UNIVERSIDADE FEDERAL DO PAMPA

KIM MANSILHA LABRÊA

SERVIÇOS DE SEGURANÇA SOB DEMANDA: UM ESTUDO DE CASO

ALEGRETE
2015

KIM MANSILHA LABRÊA

SERVIÇOS DE SEGURANÇA SOB DEMANDA: UM ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Diego Luís Kreutz

ALEGRETE
2015

KIM MANSILHA LABRÊA

SERVIÇOS DE SEGURANÇA SOB DEMANDA: UM ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 23/02/2015.

Banca examinadora:

Prof. Me. Diego Luís Kreutz
Orientador - (UNIPAMPA)

Prof. Dr. Fabio Natanael Kepler
(UNIPAMPA)

Prof. Me. Sam da Silva Devincenzi
(UNIPAMPA)

AGRADECIMENTOS

Durante os sete anos que convivi no ambiente acadêmico, muitas pessoas chegaram e foram embora, mas todas elas colaboraram um pouco com meu amadurecimento, gostaria de agradecer a todos vocês por fazerem parte dessa fase da minha vida. Eu dedico esse trabalho para minha família que sempre me incentivou, mesmo eu não merecendo. Minha mãe que com aquele amor incondicional aturou meu mau humor, meu pai que mesmo não sabendo nada sobre o assunto, sempre perguntava se tinha alguma coisa que pudesse fazer para me ajudar, e minha irmã que segurou minha peteca quando nada mais fazia sentido. Meu orientador Diego Kreutz pela paciência sem limites. Sem eles, certamente eu não teria me formado.

Gostaria também de dar um agradecimento especial para as pessoas que me ajudaram no desenvolvimento desse trabalho, seja com ideias ou revisões. Um obrigado para o Vitor por me receber na base secreta N° 1, ao Igor por me receber na base secreta N° 2, vocês tornaram essa minha última semana de edição muito mais agradável. Ao Marcos e ao Cezar Luiz Krause por me ajudarem no desenvolvimento inicial do servidor, ao Douglas Lautert, Marcelo Aymone, Matheus Serpa, Luana Alonso, Fabiano Castoldi, Bruno Miranda, Bruna, Bruno Felice, Kauê Vargas, Marcell dos Santos, Carla Unamuzaga, Lincoln Saraiva, Maicon Isoton, Willian da Silva, Wagner Reck e Luma Mulinari que colaboraram de diversas maneiras, seja revisando ou codificando. Ao meus colegas que iniciaram o curso comigo e fizeram inúmeros churrascos e jogatinas, Iuri Londero, Marcelo Neimayer, Toni Montenegro, Lucas Arbiza, Jarbas Gonçalves. Ao meu pai adotivo Luis Guilherme por ser um amigão e ter me acompanhado nas minhas ideias mais malucas, minha mãe adotiva Débora Pontes por me levar para beber nas segundas, e a minha maninha Larissa Hemman que me ajudou quando eu mais precisei. Me perdoem os que eu esqueci os nomes. Esse trabalho é para todos vocês.

“A violência é o último refúgio do incompetente”

Isaac Asimov

RESUMO

Manter sistemas Web seguros é um dos grandes desafios encontrados nos dias de hoje. Empresas precisam investir cada vez mais em segurança a fim de minimizar os riscos de incidentes de segurança e perdas para o negócio. Uma das formas de investir o mínimo necessário, sem correr riscos, é utilizar serviços terceirizados. Neste sentido, este trabalho tem como objetivo principal contribuir no preenchimento de uma lacuna latente, i.e., oferecer serviços de segurança de sistemas sob demanda. Para atingir o objetivo o trabalho propõe uma arquitetura de sistema capaz de oferecer serviços de segurança, mais especificamente scanners de vulnerabilidade de sistemas Web, de forma simplificada e elástica, ou seja, de acordo com as demandas dos clientes. A solução para este problema parte de uma arquitetura integrada e elástica para receber as requisições dos clientes, ativar os scanners de vulnerabilidades para detectar fragilidades dos sistemas Web alvo, filtrar e armazenar os resultados para posterior envio ao cliente. A elasticidade do sistema é baseada em computação em nuvem, mais precisamente máquinas virtuais que podem ser alocadas dinamicamente em provedores de nuvem como a Amazon, Google e Rackspace. Com isso, o sistema é capaz de comportar números variados de scanners de vulnerabilidades e múltiplos clientes sem comprometer a qualidade e os resultados finais. Para avaliação e discussão da arquitetura proposta foi implementado um protótipo denominado de SafeWeb. O protótipo desenvolvido foi avaliado através de utilização de dois scanners de vulnerabilidades e um conjunto de cinco sistemas vulneráveis. Além disso, o trabalho discute alguns dos principais desafios encontrados e trabalhos futuros, como a medição precisa dos recursos consumidos por cada scanner, com o intuito de equacionar o custo associado a cada ferramenta no valor final do serviço e otimizar o uso de recursos computacionais. Outro trabalho futuro consiste em criar um modelo financeiro auto ajustável, ou seja, que automaticamente reduz ou aumenta o custo do serviço de acordo com o tamanho do sistema alvo.

Palavras-Chave: Segurança de Sistemas Web, Scanners de Vulnerabilidade, Integração de Sistemas, Serviços sob Demanda, Computação em Nuvem

ABSTRACT

Nowadays, to keep Web systems safe is one of the greatest challenges of the industry and academy. Companies need to spend increasing amounts of money to minimize the risk of security incidents and losses for the business. One way of mitigating risks, while still reducing the amount of investments, is by outsourcing the services that are not part of the business portfolio. This work aims to contribute on fulfilling this gap, i.e., provide a solution for offering on-demand security services. To achieve the goal we propose a architecture for offering security services, in particular Web application vulnerability scanners, in a simplified and elastic way. A solution to this problem can be achieved through an integrated system that receives the clients' requests, automatically executes vulnerability scanners, as well as process and send the output of the scanners to the customer. The elasticity of the system can be achieved through cloud computing, more precisely by virtual machines that can be dynamically allocated in cloud providers such as Amazon, Google and Rackspace. By having a dynamic pool of resources, the system is capable of supporting a variable number of vulnerability scanners and a potentially unlimited number of clients without affecting or compromising the services provided to the end users. A prototype implementation, named SafeWeb, was developed to evaluation the proposed architecture. Lastly, we also discuss some of the challenges and future work, such as the accurate measurement of the resources required by each scanner, which provides the means for defining the costs of the services and also to optimized the use of the available resources. Another future work is to come up with an adequate financial model for the proposed system, which takes into account the size and complexity of the target systems, for instance.

Keywords: Security of Web Systems, Vulnerability Scanners, Systems Integration, Services on Demand, Cloud Computing

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PHP – Hypertext Preprocessor
HTML – Hypertext Markup Language
HTTP – HyperText Transfer Protocol
XML – eXtensible Markup Language
API – Application Programming Interface
IaaS – Infrastructure as a Service
PaaS – Plataform as a Service
SaaS – Software as a Service
SOAP – Simple Object Application Protocol
WSDL – Web Service Description Language
UDDI – Universal Definition Discovery Interface
XSS – Cross-site scripting
SQL – Structured Query Language
SQLI – Structured Query Language Injection
TI – Tecnologia da Informação

SUMÁRIO

1. INTRODUÇÃO

Manter um ambiente seguro em uma aplicação na Internet, requer conhecimentos e ferramentas da área de segurança de sistemas. Identificar e corrigir as vulnerabilidades dos sistemas pode economizar tempo e dinheiro de uma empresa, bem como elevar o nível de confiança e credibilidade junto aos seus clientes.

Este trabalho tem como objetivo analisar o cenário atual de segurança de sistemas Web, propondo uma solução de sistemas e ferramentas capazes de auxiliar empresas e usuários a manter bons níveis de segurança em suas aplicações Web. Para isso, foi realizado um estudo de caso para identificar os requisitos, desafios e possibilidades de integração de sistemas Web e de segurança, de forma escalável, para oferecer o sistema SafeWeb, uma solução capaz de prover serviços de segurança sob demanda.

1.1. Contexto e Motivação

Com o avanço da tecnologia, a internet faz parte do cotidiano da maioria da população e vem mudando a maneira como vivemos, criando novas possibilidades a cada nova invenção. A Web 2.0, também conhecida como “Web como plataforma”, revolucionou a internet, saindo de um mundo essencialmente estático para um profundamente dinâmico. E um dos aspectos marcantes, sempre presente, é a liberdade de os usuários criarem sistemas e conteúdos através de plataformas e tecnologias existentes, cada vez mais acessíveis.

Em contrapartida, a liberdade e flexibilidade vem com um preço. Enquanto que pessoas menos experientes conseguem criar sistemas e disponibilizar conteúdos de forma descontrolada e desordenada, usuários mal intencionados aproveitam este cenário para encontrar falhas de segurança nos sistemas, que passaram a ser comuns, permitindo-os tomar controle, roubar dados, infringir direitos autorais ou ainda afetar a disponibilidade de sistemas e conteúdos.

O cenário começa a ficar cada vez mais complicado, uma vez que a maioria das empresas e organizações passam a depender de sistemas Web para crescerem e continuarem competitivas num mercado globalizado. Sistemas online permitem atingir mais consumidores e novos mercados, que ultrapassam barreiras geográficas.

Logo, a segurança e a confiabilidade desses sistemas passa a ser um ponto crucial para o desenvolvimento e sucesso do negócio. Nesta perspectiva, empresas passam a preocupar-se cada vez mais com formas para manter os seus sistemas seguros, confiáveis e disponíveis.

Ao mesmo tempo em que o número de ameaças e incidentes de segurança aumentam, cresce também a quantidade de ferramentas e recursos capazes de identificar vulnerabilidades e falhas de um sistema, como é o caso do Uniscan, Rocha (2012). Entretanto, o uso dessas ferramentas por si só não resolvem o problema. É preciso uma infraestrutura com máquinas e pessoas treinadas para uma solução efetiva. Contudo, as empresas e instituições públicas e privadas não podem ou não tem interesse em manter infraestruturas e equipes de segurança próprias. A solução para esse impasse passa pela terceirização de serviços especializados de segurança.

Existem poucas empresas oferecendo serviços de segurança online, sob demanda. Isso aponta para um nicho de mercado com possibilidades e oportunidades a serem exploradas. Serviços de segurança sob demanda poderão, num futuro próximo, começar a fazer parte do cotidiano da maioria das empresas na Web, uma vez que não há solução definitiva devido ao fato de os sistemas estarem em constante evolução, e novas tecnologias surgirem. Consequentemente, a necessidade da verificação e comprovação da segurança de sistemas tende a ser um ciclo sem fim, perpétuo.

Oferecer serviços de segurança sob demanda implica em uma infraestrutura de computação elástica. Esta demanda vai de encontro com a computação em nuvem, cujo principal foco é prover serviços, plataformas e infraestruturas de computação sob demanda. Isso faz com que as nuvens sejam capazes de oferecer a elasticidade necessária para o desenvolvimento de sistemas escaláveis, que podem dinamicamente atender mais ou menos clientes sem necessitar de uma infraestrutura e custos fixos.

Por fim, a principal motivação do trabalho surge do fato de praticamente inexistirem trabalhos que demonstram os desafios e possibilidades de integração de soluções de segurança (como scanners de vulnerabilidades) e sistemas Web, sobre ambientes elásticos, para prover serviços de segurança sob demanda. Vale também ressaltar que apenas o fato de executar ferramentas de segurança em ambientes em nuvem é um desafio por si só. Isso deve-se ao fato de infraestruturas de rede, em

especial, possuírem diferentes sistemas de detecção de ataques e mecanismos automáticos de controle. Um scanner de vulnerabilidades gera grandes quantidades de conexões e requisições aos sistemas Web, o que pode ser interpretado como um ataque por sistemas de segurança ao nível da rede, por exemplo. Portanto, também é necessário investigar a formas de prestar serviços de segurança online sem incorrer em novos problemas.

1.2. Objetivos

Objetivo Geral

Propor uma arquitetura de integração de aplicações de segurança e sistemas Web para oferecer serviços de segurança sob demanda. Esta arquitetura deve ser elástica e capaz de atender a diferentes tipos de demanda e usuários.

Para realizar os testes de segurança, ferramentas como o scanners de vulnerabilidades serão utilizadas de maneira integrada, ou seja, através de uma interface Web simples e prática, onde o usuário final pode contratar os serviços conforme a sua necessidade.

Objetivos Específicos

O objetivo geral do trabalho pode ser pormenorizado conforme segue.

a) Integrar sistemas Web com ferramentas de segurança como scanners de vulnerabilidades (exemplo: Uniscan);
b) Criar um sistema que explore recursos em nuvem, oferecendo elasticidade à solução proposta;
c) Identificar e relatar os principais problemas e desafios no desenvolvimento dos três objetivos anteriores.

1.3. Justificativa

Problemas de segurança como ataques a vulnerabilidades de sistemas, injeção de código, acesso não autorizado e ataques de negação de serviço são comuns em ambientes conectados em rede, como as aplicações Web em especial. Segundo relatórios recentes, como 2013 Data Brach Investigations Report (2013), 76% dos ataques ocorrem por brechas no sistema ou credenciais roubadas. Mais assustador ainda é que o número, a força e a criticidade dos ataques vem crescendo, sem uma perspectiva de diminuir.

Scanners de vulnerabilidades estão entre as ferramentas mais corriqueiramente utilizadas para identificar e diagnosticar falhas e problemas em sistemas Web. Consequentemente, eles são um importante recurso para mitigar ataques e intrusões em aplicações online.

Contudo, em primeiro lugar, a segurança de sistemas ainda é uma área nova e pouco desenvolvida na maioria dos países, como é o caso do Brasil. Além disso, há uma carência de formação e disponibilidade de mão-de-obra qualificada na maioria dos países. Em segundo lugar, as empresas não tem condições e/ou interesse em investir e manter uma equipe de segurança que seja especializada em identificar, diagnosticar e resolver problemas de segurança dos mais diversos sistemas e ambientes da infraestrutura 2013 Data Branch Investigations Report (2013).

Dito isso, o caminho natural para resolver o problema passa pela terceirização de serviços especializados de segurança, como identificação e diagnóstico de fraquezas dos sistemas. É neste sentido que surge o potencial interesse latente em sistemas online de segurança sob demanda.

Com a ascensão da computação em nuvem, serviços online podem ser frequentemente contratados sob demanda. O cliente paga pelo serviço na hora que precisa e na quantidade que precisa, não necessitando manter uma equipe de segurança dentro da sua folha de pagamento para manter um sistema ou serviço especializado em particular.

Os custos operacionais para manter uma infraestrutura para ferramentas de segurança bem como máquinas, instalação, atualização e equipes treinadas, pode muitas vezes extrapolar a quantia que uma empresa dispõe para manter seu negócio.

Dado esse cenário, a terceirização do serviço de segurança oferece uma solução para pequenas e médias empresas que não dispõem de recursos para manter a segurança de seus sistemas.

Desta forma, o trabalho se justifica pelo fato de contribuir com o desenvolvimento de serviços de segurança sob demanda, necessários para atender uma demanda crescente e latente do mercado. O estudo de caso do trabalho poderá servir de base para gerar conhecimento na área de segurança em sistemas Web, integração entre diferentes sistemas e a criação de um modelo de negócio para serviços Web.

1.4. Estrutura do trabalho

O Quadro 1 resume a estrutura e abordagem previstas para o trabalho.

Quadro 1 - Estrutura do Trabalho

Capítulo	Abordagem
Capítulo 1	Compreende introdução, contexto e motivação da pesquisa, objetivos e justificativa do trabalho.
Capítulo 2	Apresentação da fundamentação teórica.
Capítulo 3	Este capítulo aborda a solução proposta para resolver o problema apresentado e os passos a seguir para a sua concepção.
Capítulo 4	Este capítulo descreve como foi implementada a arquitetura proposta no capítulo 3.
Capítulo 5	Este capítulo apresenta o ambiente de testes, a instalação, os scanners utilizados, os resultados obtidos e os desafios propostos para trabalhos futuros.
Capítulo 6	No último capítulo, são expostas as considerações finais do trabalho de acordo com o desenvolvimento e resultados atingidos.

Fonte: Elaboração Própria

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Segurança na Web

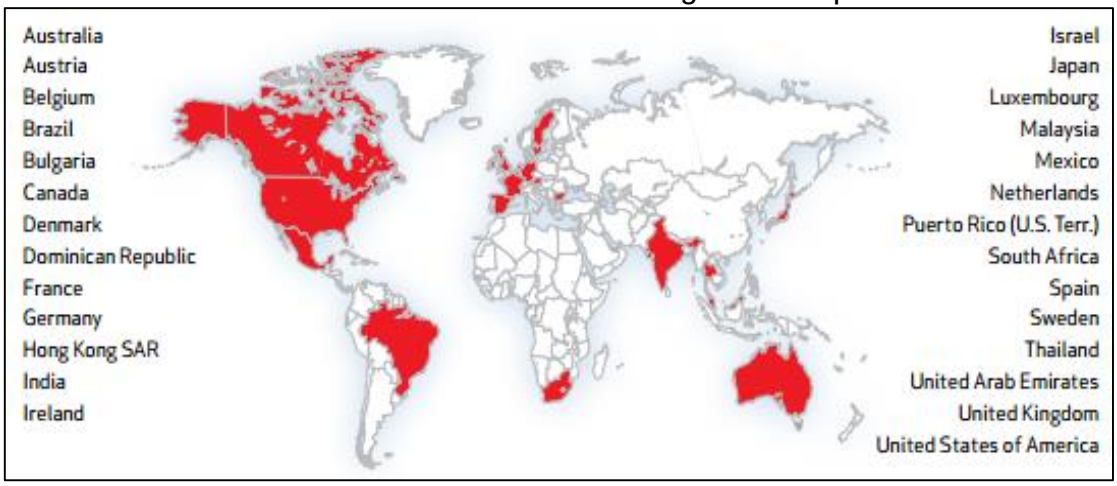
Estudos recentes feitos pelo Instituto Ponemon (2012), sugerem que o custo de segurança cresceu 6% em 2012 com um aumento de 42% de ataques cibernéticos. O custo médio dado as empresas chegou a $591,780 dólares.

É necessário pouca habilidade para encontrar vulnerabilidades e criar maneiras de explora-las para iniciar um ataque. Devido a isso, existe um existe um mercado negro para os criminosos cibernéticos, no qual ferramentas e produtos podem ser facilmente adquiridos com o objetivo de atacar e/ou invadir sistemas online. Nesse mercado, hackers encontram e vendem vulnerabilidades para empresas ou indivíduos mal intencionados.

Análise de segurança

A segurança na Web é um ponto crítico e existem poucas soluções disponíveis no mercado para resolver este problema. Segundo o 2013 Data Breach Investigations Report (2013), em 27 países pesquisados 78% das organizações são consideradas como baixa dificuldade de intrusões. Identificar e corrigir estes problemas pode ser vital para o crescimento da organizações. A Figura 1 ilustra os países investigados.

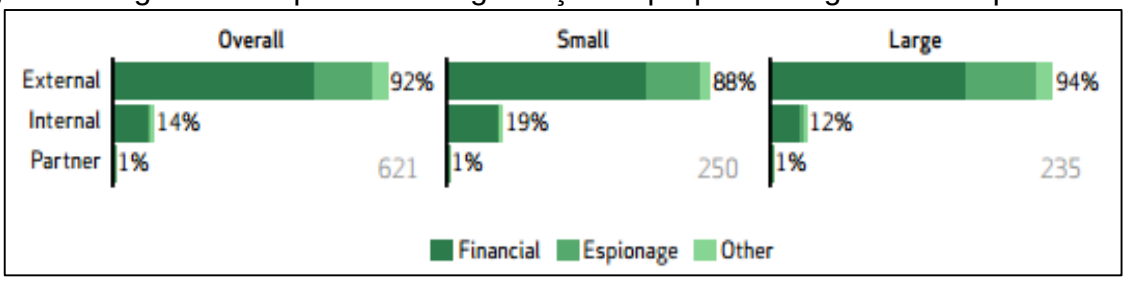
Figura 1 – Países avaliados no Data Breach Investigations Report



Fonte: 2013 Data Breach Investigations Report (2013)

Como maior parte dos ataques são de origem externas às empresas, fica evidente a necessidade de investimento em segurança. Na Figura 2, vemos que em geral, 92% das falhas de segurança são caudas por agentes externos.

Figura 2 – Agentes na quebra de segurança em pequenas e grandes empresas



Fonte: 2013 Data Breach Investigations Report (2013)

Para contribuir na resolução de alguns dos problemas de segurança na Internet, em especial em sistemas Web, existem dois tipos de testes que são executados pelas ferramentas de análise de segurança: os testes dinâmicos e os estáticos.

Testes dinâmicos requerem esforço humano para investigar os códigos da aplicação e a detecção de falhas acontece manualmente.

Segundo Chen et. al., (2013), testes estáticos consistem em checagens de sintaxe no código realizados de forma pré-definida e automática. Estes testes podem ser feitos na etapa de desenvolvimento da aplicação e encontrar falhas nas primeiras etapas de desenvolvimento.

Testes estáticos não requerem mão-de-obra especializada, ou seja, possuem um custo baixo para as empresas. Pequenas e médias empresas em sua maioria optam por essa solução por ser viável e por obter bons resultados.

Em comparação com testes dinâmicos, os testes estáticos tem vantagem por detectarem erros muito mais rápido e de maneira que encontram problemas escondidos no código, diferentemente do que analisar os sintomas apresentados e procurar pelos erros.

Ralph Decker (2010) propôs que a essência da computação em nuvem deveria ser utilizada para realizar os testes de segurança. Baseados no modelo de computação em nuvem, os testes apresentariam todos os benefícios que ela oferece, como escalabilidade e configuração, permitindo o cliente a contratar o serviço que precisa e pagar somente pelo que precisa.

Ferramentas de segurança

Existem no mercado algumas empresas que prestam serviços online de segurança. Entre elas, TREND MICRO™, WEBSECURIFY™ e ON-S™.

A TREND MICRO™ oferece uma ferramenta gratuita para avaliar o nível de segurança da empresa. A solução pode ser utilizada para planejar e implementar medidas de segurança necessários para garantir que o sistema Web esteja protegido contra ataques. O processo de análise é constituído de um questionário com 25 perguntas de respostas sim/não que mede o nível de fragilidade do sistema. A partir das respostas, o sistema apresenta os resultados e as sugestões de como aperfeiçoar os sistemas das empresa no sentido de remover eventuais falhas de segurança.

A WEBSECURIFY™ é um scanner de vulnerabilidades multiplataforma que pode ser utilizado como um complemento (ADDON) para navegadores. Para realizar os testes de segurança, o utilizador precisa baixar e configurar um componente de software da WEBSECURIFY que é integrado ao navegador do usuário para realizar os testes de segurança no sistema web alvo. Ao final da execução, o scanner gera um relatório contendo um resumo do problemas de segurança encontrados no sistema alvo, como e-mails cadastrados no site, usuários, falhas na programação e bugs.

Testes de vulnerabilidades em aplicações Web

Baul (2010) realizou um teste de vulnerabilidades em aplicações Web utilizando oito diferentes ferramentas que verificaram os tipos de vulnerabilidades encontrados por estes scanners, a eficiência contra vulnerabilidades alvo e a relevância entre as vulnerabilidades alvo comparadas com as vulnerabilidades encontradas por acaso. Os scanners testados podem ser vistos no Quadro 2.

Quadro 2 - Scanners testados

Empresa	Produto	Versão	Perfis usados no Scanneamento
Acunetix	WVS	6.5	Default and Stored XSS
Cenzic	HailStorm Pro	6.0	Best Pratices, PCI Infrasctructure, and Session

HP	WebInspect	8.0	All Checks
IBM	Rational AppScan	7.9	Complete
McAfee	McAfee SECURE	Web	Hack Simulation and DoS
N-Stalker	QA Edition	7.0	Everything
Qualys	QualysGuard PCI	Web	N/A
Rapid7	NeXpose	4.8	PCI

Fonte: Baul 2010

Para testar a eficiência das ferramentas, foram utilizadas diferentes aplicações Web com falhas de segurança conhecidas. O Quadro 3 resume os sistemas, as respectivas vulnerabilidades e os resultados dos scanners de vulnerabilidades.

Quadro 3 - Ameaças conhecidas vs Ameaças encontradas utilizando os scanners

Categoria	Drupal 4.7		phpBB2 2.0.19		Wordpress 1.5strayhorn	
	Conhecidas	Encontradas	Conhecidas	Encontradas	Conhecidas	Encontradas
XSS	6	2	5	2	13	7
SQLI	2	1	1	1	8	4
XCS	4	0	1	0	8	3
Session	5	4	4	4	6	5
CSRF	2	0	1	0	1	1
Info Leak	4	3	1	1	6	4

Fonte: Baul 2010

Como mostrado no experimento de Bau, os resultados indicam que as falhas mais encontradas foram as do tipo Cross-Site Scripting e SQL Injection. Contudo, os resultados ainda deixam a desejar com relação aos testes avançados de XSS e SQLI, outras formas de Cross-Channel Scripting, Cross-Site Request Forgery, e presença de Malware. Em resumo, um dos caminhos futuros é a utilização de uma combinação variada de ferramentas para vasculhar vulnerabilidades de sistemas.

2.2. Integração de Sistemas

> Entende-se por integração de Sistemas de Informação a partilha de informação e processos entre aplicações em rede ou fontes de dados numa organização (MARTINS, 2005, p. 7).

Na computação, integração de sistemas é definido como um processo que une diversos componentes de subsistemas de forma que trabalhem em conjunto para formar um único sistema. Além disso, a integração entre sistemas permite às organizações o suporte a constantes exigências e mudanças que seu meio ambiente pode ter.

Para realizar a integração de sistemas podem ser utilizadas técnicas como RPC (Remote Procedure Call) e Objetos Distribuídos. Estas técnicas são empregadas na comunicação entre os diferentes tipos de sistemas e protocolos.

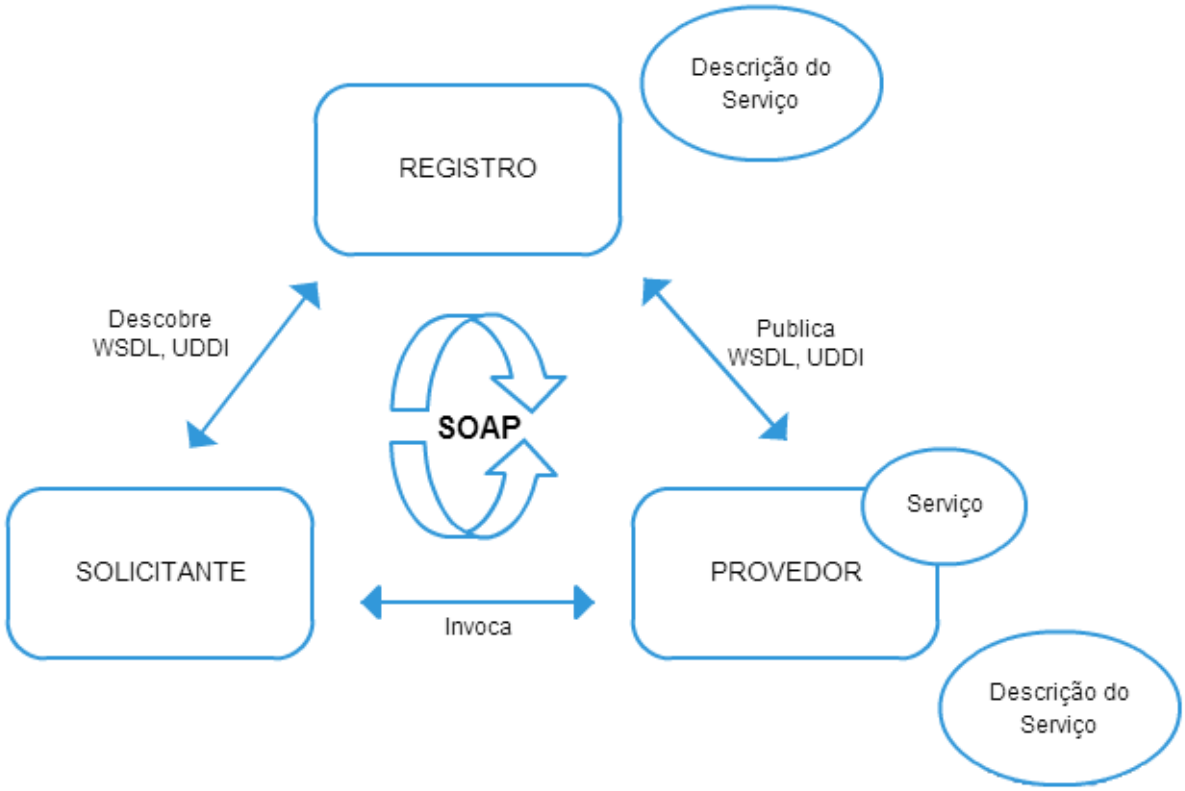
Web Services

De acordo com a W3C, Web Service, é uma aplicação de software identificada por um Uniform Resource Identifier (URI), que contém interfaces e ligações definidas e descritas como artefatos XML. Segundo Booth et. al., (2004), Um serviço Web suporta interações diretas com outras aplicações usando mensagens baseadas em XML, trocadas através de protocolos baseados na Internet.

Os Web Services permitem uma comunicação uniformizada entre diferentes plataformas e sistemas. Em outras palavras, um Web Service é um sistema de software projetado para suportar interações máquina-a-máquina através de uma rede. Os usuários de Web Services não precisam saber nada sobre a plataforma, modelo, ou linguagem de programação que foi utilizada para implementar o serviço. Em suma, a integração entre diferentes sistemas e plataformas é transparente, ou seja, realizada através das interfaces dos Web Services.

Conforme Muller (2007), a arquitetura de um Web Service é baseada na interação de três papeis, apresentada na Figura 3:

Figura 3 - Arquitetura de Web Services



Fonte: Muller (2007)

O provedor de serviço: é o espaço onde o serviço está hospedado. O criador do Web Service, responsável por fazer sua descrição em algum formato padrão e publicar os detalhes em um registro de serviço central.

O solicitante de serviço: é a aplicação que invoca ou inicializa uma interação com um serviço. Pode ser usando um navegador ou um Web Service. Usando a descrição do serviço, é possível descobrir e invocar Web Services.

O registro de serviço: é o local onde os provedores publicam as descrições dos serviços. O solicitante consulta o registro, busca pelas informações de ligação da descrição do serviço durante a fase de desenvolvimento ou em tempo de execução.

A interação entre os papeis ocorre através de mensagens baseadas em XML. SOAP (Simple Object Access Protocol) é um exemplo de formato amplamente utilizado na comunicação com e/ou entre Web Services.

Simple Object Access Protocol (SOAP)

O SOAP é um protocolo para a troca de informações baseado na linguagem XML e realiza a troca de mensagens em um ambiente distribuído e descentralizado. Pode ser usado com a combinação de outros protocolos como HTTP, SMTP e JMS.

Para entender o SOAP, é necessário levar em conta as três interações entre cliente, servidor e mensagem. O cliente é um programa que irá criar o documento XML que contém a informação necessária para invocar um método remoto. O servidor é responsável por transmitir a mensagem SOAP e age como interpretador das diferentes linguagens. E a mensagem contém o conjunto de regras que será usado para invocar os métodos da aplicação no servidor.

RESTful Web Services

A Transferência de Estado Representativo (Representational State Transfer) ou (REST), segundo Fielding (2000), é pretendida como uma imagem do design da aplicação se comportará.

Uma rede de Websites (um estado virtual), onde o usuário progride com uma aplicação selecionando as ligações (transições do estado), tendo como resultado a página seguinte (que representa o estado seguinte da aplicação) que está sendo transferida ao usuário e apresentada para seu uso.

O REST define um conjunto arquitetural onde podem ser desenvolvidos Web Services que focam em sistemas de recursos, incluindo como os recursos são endereçados e transferidos através do HTTP em uma diferente gama de linguagens. Além disso, REST teve um impacto tão grande na Web que acabou por tornar obsoletos os sistemas baseados em SOAP e WSDL. O principal motivo é o fato de a interface REST ser considerada mais simples e prática Rodriguez (2008).

RESTful usa a arquitetura REST aplicada na Web para aproveitar dos benefícios de performance, escalabilidade e mobilidade. Tipicamente os serviços RESTful usam os quatro principais métodos HTTP, create, retrieve, update e delete mostrados no Quadro 4.

Quadro 4 - Métodos HTTP e suas operações

Método HTTP	Operação Realizada
GET	Pega um recurso
POST	Cria um recurso e outras operações já que não tem semântica definida
PUT	Cria ou atualiza um recurso
DELETE	Deleta um recurso

Fonte: RESTful Web Services Developer's Guide (2010)

Banco de dados

Um banco de dados representa coleções de dados que existem por um período de tempo e que são gerenciadas por um Sistema Gerenciador de Banco de Dados (SGBD). Segundo Takai et. al. (2005), um SGBD é uma coleção de programas que permitem aos usuários criarem e manipularem uma base de dados. Um SGBD é, assim, um sistema de software de propósito geral que facilita o processo de definir, construir e manipular bases de dados de diversas aplicações.

A necessidade de uma ou mais organizações podem requerer uma relação de uma coleção de dados, e para isso é necessário um sistema capaz de realizar e gerenciar essas relações. O uso do banco de dados, ao invés de arquivos, resulta numa maior disponibilidade, facilidade de acesso e atualização das informações, e torna possível a execução de transações complexas com redundância de dados mínima.

2.3. Computação em Nuvem

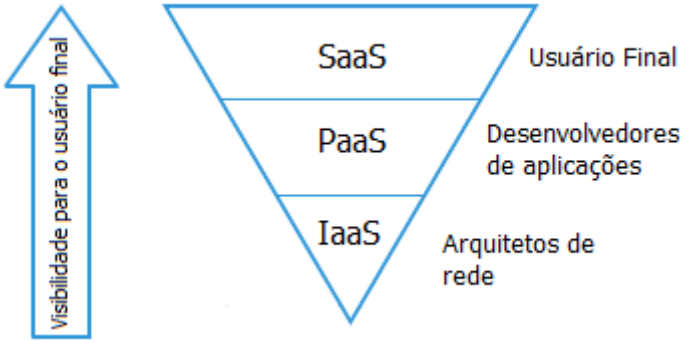
Segundo o NIST (National Institute of Standards and Technology) (2011), Computação em Nuvem é um modelo que permite acessar de maneira onipresente, conveniente e sob demanda recursos configuráveis de computação (como por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem

rapidamente ser supridos e lançados com o mínimo de esforço ou interação com o provedor do serviço.

Modelos de serviços

Existem três tipos de modelos de Computação em Nuvem, como pode ser observado na Figura 4. Cada modelo define um diferente nível de poder de controle do usuário sobre a infraestrutura, como a aplicação existe, como ela é acessada, e assim por diante.

Figura 4 - Visibilidade dos modelos em computação em nuvem



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

Permite ao usuário a utilizar os serviços disponibilizados pelo provedor de nuvem. As aplicações podem ser acessadas por vários dispositivos como um computador ou celular. Exemplos: Gmail, Google Docs, Dropbox, etc.

O cliente não gerencia ou controla as camadas de mais baixo nível, como rede, servidores, sistema operacional e armazenamento.

- **Plataform as a Service (PaaS)**

Permite ao cliente o serviço da infraestrutura da nuvem para que sejam hospedadas as aplicações criadas pelo consumidor usando linguagens de programação, bibliotecas, serviços, e ferramentas suportadas pelo provedor do serviço. Exemplo: Google Cloud Platform.

O cliente não gerencia ou controla as camadas de mais baixo nível, como rede, servidores, sistema operacional, armazenamento. Entretanto, o cliente possui controle sobre as aplicações implantadas e pode configurar o ambiente da aplicação.

- **Infrastructure as a Service (IaaS)**

Um ambiente IaaS provê processamento, armazenamento, rede e outros recursos fundamentais da computação para que o cliente. Cabe a este decidir quais sistemas operacionais, bibliotecas e configurações de sistema e rede a utilizar. Exemplo: Amazon EC2.

O cliente não tem controle sobre a infraestrutura da nuvem, mas tem controle sobre o sistema operacional, armazenamento e as aplicações implementadas. O cliente também tem a possibilidade de controlar configurações da rede, como firewalls.

O nível de controle de cada modelo de serviço é mostrado no Quadro 5.

Quadro 5 - Nível de controle na nuvem

	Software	Configuração de Ambiente	Controle de Rede	Controle de SO
SaaS	X			
PaaS	X	X		
IaaS	X	X	X	X

De uma maneira simples, os modelos SaaS, PaaS e IaaS são substitutos para a infraestrutura tradicional com o diferencial do modelo de comercialização, que, ao invés de licenciamento, utiliza um modelo baseado em pagamento por utilização de recursos.

Modelos de Nuvens

Existem quatro modelos de Computação em Nuvem. Cada modelo provê características e propriedades distintas.

- **Nuvem Privada**

A infraestrutura da nuvem é provisionada para use exclusivo de uma organização comprometida com múltiplos consumidores (ex. unidade de negócio). Ela pode ser gerenciada e operada pela própria organização, por um serviço terceirizado ou uma combinação dos dois.

- **Nuvem Comunitária**

A infraestrutura da nuvem é provisionada para uso exclusivo de uma comunidade de consumidores com um interesse compartilhado (ex. missão, requerimentos de segurança, política e considerações de conformidade). Ela pode ser gerenciada e operada por uma ou mais organizações da comunidade, por um serviço terceirizado ou uma combinação deles.

- **Nuvem Pública**

A infraestrutura da nuvem é aberta para uso aberto do público geral. Pode ser gerenciada e operada por uma empresa, universidade, governo ou uma combinação deles.

- **Nuvem Híbrida**

A infraestrutura da nuvem é uma composição de duas ou mais distintas infraestruturas (privada, comunitária ou pública) que retém entidades únicas, mas estão unidas por padrões ou propriedades tecnológicas que permitem a portabilidade dos dados e aplicação.

O nível de visibilidade de cada modelo de nuvem é mostrado no Quadro 6.

Quadro 6 - Visibilidade na nuvem

	Organização/Empresa	Comunidade	Público Geral

Nuvem Privada	X		
Nuvem Comunitária	X	X	
Nuvem Pública	X	X	X

Fonte: Elaboração Própria

Benefícios e malefícios de usar computação em nuvem

Uma empresa ou um negócio, na maioria das vezes, começa pequeno e com pouco investimento. Serviços na nuvem garantem a elasticidade necessária para estas empresas, ou seja, contrata-se gradualmente apenas o que é necessário. Caso a empresa cresça, ela pode acabar necessitando usar cinco mil servidores para manter seu negócio funcionando. Caso o negócio falhe, basta a empresa encerar o contrato com o provedor de nuvem.

Segundo o valor da computação em nuvem, DELL (2011), 80% dos gastos de uma empresa estão relacionados com a infraestrutura de manter o negócio fora da nuvem. Em outras palavras, sobram apenas 20% para desenvolver novas metas e objetivos, o que representa um desperdício estratégico considerável para o negócio.

Para equilibrar e reduzir os gastos com infraestrutura de tecnologia e profissionais especializados, a empresa pode optar por contratar um serviço de acordo com a sua necessidade e ir expandindo ou retraindo conforme sua necessidade. Desta forma a empresa corre menos riscos e acaba por gastas mais efetivamente e apenas o necessário em infraestrutura.

Benefícios

A seguir alguns benefícios que um serviço baseado em computação em nuvem pode oferecer.

- **Redução de custos**

Manter uma estrutura e equipes de TI pode ser oneroso para uma empresa ou instituição. Uma alternativa é contratar um serviço hospedado em uma nuvem e pagar sob demanda. Ou ainda, migrar os serviços da empresa para uma infraestrutura alocada sob demanda (IaaS), evitando os gastos locais com estrutura física e equipes de manutenção.

- **Escalabilidade/Flexibilidade**

O dimensionamento da demanda não precisa ser previamente bem definido. A empresa pode ir contratando recursos sob demanda, conforme as suas necessidades momentâneas. Esta flexibilidade e elasticidade provida pelos serviços em nuvem evita que as empresas tenham que investir muito para manter infraestruturas que atendem as demandas de pico e, também, reduz os riscos associados ao negócio, uma vez que o contrato do serviço pode sessar a qualquer momento. Já no caso de uma estrutura própria, não há como simplesmente, de uma hora para outra, se desfazer dela.

- **Disponibilidade**

Serviços podem ser hospedados em múltiplos locais de forma redundante para suportar a continuidade do serviço. Desta forma, são capazes de contornar desastres como a perda de dados caso algum servidor falhe, ou estrague.

- **Manutenção**

Provedores de computação em nuvem fazem a manutenção no sistema dentro do próprio servidor, então não requerem a instalação nem atualização da aplicação nos computadores, reduzindo assim o tempo e dinheiro gasto em manutenção de software e hardware.

- **Acessibilidade móvel**

Dispositivos móveis podem acessar o serviço de qualquer lugar, a qualquer hora. Aumentando assim a produtividade dos usuários do serviço.

Aspectos Negativos

Ao utilizar a computação em nuvem, o usuário entrega seus dados e informações importantes aos cuidados de outra empresa. Para muitos, o sigilo dessa informação deve ser preservada, mas qual a garantia que a empresa que fornece o serviço em nuvem oferece para seu usuário?

O principal problema da computação em nuvem está relacionado com a privacidade dos dados. Como tudo é armazenado na nuvem e a infraestrutura física é gerenciada pelo provedor da nuvem, não há garantias de que a privacidade dos dados não seja violada. Portanto, é necessário avaliar os riscos e quais tipos de informações serão armazenadas na nuvem para que não haja o vazamento de informações indesejadas.

Elasticidade de sistemas Web usando Computação em Nuvem

> Mil computadores ligados por uma hora, custa o mesmo que um computador ligado por mil horas. Usando paralelismo o resultado seria gerado mil vezes mais rápido (HARA, 2011, p. 6).

Recursos podem ser adquiridos e retraídos conforme a demanda da aplicação. Isto se torna um componente chave na busca para diminuir o desperdício de recursos computacionais. Para os usuários os recursos são transparentes dando a sensação de serem ilimitados, com capacidade dinâmica de expansão e retração.

Na Figura 5 exemplifica a comparação entre a capacidade de um servidor, e os momentos em que eles ficam inativos devido à baixa demanda, e os momentos em que estão sobrecarregados. A computação em nuvem consegue gerenciar estes momentos de forma rápida e prática.

Figura 5 – Uso de recursos comparados com a capacidade e desperdício

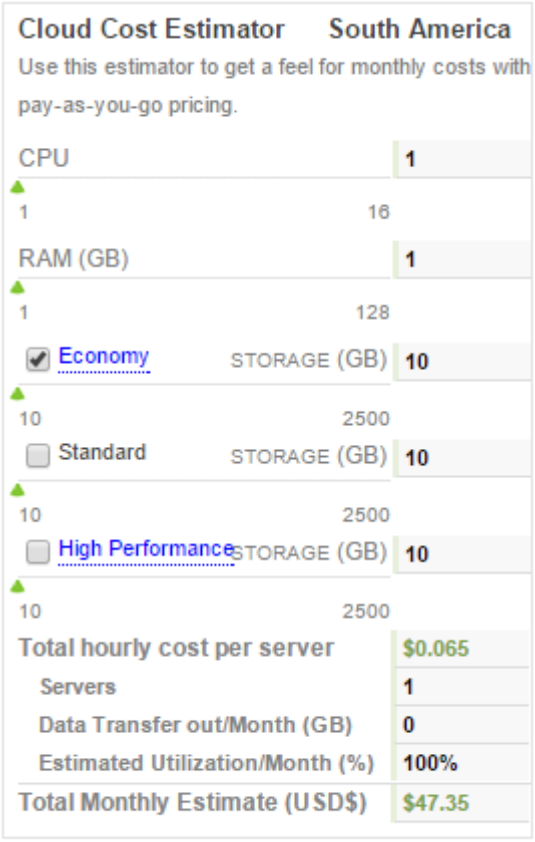
Fonte: M. Armbrust et. al., CACM (2010)

A propriedade elástica dos serviços em nuvem é oferecida através do modelo IaaS onde são criadas várias máquinas virtuais (VM), gerenciadas por um monitor de máquinas virtuais (VMM) que compartilham e gerenciam os recursos de hardware das máquinas físicas. Cada máquina virtual tem seu próprio sistema operacional e é isoladas das demais.

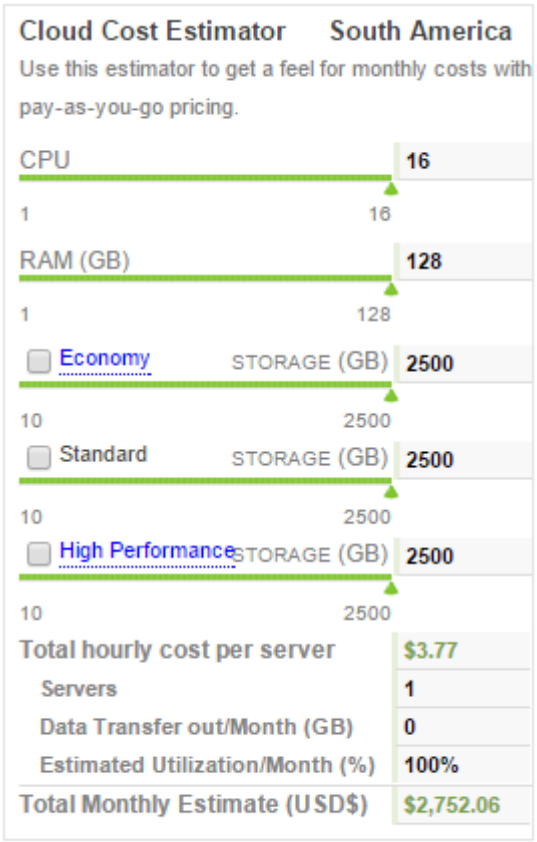
2.4. Modelos de Negócio para Serviços Online (sob demanda)

Os custos para a contratação de serviços sob demanda variam conforme número de recursos utilizados, como núcleos de processador, memória RAM e espaço em disco. De acordo com o site Dimension Data Cloud, em setembro de 2013, os preços para a contratação de serviços de nuvem sob demanda mensais variam entre $48.18 e $3, 898.20 dólares conforme configuração e utilização do serviço, como pode ser visto na Figura 6.

Figura 6 – Custo estimado do serviço de computação em nuvem



Custo mínimo



Custo máximo

Fonte: Dimension Data Cloud 2015

Em comparação, a empresa Cloud Vertical oferece diferentes pacotes de serviços de computação em nuvem com preços mensais variando entre $19.71 e $1, 576.80 dólares conforme visto na Tabela 1.

Tabela 1 – Comparação de preços de serviços em nuvem sob demanda da empresa Cloud Vertical

Instance type	**CPU**	**RAM**	**On-demand**	**Spot**	**Reserved**
Second Generation Medium (m3.medium)	1	3.75	$0.070	N/A	$0.027
Second Generation Double Extra Large (m3.2xlarge)	26	30	$0.560	N/A	$0.219
Small (m1.small)	1	1.7	$0.044	Min $0.0270 < $0.700 < max $3.000	$0.015
Medium (m1.medium)	2	3.75	$0.087	min $0.0380 < $0.038 < max $3.000	$0.031
Large (m1.large)	4	7.5	$0.175	min $0.1080 < $0.108 < max $0.120	$0.061
Extra Large (m1.xlarge)	8	15	$0.350	min $0.2160 < $0.216 < max $0.900	$0.123
High-CPU Medium (c1.medium)	5	1.7	$0.130	min $0.0540 < $0.054 < max $1.000	$0.049
High-CPU Extra Large (c1.xlarge)	20	7	$0.520	min $0.2160 < $0.640 < max $10.000	$0.194

Fonte: Cloud Vertical 2015

Os valores descritos na coluna *Spot* são atualizados a cada hora e representam o preço mínimo, médio e máximo. Os valores na coluna *Reserved* são referentes a amortização de preço por hora caso seja feita uma reserva mínima de consumo.

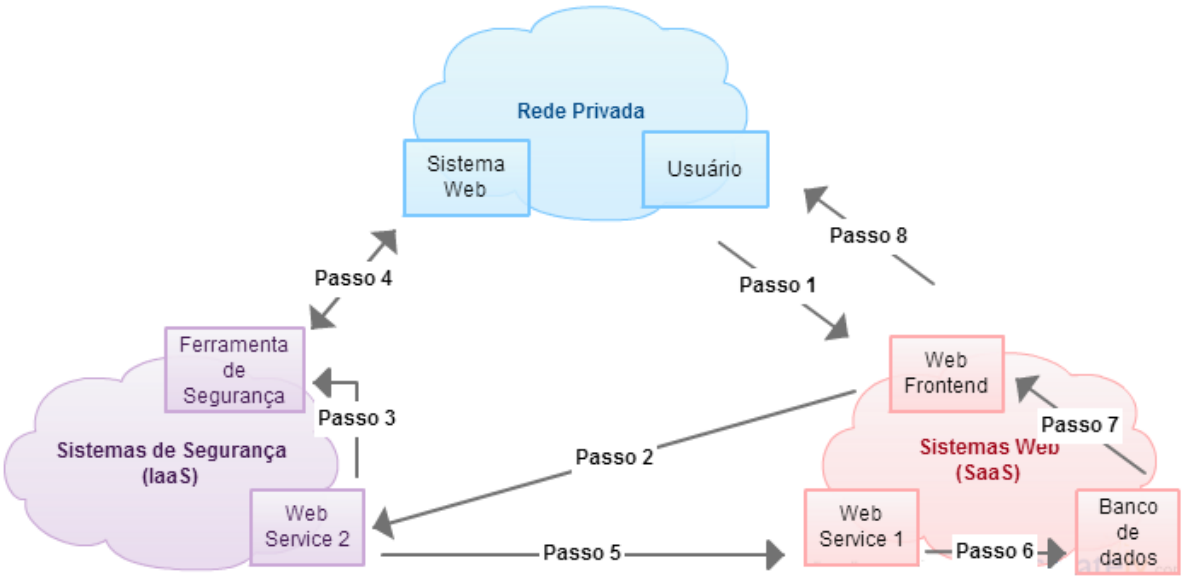
3. SOLUÇÃO PROPOSTA

A seguir é apresentada uma proposta para os problemas de segurança em aplicações Web. A solução passa por uma arquitetura dinâmica e flexível, capaz de integrar diferentes sistemas e recursos de infraestrutura para prover serviços de segurança sob demanda.

3.1. Visão Geral da Arquitetura

A Figura 7 apresenta uma visão geral da arquitetura do SafeWeb e a integração dos diferentes sistemas. São essencialmente três grandes blocos, ilustrados pelas nuvens. O primeiro é a rede privada, que representa os usuários e seus respectivos sistemas Web, que serão analisados pelas ferramentas de segurança. Depois, a nuvem dos sistemas Web (SaaS) abrange os recursos necessários para oferecer um sistema online onde os usuários podem facilmente contratar serviços de segurança sob demanda. Estes sistemas também atuam como Frontend das ferramentas de segurança. Por fim, a terceira nuvem contém as ferramentas de segurança propriamente ditas. Estas são executadas contra os sistemas Web dos usuários, gerando analises e relatórios dos problemas encontrados. Adicionalmente, as nuvens SaaS e IaaS possuem Web Services que são utilizados como forma de integração entre os dois ambientes. Finalmente, a Figura 7 ilustra também a sequência de passos entre uma requisição do usuário e o resultado final (relatório das ferramentas de segurança).

Figura 7 - Visão Geral do Funcionamento do SafeWeb



Para melhor compreensão dos passos exemplificados na Figura 7, serão descritas a seguir as 8 etapas que demonstram o funcionamento do sistema:

1: O cliente requisita o serviço de análise de segurança de um dos seus sistemas Web ao servidor Web Frontend. O Sistema Web registra a demanda no banco de dados.

2: A demanda do cliente é enviada para o Web Service 2.

3: O sistema associado ao Web Service 2 analisa a requisição e ativa as respectivas ferramentas de segurança.

4: As ferramentas de segurança executam os testes de vulnerabilidades no sistema Web do cliente e geram os respectivos relatórios de vulnerabilidades e problemas de segurança.

5: Os resultados das análises são enviados para o Web Service 1.

6: O Web Service 1 salva os relatórios e dados no banco de dados.

7: O Frontend Web é notificado pelo Web Service 1 assim que os dados foram consolidados no banco de dados.

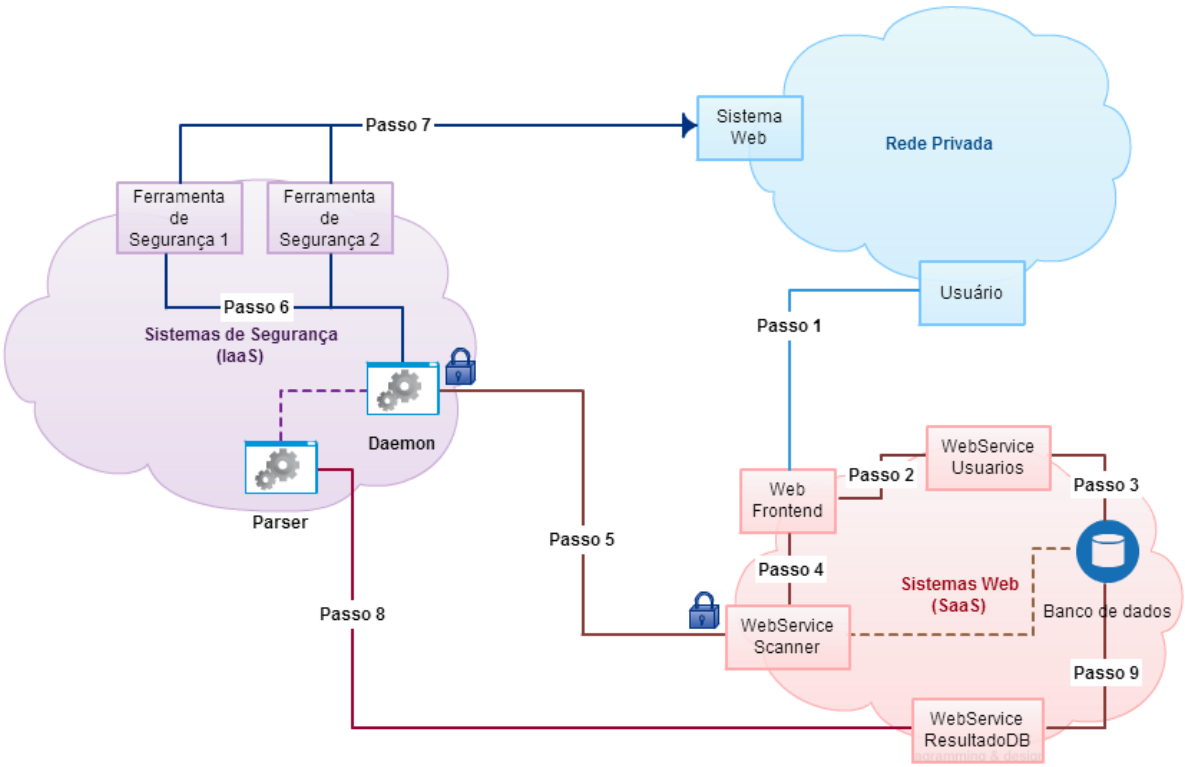
8: O cliente é notificado, isto é, informado que os dados dos relatórios estão disponíveis para visualização.

4. IMPLEMENTAÇÃO

A Figura 8 representa a implementação da arquitetura proposta na Figura 7. Na nuvem de sistemas Web há um servidor Web a rodar o front-end, um sistema de gerenciamento de banco de dados e três Web services (WSs). O primeiro WS é utilizado para gerir o cadastro e o acesso dos usuários ao front-end. O segundo WS gerencia o lançamento dos scanners de vulnerabilidades de acordo com as demandas dos usuários. Este WS comunica-se com os serviços (Daemons) a executar nas máquinas virtuais que executarão os scanners. Novas máquinas virtuais podem ser geradas, sob demanda, para executar mais scanners, o que confere elasticidade à solução proposta. Por fim, o terceiro WS recebe o resultado, devidamente processado, da execução dos scanners de vulnerabilidades realizada pelos serviços ativados na nuvem de sistemas de segurança.

A utilização de Web Services deixa a arquitetura do sistema modular e escalável, uma vez que as aplicações e serviços não possuem acesso direto ao banco de dados. Consequentemente, novas instâncias de quaisquer componentes do sistema podem ser facilmente ativadas.

Figura 8 - Arquitetura do SafeWeb implementado



As 9 etapas ilustradas na Figura 8 são:

1: O cliente acessa o sistema através do *Web Front-end* para realizar seu cadastro. O processo passa pelo preenchimento de um formulário com os dados essenciais do cliente, como nome, e-mail e senha.

2: O *Web Front-end* envia os dados do cadastro para o *Web Service* Usuários através do método POST.

3: O *Web Service* Usuários recebe a solicitação de cadastro em formato XML, extrai os dados cadastrais e armazena-os no banco de dados.

4: Após completar o cadastro com sucesso, o usuário pode realizar uma solicitação de análise de segurança do sistema Web alvo. O usuário seleciona o(s) *scanner(s)* desejados, define a URL do sistema alvo e finaliza a solicitação. O *Front-end* envia a requisição do usuário para o *Web Service Scanner*.

5: O *Web Service Scanner* recebe o endereço do sistema Web a ser varrido, o identificador de qual scanner será utilizado e os parâmetros necessários para a execução do mesmo. Antes de enviar o pedido ao *daemon*, o WS gera também uma *HASH* de autenticação e verificação de integridade dos dados, que será enviada junto com o pedido, e registra a solicitação no banco de dados. Finalmente, a requisição é enviada ao *Daemon* através de um canal TCP/IP.

6: O *Daemon* recebe a solicitação e verifica a autenticidade e integridade da mesma. Essa verificação é realizada através da *HASH* única que é gerada a partir do conteúdo da mensagem e uma chave secreta que é partilhada entre o *Daemon* e o *Web Service Scanner*. Em outras palavras, apenas os dois elementos conseguem gerar e verificar uma *HASH* válida para cada solicitação. Após confirmada a autenticidade e integridade dos dados recebidos, o *Daemon* executa as respectivas ferramentas (*scanner* de vulnerabilidades).

7: A ferramenta é executada para encontrar eventuais vulnerabilidades no Sistema Web do cliente. Ao término do processo de verificação é gerado um relatório de vulnerabilidades e problemas de segurança encontrados. Esses dados de saída são, então, processados por um *parser* (específico a cada *scanner*) que é responsável por filtrar e padronizar os resultados.

8: Os dados de saída do relatório são processados por um *parser* (específico a cada scanner) que é responsável por filtrar e padronizar os resultados e, então, o relatório é enviado para o *Web Service ResultadoDB*.

9: O *Web Service ResultadoDB* salva os resultados no banco de dados.

4.1. Sistema Web

O sistema web desenvolvido em PHP5 foi organizado em 4 blocos diferentes:

- ***Layout*:** Este bloco é responsável pelo visual do sistema *Web n*ão contendo portanto, nenhuma funcionalidade do sistema. Alterando os arquivos *layout_cabecalho.php* e *layout_rodape.php* é possível editar a interface visual sem se preocupar com a funcionalidade do sistema.

- ***Form*:** Este bloco contém os formulários para preenchimento de dados. Caso seja necessário alguma alteração nos formulários para adicionar mais campos de entrada, é necessário editar os arquivos com os nomes começados em *form_*.

- ***Handle*:** Este bloco trata os arquivos digitados pelo usuário. Caso seja necessária uma alteração na formatação dos dados, ou a inclusão de novos campos nos formulários, é necessário alterar este bloco a fim de salvar os dados de maneira correta. Os arquivos desse bloco estão nomeados com o prefixo *handle_*.

- ***Web Services*:** Intermediário entre o sistema PHP e o servidor das ferramentas de segurança e banco de dados. Os *Web Services* são os responsáveis pela interação com o banco de dados do sistema. São eles que leem e escrevem os dados no banco de dados. Foram utilizados 3 *Web Services*, o *user.php* responsável pelo cadastro e login de usuários, o *scanner.php* responsável pela ativação das

ferramentas de segurança e o *resultados.php* responsável por importar os relatórios para o banco de dados.

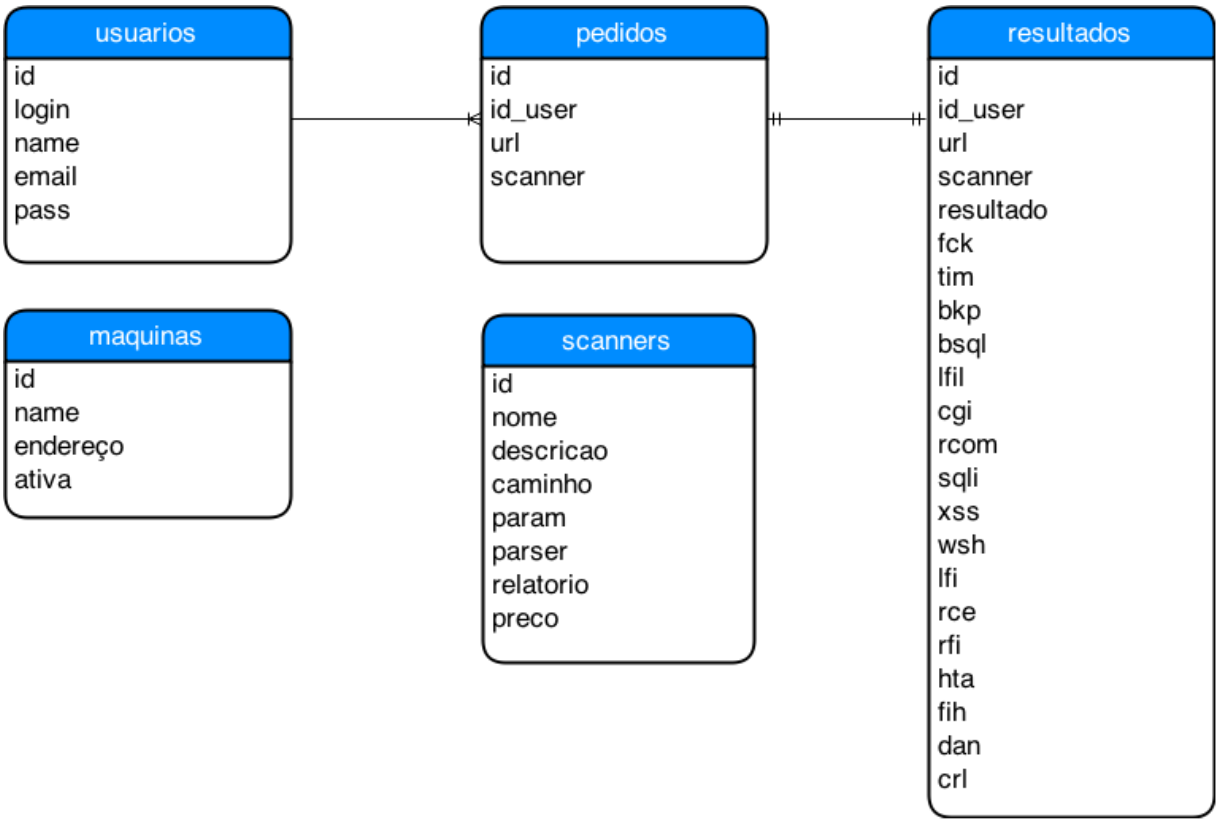
4.2. Banco de dados

O banco de dados MySQL foi utilizado em conjunto com phpMyAdmin para gerenciar e visualizar os dados.

A base de dados criada contém cinco tabelas para guardar os dados de usuários cadastrados, pedidos de escaneamentos, resultados e configurações dos *scanners* e máquinas virtuais.

A Figura 9 mostra o diagrama ER das tabelas utilizadas no sistema SafeWeb.

Figura 9 - Diagrama ER Banco de Dados



A tabela *usuarios* contém os dados cadastrais dos usuários do sistema. O cadastro guarda os dados de login, nome, email e senha. A senha é guardada no formato codificado md5.

A tabela *pedidos* contém os pedidos de escaneamento solicitados pelo usuário. É guardado a url, o ID do scanner e a chave estrangeira *id_user* que guarda o id do usuário.

A tabela *scanners* guarda as configurações das ferramentas de segurança instaladas no servidor. A instalação de novas ferramentas de segurança no sistema requer o cadastro de uma nova linha nesta tabela além do parser necessário para formatar o relatório. Um exemplo pode ser visto na Tabela 2.

Tabela 2 - Tabela *scanner*

id	***nome***	***descricao***	***caminho***	***param***	***parser***	***relatorio***	***preco***
1	*Wapiti*	*Meu primeiro scanner*	*wapiti*	*-n 10 -b folder -u -v 1 -f html -o relatorios/wapiti/*	*/opt/lampp/htdocs/safeweb/format/jsontobd.php*	*relatorios/wapiti/vulnerabilities.json*	*1,00*

A tabela resultados possui o ID do usuário que ativou a ferramenta, a url, o ID do scanner, o resultado que compõe o relatório completo em HTML da ferramenta que fica armazenado codificado em base64, e o número de falhas encontradas para cada um dos testes descritos nas colunas detalhadas a seguir:

- **fck:** *FCKeditor*
- **tim:** *Timthumb* < 1.33 *vulnerability*
- **bkp:** Arquivos de *backup*
- **bsql:** *Blind SQL Injection*
- **lfil:** Inclusão de arquivo Local
- **cgi:** Inclusão de arquivo Local estático
- **rcom:** Execução de comando remoto
- **sqli:** *SQL Injection*
- **xss:** *Cross-Site Scripting* (*XSS*)
- **wsh:** *Web Shell Finder*
- **lfi:** Inclusão de arquivo Local estático
- **rce:** Inclusão de arquivo local
- **rfi:** Inclusão de arquivo remoto
- **hta:** *Htaccess Bypass*
- **fih:** *File Handling*
- **dan:** Arquivos potencialmente perigosos
- **crl:** *CRLF Injection*

Manter os resultados de forma uniformizada, separando a quantidade de falhas encontradas em seus respectivos testes, tem como objetivo organizar os resultados e fornecer um comparativo entre os scanners. O responsável por fazer essa fragmentação é o parser do scanner.

A tabela maquinas possui a lista de máquinas virtuais disponíveis e o endereço IP onde se encontram. Cada linha nessa tabela representa uma máquina diferente. Esta tabela é importante para permitir a elasticidade do sistema possibilitando o uso de várias máquinas.

Um exemplo de configuração pode ser visto na Tabela 3:

Tabela 3 - Tabela maquina

Id	Name	endereco	ativa
1	VM_1	unix:///var/run/mysocket.sock	0
2	VM_2	200.201.200.123:80	0

A coluna ativa representa se a máquina está ou não sendo usada por um scanner. Os responsáveis por gerenciar as máquinas ocupadas e alterar esse valor são, o Web Service scanner que altera o valor para 1, sinalizando que a máquina está ocupada, e o parser que altera o valor para 0, sinalizando que a máquina está ociosa e pronta para executar um scanner.

4.3. Web Services

Para o funcionamento do sistema, foram implementados três Web Services, sendo eles: user, resultados e scanner. Os dois primeiros interagem com o banco de dados e o terceiro ativa as ferramentas de segurança. Dentre os frameworks testados, incluindo CakePHP e Zend Framework, foi optado pela utilização de uma classe RESTFul encontrada no livro Samisa Abeysinghe RESTful PHP Web Services. Essa classe fornece o conjunto de operações CRUD (Criar, Ler, Atualizar e Deletar) pronto para utilização. A utilização dessa classe é feita através das chamadas das operações necessárias.

Exemplo:

Método GET
$result = $client->get("http://localhost/safeweb/ws/user.php", array());

Método PUT
$result = $client->post("http://localhost/safeweb/ws/user.php", $data, "text/xml");

Documento XML

O método POST da classe RESTful utiliza a linguagem XML para a troca de mensagens. O XML é uma linguagem de marcação de simples leitura utilizada para encodar dados em tags e foi definida pela W3C's XML 1.0 Specification.

A seguir um exemplo de uma marcação XML:

```
<usuarios>
   <usuario>
         <nome>João</nome>
         <sobrenome>Da Silva</sobrenome>
   </usuario>
</usuarios>
```

Detalhes dos *Web Services*

A seguir serão detalhados os três *Web Services* implementados:

- ***user.php***

O *Web Service user* é o responsável pelo cadastro de novos usuários no sistema *Web*. A página *cadastro.php* contém o formulário necessário para o cadastro e a *handle_cad.php* formata os dados do cadastro para o formato XML. Quando o usuário se cadastra no site, a página *handle_cad.php* envia o XML para o *Web Service user.php* que checa a entrada do formulário e guarda cadastro no banco de dados. A Tabela 4 mostra os métodos utilizados:

Tabela 4 - Web Service user

URL	Método	Coleção	Operação	Descrição
/ws/user	POST	usuarios	Criar	Cadastro de usuário
/ws/user	GET	usuarios	Retornar	Lista todos usuários
/ws/user/{id}	GET	usuarios	Retornar	Lista usuário {id}
/ws/user/login	POST	usuarios	Retornar	Realizar Login

O método POST /ws/user/ é utilizado para o cadastro de novos usuários e é executado pela página *handle_cad.php* que envia um documento XML. O método GET /ws/user/ é utilizado para a listagem de usuários cadastrados no sistema.

O método /ws/user/{id} retorna um usuário identificado pela chave primária ID. E o método /ws/user/login/ é utilizado para realizar o login no sistema.

O formato do documento XML que o *Web Service user*.php utiliza é o que segue:

```
<users>
   <user>
         <name></name>
         <login></login>
         <email></email>
         <pass></pass>
   </user>
</users>
```

- **resultados.php**

O Web Service resultados é o responsável por guardar os relatórios das ferramentas de segurança no banco de dados. Quando a ferramenta de segurança termina o escaneamento, e o parser da ferramenta formata o relatório, o documento XML é enviado para o Web Service, cuja a função é armazenar os resultados no banco de dados e notificar o cliente que o relatório está pronto para ser acessado na página resultados.php. É importante ressaltar que a chamada desse Web Service é dada dentro do parser de cada ferramenta, sendo necessário a formatação dos dados e o envio para o Web Service seja feito em cada um dos parsers. A seção 4.5 contém mais detalhes sobre as tags de formatação do XML. A Tabela 5 mostra os métodos utilizados:

Tabela 5 - Web Service resultados

URL	Método	Coleção	Operação	Descrição
/ws/resultados	POST	Resultados	Criar	Guardar relatório
/ws/resultados/{id_user}	GET	Resultados	Retornar	Lista todos testes
/ws/resultados/{id_user}/teste/{id_teste}	GET	Resultados	Retornar	Detalhes do relatório {id_teste}

O método POST /ws/resultados recebe o documento XML enviado pelo parser e salva no banco de dados o resultado do scanners.

O método GET /ws/resultados/{id_user} retorna os resultados das solicitações realizadas pelo respectivo usuário, identificado pela chave primária id_user.

O método GET /ws/resultados/{id_user}/teste/{id_teste} retorna o relatório detalhado (saída completa do scanner) do teste identificado pelo seu ID. Isso permite ao usuário visualizar, também, as saídas detalhadas das ferramentas de segurança.

- **scanner.php**

O Web Service scanner é o responsável pela ativação das ferramentas de segurança. A página order.php, contém o formulário necessário para o registro do pedido de escaneamento e a handle_order.php formata os dados para o formato XML. Quando o usuário registra um pedido de escaneamento, a página handle_order.php recupera do banco de dados os parâmetros necessários para a execução do scanner, constroi a mensagem e gera uma HASH de autenticação e verificação de integridade. Esta HASH é utilizada pelo Daemon para autenticar as requisições do Web Service scanner. A Tabela 6 mostra os métodos utilizados:

Tabela 6 - Web Service scanner

URL	Método	Coleção	Operação	Descrição
/ws/scanner	POST	orders	Criar	Ativa a ferramenta de segurança escolhida e registra no banco de dados o pedido

O formato do documento XML que o Web Service scanner.php utiliza é o que segue:

```
<orders>
   <order>
         <client></client>
         <url></url>
         <scanner></scanner>
   </order>
</orders>
```

O método POST /ws/scanner envia os parâmetros necessários para ativação das ferramentas de segurança.

4.4. *Daemon*

Para realizar a comunicação entre o sistema Web e as ferramentas de segurança, foi desenvolvido um Daemon em linguagem C. A comunicação entre o Web Service scanner e o Daemon é realizada através de TCP/IP. Cada mensagem recebida pelo Daemon é verificada em termos de autenticidade e integridade.

O processo de autenticação e verificação de integridade conta com uma chave partilhada (entre o Daemon e o Web Service), que é combinada com a mensagem na geração da HASH. Somente se a autenticação e verificação de integridade conferirem, i.e., HASH recebida é igual a HASH gerada pelo Daemon (mensagem + chave partilhada), a ferramenta de segurança é executada.

Os parâmetros da mensagem recebida são: scanner, url, parâmetros, id_user, cp_relatório, cd_parser, cd_scanner e digest.

- **scanner:** ID do *scanner* no banco de dados a ser executado.

- **url:** Endereço do sistema *web* a ser testado pelos *scanners*.
  EX: www.google.com
  127.0.0.1/sistemaweb/

- **Parâmetros:** Parâmetros da ferramenta de segurança a ser ativada.
  Ex: -qweds

- **id_user:** ID do usuário no banco de dados que ativou a ferramenta.

- **cp_relatório:** Caminho do relatório onde é salvo o relatório da ferramenta. Este parâmetro é recuperado do banco de dados, na linha referente a ferramenta que será ativada.
  EX: relatorios/wapiti/

- **cd_parser:** Nome do arquivo *parser* da ferramenta a ser ativada.
  EX: htmltobd.php

- **cd_scanner:** Comando necessário para ativar a ferramenta.
  EX: ./uniscan.pl
  wapiti

- **digest:** *HASH* da chave compartilhada + mensagem.

A mensagem recebida contém uma cadeia de caracteres que é delimitada pelo operador "**#**". O Daemon separa a mensagem em parâmetros que serão enviados para a função chamada_scanner() que irá ativar as ferramentas posteriormente.

Exemplo de mensagem recebida:
1#192.168.205.132/joomla/#-qweds#1#-1#/opt/lampp/htdocs/safeweb/format/htmltobd.php#./uniscan.pl –u#

Ativando a Ferramenta de Segurança

O Daemon executa a função *chamada_scanner* (char scanner, char url, char parametros, char i_user, char cp_relatorio, char cd_parser, char cd_scanner) que irá ativar a ferramenta.

A função chamada_scanner constrói, dinamicamente, o comando a ser executado conforme os parâmetros recebidos e ativa as ferramentas através da função system().

4.5. *Parser*

O *parser* é um analisador de sintaxe de *strings*. Dado um padrão linguagem, o *parser* percorre a *string* para encontrar informações ou checar se a *string* pertence a aquele padrão testado. Neste trabalho, o *parser* foi utilizado para contar o número de vulnerabilidades encontradas dentro dos relatórios de segurança gerados pelos *scanners*.

Cada *scanner* possui um formato diferente de relatório, então, para isso, é necessário uma análise especifica para cada um. Os *scanners* instalados no sistema precisam de um *parser* próprio para poder formatar o relatório de acordo com o padrão previamente definido.

O *parser* é responsável por calcular a quantidade de falhas descobertas no relatório do *scanner*, procurando pelo padrão de palavras único que cada relatório possui. Para realizar os testes deste trabalho, foram desenvolvidos dois parsers, o *safeweb/format/htmltobd.php* que formata os resultados do Uniscan e o *safeweb/format/jsontobd.php* que formata os resultados do Wapiti.

Um *parser* requer quatro argumentos:

- **id_user:** ID do usuário que ativou a ferramenta.
- **scanner:** ID do scanner que foi utilizado.
- **url:** Endereço do sistema testado.
- **arquivo:** Nome do arquivo do relatório gerado.

A chamada que ativa a execução do parser é ilustrada a seguir:

php /opt/lammp/htdocs/safeweb/format/htmltobd.php 1 1 192.168.205.132/joomla/ 192.168.205.132.html

Após formatar os resultados, o *parser* envia um documento XML, cujo formato é definido a seguir, para o *Web Service resultados*.

```
<results>
    <resulta>
        <id_user></id_user>
```

```
            <url></url>
            <resultado></resultado>
            <scanner></scanner>
            <fck></fck>
            <tim></tim>
            <bkp></bkp>
            <bsql></bsql>
            <lfil></lfil>
            <cgi></cgi>
            <rcom></rcom>
            <sqli></sqli>
            <xss></xss>
            <wsh></wsh>
            <lfi></lfi>
            <rce></rce>
            <rfi></rfi>
            <hta></hta>
            <fih></fih>
            <dan></dan>
            <crlf></crlf>
        </resulta>
</results>
```

5. AVALIAÇÃO

Nesta seção, serão discutidos os testes realizados, a instalação e configuração, as ferramentas de segurança utilizadas e a elasticidade do sistema.

5.1. Ambiente de testes

O desenvolvimento e os testes foram realizados em máquinas virtuais.

Máquina física: A máquina física utilizada durante o desenvolvimento e testes foi um notebook DELL equipado com um processador *Intel® Core™ i5-4200U CPU @ 1.60GHz 2.30GHz*, memória instalada (RAM): *4,00GB*, Sistema Operacional de *64 bits Windows 8.1* e processador com base em x64.

Rede: Rede local Ethernet 100 Mbit/s.

Máquinas virtuais: A máquina virtual utilizada possui 2GB de memória RAM, 10GB de espaço em disco rígido e processador com um núcleo.

Emulador de Máquinas Virtuais: O emulador utilizado foi o *VMware® Player 6.0.3 build-1895310.*

Sistemas Operacionais: O sistema utilizado na máquina física foi o *Windows 8.1, 64-bit (Build 9600) 6.3.9600* e o sistema utilizado no emulador da máquina virtual foi o *Ubuntu 13.04, 32-bit*.

Servidor Web: O servidor Web utilizado foi o *Apache/2.4.4 (Unix) OpenSSL/1.0.1* e *mod_perl/2.0.8-dev Perl/v5.16.3*

Versão do compilador: O compilador utilizado para compilar o *Daemon* foi o *gcc (Ubuntu/Linaro 4.7.3-1ubuntu1) 4.7.3*.

Versão do PHP: *PHP 5.4.16*

Versão do XAMPP: *XAMPP for Linux 1.8.2*

Versão do Banco de dados: *MYSQL (Build mysqlnd 5.0.10 – 20111026)*

5.2. Instalação e Configuração

A instalação está dívida em três módulos conforme descrito abaixo:

- Banco de Dados
- Sistema Web
- Ferramentas de segurança

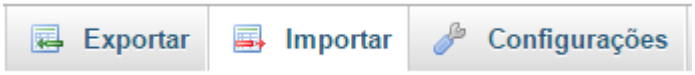
Scripts do Banco de Dados

O primeiro passo para a instalação do sistema SafeWeb é a criação das tabelas no banco de dados.

- Abrir o gerenciador do banco de dados. Neste guia será utilizado como exemplo o phpMyAdmin hospedado em uma máquina localizada em http://192.168.205.131/phpmyadmin.

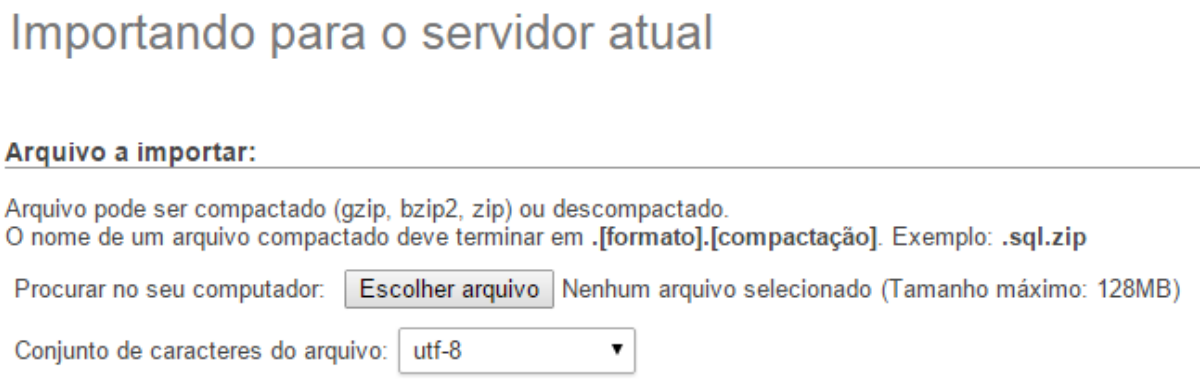
- Encontrar a opção *Importar* e clique nela. Conforme vista na Figura 10.

Figura 10 - Importar base de dados



- Importar o arquivo base.sql clicando na opção Escolher arquivo conforme vista na Figura 11.

Figura 11 - Escolher arquivo base.sql



Se tudo estiver certo, a base de dados e as tabelas terão sido criadas.

Instalação do Sistema Web

O segundo passo para a instalação é copiar os arquivos PHP para o servidor Web.

- Copiar a pasta *safeweb* para a raiz do servidor web.
- Abrir o arquivo *db_connection.php* e configure o *login* e usuário do banco de dados.

Compilando *Daemon*

O terceiro passo para a instalação é compilar o *Daemon*.

- Compilar o arquivo *server.c* utilizando o comando:
  # gcc server.c –o server

Adicionando Máquinas Virtuais e Scanners

O último passo é adicionar as configurações das máquinas virtuais disponíveis e os parâmetros dos *scanners* que se deseja utilizar.

- Insirir na tabela *maquinas* o nome e endereço da(s) máquinas virtuais disponíveis conforme o exemplo na Tabela 7:

Tabela 7 - Exemplo de configuração das máquinas virtuais

id	***nome***	***Endereço***	***Ativa***
1	*VM_1*	*unix:///var/run/mysocket.sock*	*0*
2	*VM_2*	*200.120.214.234:50*	*0*

- Insirir na tabela scanners os parâmetros necessários para a utilização dos scanners que serão utilizados conforme a Tabela 8.

Tabela 8 - Exemplo de parâmetros dos *scanners*

id	***nome***	***descricao***	***caminho***	***Param***	***parser***	***relatorio***	***preco***
1	*Uniscan*	*Meu primeiro scanner*	*./uniscan.pl -u*	-qweds	/opt/lampp/htdocs/safeweb/format/htmltobd.php	*-1*	*10,00*
2	*Wapiti*	*Meu segundo scanner*	*wapiti*	*-n 10 -b folder -u -v 1 -f html -o relatorios/wapiti/*	*/opt/lampp/htdocs/safeweb/format/jsontobd.php*	*relatorios/wapiti/vulnerabilities.json*	*1,00*

Para maior informações sobre a instalação de novas ferramentas, consulte a seção 5.5.1.

5.3. *OWASP Broken Web Applications Project* (Conjunto de aplicações vulneráveis)

Para realizar os testes das ferramentas Uniscan e Wapiti, foi utilizado o pacote de sistemas Web vulneráveis *OWASP Broken Web Applications Project*, disponibilizado pelo *OWASP* (*Open Web Application Security Project*), uma organização sem fim lucrativos que tem como objetivo aprimorar a segurança na internet.

OWASP Broken Web Applications Project é uma coleção de aplicações web com vulnerabilidades projetadas para testes e práticas de invasões. O Projeto consiste em uma máquina virtual rodando uma variedade de aplicações com vulnerabilidades conhecidas e é disponibilizado de graça através do (lei software livre) de código aberto para as pessoas com interesse em aprender sobre segurança de aplicações web [owasp.org/index.php/OWASP_Broken_Web_Applications_Project].

Sistemas Vulneráveis

Os sistemas utilizados escolhidos para serem utilizados nos testes foram os seguintes:

CryptOMG challenge1: Sistema em PHP para treinar habilidades *hacker*.

CryptOMG challenge4: Sistema em PHP para treinar habilidades *hacker*.

Vicnum: Sistema em PHP que simula um jogo de adivinhação de palavras e números.

Wivet: Sistema em PHP fictício com acesso a um banco de dados.

Webcal: Sistema PHP de calendário.

Apesar do *OWASP Broken Web Applications Project* fornecer a informação de quais vulnerabilidades os sistemas possuem, não é dito a quantidade de vulnerabilidades que cada sistema possui. Sendo assim, os testes demonstrarão se

foram ou não encontradas as falhas conhecidas do sistema vulnerável. Na Tabela 9 é discriminada a informação das vulnerabilidades conhecidas de cada sistema.

Tabela 9 - Vulnerabilidades conhecidas dos sistemas

Sistema	sqli	xss	bkp	lfil	cgi	LFI
CryptOMG/clg1/	X	X		X		X
CryptOMG/clg4/	X	X		X	X	X
Vicnum	X	X		X		X
Wivet	X	X	X	X		X
Webcal	X	X	X	X		X

5.4. *Scanners* utilizados

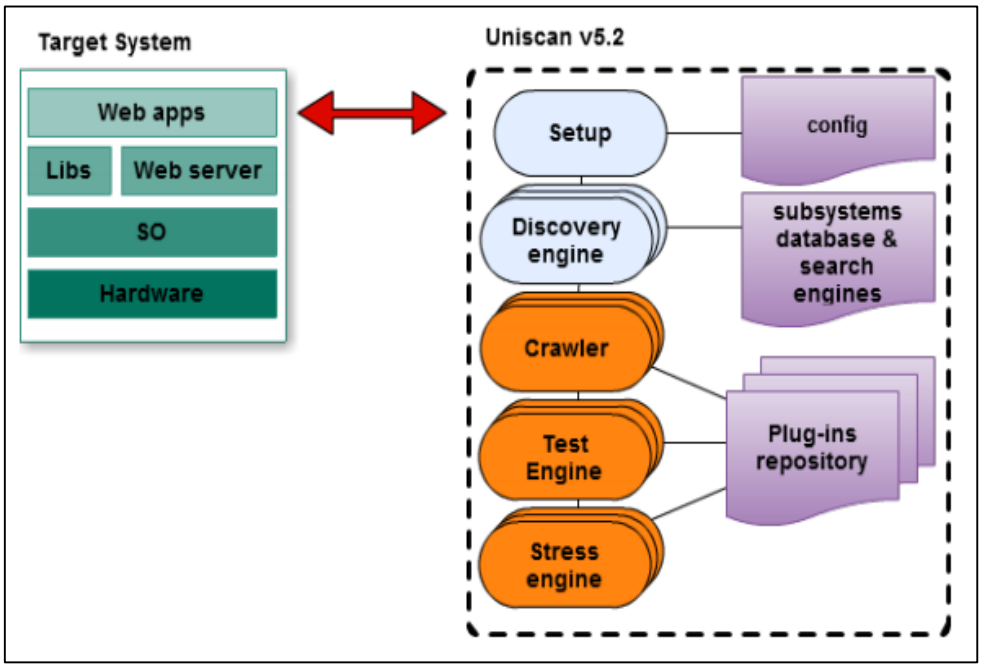
Para testar o sistema SafeWeb foram utilizados dois *scanners*: Uniscan 6.2 e Wapiti 2.3, ambos são disponibilizados gratuitamente para *download* via internet. Cada *scanner* possui uma maneira diferente de realizar os testes de segurança, pontos fracos e fortes, conforme serão descritos nas próximas seções.

Uniscan

O Uniscan é uma ferramenta de *software* livre desenvolvida para escanear e detectar falhas de segurança em sistemas Web. O scanner faz uma varredura no *Website* procurando por falhas conhecidas como inclusão de arquivos remotos, inclusão de arquivos locais e execução remota de comandos.

Para realizar os testes de segurança, o Uniscan utiliza um rastreador para localizar todos os arquivos e links dentro do site alvo. Na sequência, o verificador de vulnerabilidades realiza vários testes sobre cada arquivo ou link encontrado. A Figura 12 representa a arquitetura do Uniscan.

Figura 12 - Representação da arquitetura do Uniscan



Fonte: Rocha (2012)

A versão utilizada para os testes foi a 6.2 e pode ser encontrada no repositório *SourceForge* da ferramenta localizada no endereço: *http://sourceforge.net/projects/uniscan*.

O Uniscan foi desenvolvido em PERL e tem com o objetivo de encontrar as vulnerabilidades RFI, LFI e RCE eficazmente, tornando assim um ponto forte da ferramenta, Rocha (2012). Mas o *scanner* não se limita apenas a esses testes, ele ainda consegue encontrar com sucesso falhas do tipo SQL *Injection* e *Cross-Site Scripting*.

Wapiti

O *Scanner* Wapiti realiza uma auditoria rápida do sistema *Web*. Através de *scanners* "black-box" que não leem os códigos dos arquivos, mas escaneam os formulários das páginas onde é possível injetar dados. Após encontrar as páginas, começa a bateria de teste injetando dados através de *scripts* para encontrar as vulnerabilidades.

A versão utilizada para os testes foi a 2.3 e pode ser encontrada no repositório *SourceForge* da ferramenta localizada no endereço: *http://wapiti.sourceforge.net/*

O Wapiti foi desenvolvido em Python para testar sistemas desenvolvidos em PHP, ASP e JSP. Suporta ataques utilizando os métodos HTTP GET e POST para injeção de dados SQL e XPath. O Wapiti consegue detectar falhas do tipo XSS, RFI, LFI, RCE e CRLF. Porém deixa um pouco a desejar na identificação de falhas quando comparada com o scanner Uniscan.

5.5. Testes

Nesta seção são discutidos os resultados e como foram feitos os testes de segurança realizados nos sistemas vulneráveis.

Os testes dos *scanners* (Uniscan e Wapiti), foram realizados em duas máquinas virtuais, uma contendo os sistemas vulneráveis disponibilizados pela OWASP e a outra contendo as ferramentas de segurança.

Resultados dos scanners

A Tabela 10 e 11 mostram os resultados de falhas encontradas e o tempo de execução para cada um dos scanners utilizados.

Abreviações:

- **bsql:** *Blind SQL Injection*
- **sqli:** SQL *Injection*
- **xss:** *Cross-Site Scripting* (XSS)
- **bkp:** Arquivos de *backup*
- **lfil:** Inclusão de arquivo Local
- **cgi:** Inclusão de argumento PHP CGI
- **LFI:** Inclusão de arquivo Local estático

Tabela 10- Resultados do *Scanner* Wapiti

Sistema Testado	bsql	sqli	Xss	bkp	lfil	cgi	LFI	Tempo
CryptOMG/clg1/			X		*	*	*	2 min

CryptOMG/clg4/		X	X		*	*	*	1 min
Vicnum		X	X		*	*	*	5 min
Wivet					*	*	*	1 min
Webcal					*	*	*	10 min

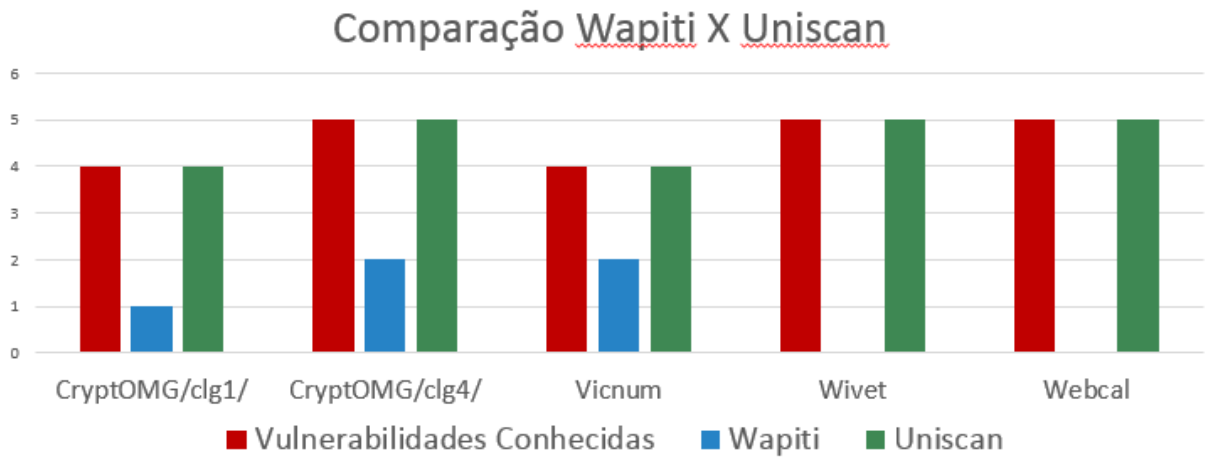
* Teste não suportado pelo *scanner*

Tabela 11- Resultados do *Scanner* Uniscan

Sistema Testado	**bsql**	**sqli**	**Xss**	**bkp**	**lfil**	**cgi**	**LFI**	**Tempo**
CryptOMG/clg1/		X	X		X		X	6 min
CryptOMG/clg4/	X	X	X		X	X	X	13 min
Vicnum	X	X	X		X		X	35 min
Wivet	X	X	X	X	X		X	16 min
Webcal	X	X	X	X	X		X	10 min

Como pode ser visto nas tabelas 10 e 11, o Uniscan encontra uma quantidade maior de vulnerabilidades quando comparado com o Wapiti, porém leva mais tempo para terminar o escaneamento. A vulnerabilidade mais comum entre os sistemas foi a *Cross-Site Scripting* (XSS) e a Inclusão de arquivos locais (LFI). O Uniscan cumpre bem seu papel em apontar as falhas do tipo de LFI conforme foi designado e também apresenta um ótimo desempenho para encontrar falhas do tipo XSS. O *scanner* Wapiti mostrou ser uma solução rápida para indicar se o sistema possui falhas ou não, sendo assim necessária uma outra ferramenta para ter um diagnóstico mais seguro. Figura 13, mostram o desempenho dos scanners comparando o número de tipos de vulnerabilidades conhecidas com as encontradas.

Figura 13 - Comparação de vulnerabilidades encontradas



5.6. Elasticidade do sistema

O sistema Safeweb possui suporte para adicionar novos scanners e utilizar múltiplas Máquinas Virtuais contendo as ferramentas de segurança. Para a instalação dos novos scanners ou máquinas virtuais, é necessário cadastrar os parâmetros nas tabelas do banco de dados que o sistema se encarrega de incorporar dinamicamente os scanners e máquinas adicionais.

Suporte a múltiplos scanners

O suporte para novos scanners garante a elasticidade em relação ao sistema não se tornar obsoleto com o lançamento de novas ferramentas de segurança mais avançadas.

A execução de novas ferramentas é feita de maneira dinâmica pelo Daemon. Para isso, é necessário que as ferramentas estejam devidamente instaladas e configuradas na máquina, e que os parâmetros necessários para a execução das ferramentas estejam cadastradas no banco de dados do sistema SafeWeb conforme visto na tabela 12.

Tabela 12 - Tabela *scanners*

id	*nome*	*descricao*	*caminho*	*param*	*parser*	*relatorio*	*preco*
1	*Uniscan*	*Meu primeiro scanner*	*./uniscan.pl -u*	-qweds	/opt/lampp/htdocs/safeweb/format/htmltobd.php	*-1*	*10,00*
2	*Wapiti*	*Meu segundo scanner*	*wapiti*	*-n 10 -b folder -u -v 1 -f html -o relatorios/wapiti/*	*/opt/lampp/htdocs/safeweb/format/jsontobd.php*	*relatorios/wapiti/vulnerabilities.json*	*1,00*

Os campos *nome* e *descricao* guardam as informações de como o *scanner* será descrito no sistema Web do SafeWeb. O campo *caminho* deve conter todo o prefixo do comando antes da *URL* necessário para executar o *scanner*, e o campo *param*, todo o resto do comando que vier depois da *URL*.

Exemplo:

> Se para executar o Wapiti no terminal do Linux, o comando seria:
>
> *# wapiti www.meusite.com –n 10 –b folder –u –v 1 –f html*
>
> No banco de dados, as colunas seguintes deveriam conter:
>
> **caminho:** *wapiti*
> **param:** *-n 10 –b folder –u –v 1 –f html*

O campo *parser* deve conter o caminho do arquivo PHP do *parser* necessário para formatar o relatório do *scanner*, e o campo *relatorio* o caminho do relatório gerado pelo sistema caso ele seja um arquivo de nome estático. Se o arquivo é gerado com o nome do site escaneado, é necessário utilizar a flag *-1*.

Com as informações dos *scanners* devidamente inseridas no banco de dados, o Web Service *scanner.php* recupera a informação das ferramentas instaladas, bem como os parâmetros necessários para a execução e envia os dados para o *Daemon* que constrói a chamada de sistema necessária para executar a ferramenta escolhida.

Suporte a múltiplas Máquinas Virtuais

Alguns *scanners* mais complexos podem necessitar de mais poder computacional e demorar horas para a execução de seus testes. Oferecer suporte para a utilização de várias máquinas virtuais e executar as ferramentas de segurança separadamente permite o controle da configuração e da quantidade de máquinas que serão necessárias para manter o sistema rápido e eficiente.

Para adicionar uma nova máquina virtual é necessário inserir uma nova linha na tabela *maquinas* contendo o endereço IP e o seu nome conforme pode ser visto na tabela 13.

Tabela 13 - Tabela máquinas virtuais instaladas

id	***nome***	***endereco***	***ativa***
1	*VM_1*	*unix:///var/run/mysocket.sock*	*0*
2	*VM_2*	*200.120.214.234:50*	*1*

O campo *ativa* serve para controlar se a máquina está ou não sendo utilizada. Se o campo estiver com o número *0*, significa que a máquina está ociosa e pronta para ser utilizada, caso este campo esteja setado em *1,* o sistema percorre a lista de máquinas até encontrar uma máquina disponível. Caso não haja, o pedido fica registrado para ser submetido posteriormente.

Criando múltiplas instâncias EC2 na Amazon

O Amazon Elastic Compute Cloud (Amazon EC2) é um serviço baseado na *web* que permite a negócios executarem programas de aplicação em ambiente de computação AWS. O EC2 pode servir a um conjunto ilimitado de máquinas virtuais, Amazon (2014).

O Amazon EC permite que sejam instanciadas várias máquinas virtuais com desempenho e recursos configuráveis de acordo com a necessidade dos clientes. Com o conhecimento dos recursos necessários para a execução de cada *scanner* é possível otimizar os custos relacionados a hospedagem de máquinas virtuais.

Para criar um Cliente usando o SDK para Java e poder gerenciar os recursos EC2, seguem os passos:

- Criar e inicializar uma instância *AWSCredentials* e especificar o arquivo de propriedades que você criou.

```
AWSCredentials credentials = new PropertiesCredentials(
        AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));
```

- Use o objeto AWSCredentials para criar uma nova instância AmazonEC2Client como segue:

```
amazonEC2Client = new AmazonEC2Client(credentials);
```

- Por padrão, o endereço do serviço é ec2.us-east-1.amazonaws.com. Para especificar um endereço diferente, use o método setEndpoint. Exemplo:

```
amazonEC2Client.setEndpoint("ec2.us-west-2.amazonaws.com");
```

Para criar múltiplas instâncias EC2 na Amazon, seguem os passos:

- Criar e inicializar uma instância RunInstancesRequest.

```
RunInstancesRequest runInstancesRequest =
          new RunInstancesRequest();

  runInstancesRequest.withImageId("ami-4b814f22")
                     .withInstanceType("m1.small")
                     .withMinCount(1)
                     .withMaxCount(1)
                     .withKeyName("my-key-pair")
                     .withSecurityGroups("my-security-group");
```

withImageId: O ID da AMI (*Amazon Machine Image*)
withInstanceType: O tipo de instância compatível com a AMI contratada.
withMinCount: Número mínimo de instâncias EC2 que serão inicializadas.

withMaxCount: Número máximo de instâncias EC2 que serão inicializadas.

withKeyName: A chave necessária para conectar à Máquina.

withSecurityGroups: Lista de grupos de segurança.

- Iniciar a instância passando o objeto *Request* para o método *runInstances.* O método retorna um objeto *RunInstancesResult*. Como segue:

```
RunInstancesResult runInstancesResult =
          amazonEC2Client.runInstances(runInstancesRequest);
```

Quando a instância estiver executando, a conexão é feita utilizando a key pair.

5.7. Trabalhos Futuros

Com o objetivo de aperfeiçoar o sistema SafeWeb desenvolvido neste trabalho destacam-se os seguintes desafios:

Medição de recursos

Pelo fato de existirem *scanners* mais complexos que outros, alguns demandam mais recursos para realizar os testes de segurança, medir a quantidade de recursos utilizados pelos *scanners* é vital para que não haja desperdícios. Verificar a quantidade necessária para executar um determinado *scanner* e configurar uma máquina virtual especialmente para esse *scanner*, implicará numa utilização eficiente dos recursos.

Virtualização Amazon EC2

Importar o sistema para os serviços de computação em nuvem como o Amazon EC2 para aproveitar a elasticidade fornecida pela infraestrutura do serviço. Criar uma instância de uma máquina virtual nova para a execução de um teste de segurança conforme as necessidades do sistema irá garantir ainda mais a utilização eficiente de recursos e diminuir os custos envolvidos na hospedagem do sistema na internet.

Firewalls e Proxy Web

Uma vez que os *scanners* fazem inúmeras requisições no sistema alvo do cliente durante os testes de vulnerabilidades, *firewalls* e *Proxys* podem bloquear as tentativas do *scanner* se comunicar com o sistema. Solicitar ao administrador do sistema web alvo dos *scanners* que sejam desabilitados os *firewalls*, garante que os testes serão realizados de maneira correta.

Plano de negócios

Criar um plano de negócios que considere o tempo de CPU, consumo de recursos de memória e rede para calcular o preço do escaneamento. O usuário requisita o escaneamento, que após terminado apresenta o preço e só da permissão para o usuário ver o relatório ser ele pagar. Especificar um custo mínimo para o escaneamento e somar com o tempo necessário para executar os scanners e recursos usados. Uma ideia para calcular o preço do escaneamento pode ser expressa pela seguinte fórmula:

*Preço_mínimo + (m1*tempo_de_cpu) + (m2*consumo_de_rede)*

Onde m1 e m2 são os multiplicadores de preços definidos conforme os preços envolvidos nos custos e a margem de lucro que se deseja obter.

6. CONCLUSÃO

Visto que o ambiente de sistemas Web é uma área onde os riscos de segurança são elevados, existe a possibilidade de pessoas mal intencionadas explorarem estas falhas de sistemas para comprometer informações e serviços. Para encontrar e corrigir estas falhas são necessárias infraestrutura e profissionais de segurança, porém, pequenas e médias empresas não dispõem de recursos para investir nesta área. Este cenário representa um nicho de mercado para negócios que ofereçam serviços de segurança sob demanda.

Este trabalho apresentou uma arquitetura integrada e elástica de sistemas, capaz de fornecer um serviço de segurança Web sob demanda, possibilitando que pequenas e médias empresas incapazes de manter uma equipe de TI, ainda possam estar com seus sistemas Web seguros a um preço acessível, enquanto investem no seu próprio negócio.

O SafeWeb foi desenvolvido para suportar a adição de novos *scanners*, o que colabora para tornar o sistema flexível e dinâmico. Também foi programado para suportar várias máquinas virtuais contendo ferramentas de segurança. Essa possibilidade permite elasticidade no sistema para controlar a quantidade de máquinas sob demanda.

Os resultados obtidos demonstraram que é possível prover scanners de vulnerabilidades sob demanda. Para avaliação da arquitetura proposta foram utilizados dois s*canners* de vulnerabilidades (Uniscan e Wapiti) e cinco sistemas Web vulneráveis. Cada scanner obteve resultados diferentes, o que reforça a ideia de utilizar múltiplos *scanners* faz-se necessária para atingir resultados de análise de segurança mais completos e precisos.

Como trabalhos futuros e desafios podem ser destacados: a virtualização em nuvem utilizando o serviço da Amazon EC2, a medição dos recursos utilizados por cada *scanner* e o desenvolvimento de um plano de negócios.

REFERÊNCIAS BIBLIOGRÁFICAS

ABEYSINGHE, S.; **RESTful PHP Web Services.** Ed. Packt, 2008.

ABOULNAGA, A. et al. **Deploying Database Appliances in the Cloud**. Artigo Técnico. Toronto, 2009.

ALBUQUERQUE, F. **Banco de Dados.** 2003. 26 f. Apostila. Universidade de Brasília, Brasília, 2003.

ALMEIDA, F. D. D. **Análise de Contratos Digitais de Software e Serviços Online: Da Complexidade a Simplicidade?** 2013. 165 f. Trabalho de Conclusão de Curso **–** Universidade Federal do Pampa, Alegrete, 2013.

Amazon. 2015 Disponível em: <http://aws.amazon.com/pt/ec2/getting-started/>. Acesso em: 08 janeiro de 2015

BACELAR, E. et al. **O Modelo de Aplicação em Nuvem e a Sua Aplicabilidade.** Artigo Técnico. Rio Grande, 2012.

Bau, J.; Bursztein, E.; Gupta, D.; Mitchell, J., **State of the Art: Automated Black-Box Web Application Vulnerability Testing.** 2010. IEEE Symposium on , vol., no., pp.332,345, 16-19 Maio 2010.

Cloud Vertical. 2015. Disponível em: <https://www.cloudvertical.com/cloud-costs#cloud_costs/index>. Acesso em: 02 janeiro de 2015.

COUTINHO, E. F. et al. **Elasticidade em Computação em Nuvem: Uma Abordagem Sistemática**. 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. 2013. 44 p.

CHEN, S.; HUANG, J.; GONG, Y. **Static Testing as a Service on Cloud.** 27º International Conference in Advanced Information Networking and Applications Workshop . 2013. 5 p.

DATA Breach Investigations Report. Colombia, 2003, 63 p.

Dimension Data Cloud. 2015. Disponível em: <http://nacloud.dimensiondata.com/Cloud-Now/Pricing>. Acesso em: 02 janeiro de 2015.

FIELDING, R. T. **Architectural Styles and theof Network-Based Software Architectures.** 2000. 180 f. Dissertação (Mestrado em Informática) **–** University of California, Irvine, 2000.

HARA, C. S. **Sistema de Gerenciamento de Dados em Nuvem.** 2011. 47 f. Apostila. Universidade Federal do Paraná, Curitiba, 2011.

INTERNET Security Threat Report. , Mountain View, 2013, 58 p.

MARCON JR, A. et al. **Aspectos de Segurança e Privacidade em Ambientes de Computação em Nuvem.** 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. 2013. 44 p.

MARTINS, V. M. M. **Integração de Sistemas de Informação: Perspectivas, Normas e Abordagens.** 2005. 218 f. Tese (Doutorado) **–** Universidade do Minho Guimarães, Braga, 2005.

MELL, P.; GRANCE, T.. **The Nist Definition of Cloud Computing.** Artigo Técnico. Gaithersburg, 2011.

MÜLLER, E. M. **Estudo e Implementação de Autenticação no Acesso Para o Portal do Husm.** 2007. 58 f. Trabalho de Conclusão de Curso **–** Universidade Federal de Santa Maria, Santa Maria, 2007.

RICHARDSON, L.; RUBY, S. **Restfull Web Services.** Sebastopol: Ed. Safari, 2007.

Rocha, D.; Kreutz, D., **A free and extensible tool to detect vulnerabilities in Web systems**, Alegrete, 2012

TAKAI, O. K.; ITALIANO, I. C.; FERREIRA, J. E. **Introdução a Banco de Dados.** 2005. 104 f. Apostila. Universidade de São Paulo, São Paulo, 2005.

VELTE, T.; VELTE, A.; ELSENPETER, R. **Cloud Computing, A Practical Approach.** Ed. Mc Grall Hill Professional, 2010.

ZAVALIK, C. **Integração de Sistemas de Informação Através de Web Services.** 2004. 72 f. Dissertação (Metrado em Informática) **–** Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.