguros devem ser cifrados de uma maneira flexível, moduloar e segura o bastante de modo que forneca usabilidade.

As duas técnicas básicas de cifragem de dados são: a de chaves simétricas e de chaves assimétricas. A simétrica é o tipo mais simples de cifragem, já que tanto o emissor quanto o receptor da mensagem possuem a mesma chave, ou seja, a mesma chave é usada tanto na codificação quanto na decodificação (ABOLAFYA, 2012). Para ser realizada, basta que o emissor, antes de enviar a mensagem cifrada, envie a chave privada que será utilizada para decifrá-la. A **Figura 79** apresenta o esquema de chaves simétricas.

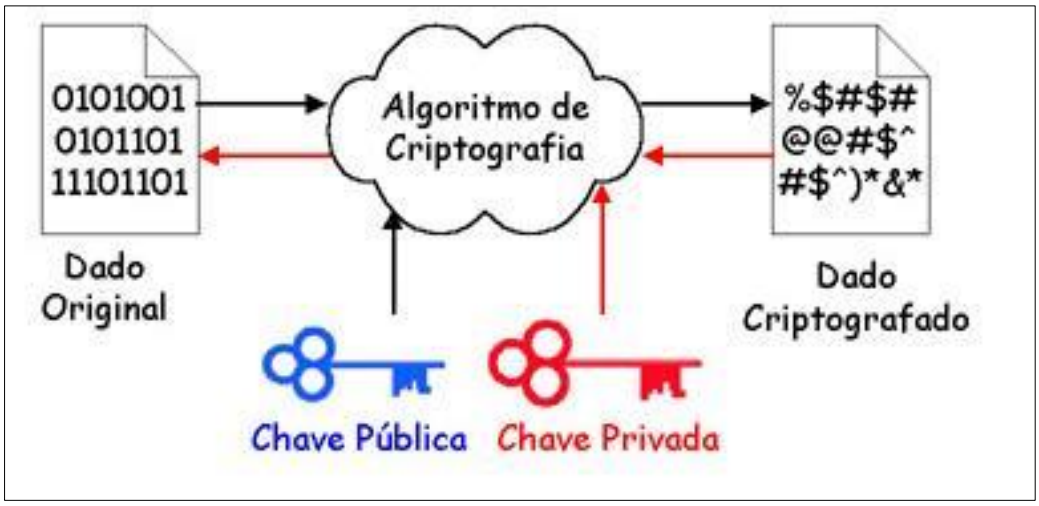
Figura 79 – Chaves Simétricas



Fonte: (PINHEIRO, 2009).

A **Figura 79** ilustra o exemplo de utilização da cifragem de chaves simétricas, em que o texto em claro é cifrado com a chave privada e somente é decifrado com a mesma chave privada utilizada. Na cifragem de chaves assimétricas. Diferentemente do método de Chave Simétrica, esse tipo utiliza duas chaves, uma pública e uma privada. O sistema funciona da forma que alguém cria uma chave e envia essa chave a quem quiser mandar informações a ela, essa é a chamada chave pública (ABOLAFYA, 2012). Com ela é feita a codificação da mensagem. Para decodificação será necessário utilizar outra chave que deve ser criada, a chave privada – que é secreta. A **Figura 80** ilustra a utilização das chaves assimétricas.

Figura 80 - Chaves Assimétricas



Fonte: (PINHEIRO, 2009).

Na **Figura 80** o emissor utiliza a chave pública para cifrar o dado original e envia, quando o receptor receber a mensagem utiliza a chave privada para decifrar a mensagem. Este esquema garante a confidencialidade dos dados e também autenticação, pois somente o proprietário da chave privada será capaz de decifrar a mensagem. Nestes algoritmos um dado que é criptografado com uma chave só poderá ser decifrado com a utilização da outra chave e vice-versa (PINHEIRO, 2009). Exemplos de implementações deste tipo de algoritmo: RSA, DSS (*Digital Signature Standard*).