VULNERABILIDADES EM APLICAÇÕES *WEB*

Fernanda Ramos de Carvalho
Germano Estevam Simão Pereira
Lucyara Silva Ribeiro
Tulio Cesar Lopes Alves

RESUMO

Atualmente, grande parte das aplicações são *online* e são denominadas como aplicações *Web*. O grande motivo disso é a expansão e a facilidade do acesso a *Internet*, por meio de computadores, *smartphones*, *tablets*, etc. Por causa desta facilidade o risco de invasões e ataques são maiores. Quanto mais aplicações na rede, maiores as chances de ataques bem sucedidos. Já existem centenas de vulnerabilidades conhecidas, porém o assunto ainda não é tratado com a devida atenção; talvez pela falta de conhecimento das consequências que essas vulnerabilidades trazem. A indiferença em relação ao assunto e o aumento crescente dos riscos aos usuários foram os principais motivos para a escolha do tema, visando salientar a necessidade e a importância em criar aplicações mais seguras. Através de pesquisas na área, foram analisadas as mais conhecidas vulnerabilidades em aplicações *Web*, trazendo maiores detalhes de suas consequências e possíveis correções. Foram estudadas também algumas ferramentas que podem auxiliar na segurança através de um escaneamento completo da aplicação e identificação das vulnerabilidades encontradas. Um ataque bem sucedido em uma aplicação pode trazer grandes prejuízos, físicos ou morais aos usuários e as empresas, já que a gravidade desse ataque dependerá especialmente da experiência do atacante. As ferramentas mostraram-se bastante eficientes, e como resultado dos estudos e testes, foi possível definir a melhor ferramenta, levando em consideração os recursos oferecidos, a flexibilidade, a facilidade, e os resultados obtidos no escaneamento.

Palavras Chaves: Aplicações *Web*. Vulnerabilidades. *Scanners*.

ABSTRACT

Nowadays, most applications are *online* and they are alled as Web applications. The big reason is the expansion and the ease of Internet access through computers, smartphones, tablets, etc. Because of this ease the risk of intrusions and attacks are higher. More and more applications on the network, the greater chances of successful attacks. There are already hundreds of known vulnerabilities, but it is still not treated with due attention, perhaps by lack of knowledge of the consequences that bring these vulnerabilities. The indifference to the subject and increasing risk users were the main reasons for choosing the topic, aiming to highlight the need and importance of creating more secure applications . Through research in the area were analyzed and the most known vulnerabilities in Web applications, bringing details of its consequences and possible corrections . Some tools that can help in security through a complete scanning application and identification of vulnerabilities found were also studied . A successful attack on an application can bring great harm, whether physical or moral users and businesses since the severity of the attack depends on the particular experience of the attacker. The tools used were quite efficient , and as a result of studies and tests , it was possible to define the best tool, taking into account the resources available , the flexibility, the ease and the results obtained in scanning.

Key words: Web Application. Vulnerabilities. *Scanners* .

1 INTRODUÇÃO

A *Internet* já se tornou o maior meio de comunicação utilizado no mundo, nela trafegam dados e informações extremamente importantes e em muitos casos sigilosas. Pessoas dos mais diversos setores, tais como setores financeiros e bancários acessam todos os dias um sistema *Web*, onde são armazenados dados de clientes, senhas, documentações, cartões de crédito, saldo, entre outros. O grande problema é que muitas vezes esses sistemas, nomeados como aplicações *Web*, são totalmente vulneráveis, devido, em sua maioria, as tecnologias adotadas na concepção.

De acordo com pesquisas feitas em 2008 pela *Web Application Security Consortium* (WASC), 12.186 aplicações estavam vulneráveis, resultando num total de 97.554 vulnerabilidades de diferentes riscos. Diante dessas informações torna-se imprescindível que desenvolvedores e empresas fiquem atentos á suas politicas de

segurança, aperfeiçoando e aprimorando os meios de segurança adotados no desenvolvimento dessas aplicações.

Os pontos de vulnerabilidades destes sistemas são os principais alvos, e quando são explorados pode se perder toda a confiabilidade do sistema, e arquivos que deveriam ser acessados apenas por pessoas autorizadas ficam expostos a pessoas em sua maioria, mal intencionadas. Com isso a segurança da informação visa garantir que os dados estejam protegidos, garantindo a integridade, confiabilidade e disponibilidade dos mesmos.

2 REFERENCIAL TEÓRICO

Um sistema de informação é todo e qualquer sistema usado para processar informações. Dessa forma, ele trabalha com dados de usuários, de organizações ou empresas, sendo fundamental proteger esses dados que serão armazenados no sistema. A segurança computacional é então, a área de estudos e conhecimentos que trata da proteção dos dados e informações que estarão contidas nesses sistemas (RESS, 2011).

Segundo Nakamura e Geus (2007), a segurança de redes é uma parte essencial para a proteção da informação, porém uma boa estratégia que deve ser levada em consideração são os aspectos humanos e processuais de uma organização. Isso é importante por que outros métodos de ataques, além de tecnológicos, afetam os níveis de segurança de uma organização.

2.1 Segurança Computacional

O objetivo da Segurança Computacional é proteger um sistema contra ameaças à confidencialidade, à integridade, à disponibilidade, à autenticidade e ao não repudio de informações e recursos. Se um sistema atente estas cinco propriedades, então é considerado seguro (COMER, 2007).

Ress, 2011, define cada uma dessas propriedades:

- A confidencialidade garante a manutenção do sigilo das informações ou recursos. Uma violação ocorre quando a informação ou recurso são revelados sem autorização.
- A integridade garante a confiabilidade da informação, ou seja, garante que um dado não foi modificado sem autorização. Viola-se a integridade quando os dados são modificados gerando resultados errôneos e incorretos;

- A disponibilidade é a capacidade de acesso e uso da informação ou do recurso, ela garante que a informação estará disponível no momento em que necessitar usá-la;
- A autenticidade garante que a pessoa que está fazendo contato é quem realmente afirma ser. Esse processo envolve a confirmação da identidade de um usuário ou sistema;
- O não repúdio garante que um autor não seja capaz de negar a autoria de informações que ele tenha criado, ou seja, garante que um usuário ou um sistema realmente realizou uma operação em um sistema de informação, excluindo a existência de dúvidas sobre tal procedimento.

De acordo com Nobre (2007), uma ameaça consiste em uma possível violação de um sistema computacional e pode ser acidental ou intencional, de forma que uma ameaça acidental é aquela que não foi intencionada, podendo ser, por exemplo, uma falha de *hardware* ou no *software*. Já uma ameaça intencional, como o nome diz, está associada à intencionalidade premeditada.

O termo genérico utilizado para quem realiza o ataque em um sistema computacional é *hacker*. Porém, essa generalização tem diversas ramificações, pois os ataques aos sistemas apresentam objetivos diferentes e o seu sucesso depende do grau de segurança dos alvos e da capacidade do próprio *hacker* (NAKAMURA; GEUS, 2007).

2.2 Mecanismos e ferramentas de Segurança

Para evitar os ataques a esses sistemas, existem os mecanismos de segurança, que são qualquer processo ou dispositivo projetado para detectar, impedir ou permitir a recuperação de um ataque à segurança de um sistema de informação. Alguns exemplos de mecanismos de segurança são cifragem, assinatura digital, controle de acesso, integridade de dados, troca de informações de autenticação, preenchimento de tráfego e controle de roteamento.

Como nos dias de hoje, a *Internet* tornou-se algo indispensável para qualquer organização, a segurança na *Internet* é certamente um tópico muito importante a ser tratado. Após algumas analises é possível identificar que a segurança na *Internet* não é muito diferente de outras formas de segurança, pois, assim como no mundo real, a conexão com o meio virtual necessita de proteção e controle de acesso (NAKAMURA; GEUS, 2007; CHESWICK; BELLOVIN; RUBIN, 2005).

Dentre os meios de garantir a proteção dos dados está a criptografia, uma ciência que tem papel fundamental para a segurança da informação. Propriedades como sigilo, integridade, autenticação e não repudio garantem a comunicação e armazenamento seguro. Cifragem é o processo de disfarçar a mensagem original de modo que sua substância seja escondida em uma mensagem cifrada, o a decifragem é o processo de transformar uma mensagem cifrada de volta em texto claro, a mensagem original. Estes processos são realizados por meio de algoritmos matemáticos (NAKAMURA; GEUS, 2007).

Outra ferramenta de segurança utilizado é o *firewall*. É uma espécie de barreira na qual todo e qualquer tipo de tráfego precisa passar por ela. É um ponto entre duas redes, onde a política de segurança define qual o tráfego tem permissão ou não para seguir, isso possibilita o controle de acesso, a autenticação e o registro do tráfego por meio de *logs* (STALLINGS, 2008; NAKAMURA; GEUS, 2007).

O *Intrusion Detection System* (IDS), também faz parte das ferramentas utilizadas para garantir a segurança, segundo Cheswick, Bellovin e Rubin (2005), são os como farejadores do tráfego de rede. Eles analisam e reúnem os pacotes em fluxos de dados, por isso devem ser instalados em lugares estratégicos, configurados corretamente e monitorados constantemente. Um IDS é parte fundamental da segurança de um sistema ou rede, sua capacidade de detecção de intrusões auxilia na proteção do ambiente. Seu objetivo é detectar atividades anormais ou anômalas no sistema, que podem ser ataques originados de portas legítimas permitidas pelo *firewall*, portanto, não serão detectadas por ele.

As *Virtual Private Network* (VPN), também utilizadas como ferramentas de segurança, são redes de computadores se comunicando através de um meio de comunicação público, geralmente a *Internet*. Porém de forma segura, utilizando a criptografia como meio de proteger seus dados. O termo virtual refere-se ao fato destas redes estarem fisicamente separadas, muitas vezes distantes. As VPNs são muito utilizadas por empresas na comunicação entre matriz e filiais, devido ao fato de ter um custo financeiro menor, se comparado com a utilização de *links* dedicados.

2.3 Segurança em Aplicações *Web*

A segurança em aplicações *Web* trata-se da segurança do código de implementação da aplicação, segurança de bibliotecas e segurança também nos próprios servidores *Web*. A probabilidade de sucesso de um ataque a uma aplicação *Web* são realmente elevadas devido à facilidade de exploração das vulnerabilidades, facilidade

no acesso a ferramentas de exploração dessas vulnerabilidades e a grande quantidade se programadores *Web* despreocupados com a segurança.

Devido a esses fatores, surgiram então organizações e fóruns como: (1) *Open Web Application Security Project* (OWASP), (2) *Web Application Security Consortium* (WASC), (3) *Common Weakness Enumeration* (CWE) e (4) *Software Assurance Forum for Excellence in Code* (SAFECode). Essas entidades são voltadas a segurança dessas aplicações, e possuem como objetivo auxiliar desenvolvedores, empresas e demais organizações interessadas na segurança de seus *softwares* aplicativos.

- *Open Web Application Security Project* (OWASP)

A OWASP é uma organização livre e sem fins lucrativos que foi estabelecida em 2001, voltada principalmente para a área de segurança em *softwares* aplicativos. É reconhecida internacionalmente e livre de pressões comerciais. Conta com colaboradores e voluntários na criação de seus projetos, documentações e ferramentas, que são disponibilizadas livre e gratuitamente para toda comunidade internacional. É suportada por empresas e indivíduos voluntários através de patrocínios financeiros para seus projetos.

- *Web Application Security Consortium* (WASC)

Fundada por Jeremiah Grossman, a WASC é uma entidade sem fins lucrativos, formada por um grupo de especialistas, profissionais da área de segurança e representante de organizações desenvolvedoras de *softwares open sources*. É voltada para a área de segurança em aplicações *Web* e contribui para que empresas, instituições de ensino, desenvolvedores e governos possam melhorar seus *softwares* aplicativos.

- *Common Weakness Enumeration* (CWE)

A *Common Weakness Enumeration* (CWE) é uma listagem criada pela Mitre, uma empresa sem fins lucrativos que atua como *Federally Funded Research and Development Centers* (FFRDC), ou seja, um centro de pesquisa e desenvolvimento financiado pelo governo, neste caso, o Estados Unidos da América. Essa lista, lançada em 2005, enumera as vulnerabilidades e fraquezas encontradas no desenvolvimento de *softwares*, incluindo aplicações *Web*. A lista já possui cerca de 940 vulnerabilidades listadas.

- *Software Assurance Forum for Excellence in Code* (SAFECode)

A SAFECode é uma organização sem fins lucrativos voltada exclusivamente para a área de segurança em *softwares*, *hardwares* e serviços. *Software*

Assurance é um conjunto de métodos e processos com o objetivo de garantir que os *softwares* possam desempenhar as funções que lhes são designadas, sem possuírem vulnerabilidades, códigos maliciosos ou defeitos que possam trazer prejuízos a usuários finais. Nesse fórum, empresas desenvolvedoras trocam informações e experiências sobre as melhores práticas de desenvolvimento de código seguro.

2.3.1 Vulnerabilidades em Aplicações *Web*

Vulnerabilidades são falhas encontradas em *softwares*, como resultado de possíveis erros em projetos, implementação ou configurações do mesmo. Essas falhas, quando exploradas por atacantes, resultam em uma violação da segurança. Geralmente essas vulnerabilidades são geradas devido a falta de mão de obra qualificada, equipes sem a competência adequada e falta de profissionais com conhecimento em segurança (Franzini, 2009). A seguir, serão tratadas as 10 principais vulnerabilidades encontradas em aplicações *Web*, extraídas dos sites e documentações das organizações e fóruns citados acima.

- Injeção de Código

A injeção de falhas consiste na introdução de dados não confiáveis ou maliciosos que são enviados a uma aplicação como parte de um comando ou consulta. O que coloca essa vulnerabilidade como a mais explorada, segundo a OWASP, é o fato de ser facilmente detectada por *scanners* e *fuzzers*, ajudando os atacantes a encontrarem não só a falha em si, como todos os detalhes de onde ela se encontra e como podem ser exploradas.

- Quebra de Autenticação e Gerenciamento de Sessão

Nesse tipo de vulnerabilidade, atacantes se aproveitam de vazamentos ou falhas nas funções de gerenciamento de autenticação ou sessão, como contas expostas, senhas ou IDs de sessão. Desenvolvedores geralmente programam sistemas de autenticação personalizados e sistemas de gestão de sessão, porém na maioria das vezes não o fazem corretamente. Os resultados frequentemente são falhas em áreas como *logout*, gerenciamento de senhas, tempo limite, lembrar senha, pergunta secreta, etc.

- *Cross-Site Scripting*

Cross-Site Scripting é uma técnica de ataque em que o atacante utiliza uma aplicação *Web* para enviar códigos maliciosos, geralmente na forma de scripts do lado

do browser, para o usuário final. Falhas que permitem esse tipo de ataque são fáceis de serem detectadas e bastante generalizadas, podendo ocorrer em qualquer lugar de uma aplicação que utilizada os dados de um usuário, digitados em uma página de entrada para gerar uma saída, sem validar ou codificar essa entrada.

Atacantes podem utilizar essas falhas para roubar sessões de usuário da vítima, desfigurar sites, inserir conteúdos maliciosos, redirecionar usuários, sequestrar o navegador da vítima usando o *malwares*, etc.

- Referência Insegura e Direta a Objetos

Nesse tipo de ataque, um usuário autorizado do sistema muda o valor do parâmetro que se refere diretamente a um objeto do sistema para outro objeto que o usuário não está autorizado, tentando obter acesso. Aplicações *Web* frequentemente utilizam o nome ou a chave de um objeto para gerar páginas *Web*, porém essas aplicações nem sempre verificam se o usuário está autorizado para acessar o objeto de destino.

- Configuração Incorreta de Segurança

Nesse tipo de vulnerabilidade, o atacante acessa contas padrões, páginas não utilizadas, falhas não corrigidas, arquivos e diretórios não protegidos, etc. Seu objetivo é obter acesso não autorizado ou conhecimento do sistema. Falhas em configurações de segurança podem ocorrer em qualquer nível da aplicação, incluindo a plataforma, servidor *Web*, servidor de aplicação, banco de dados, estrutura e código personalizado.

- Exposição de Dados Sensíveis

Dados sensíveis são dados confidenciais que não devem ficar expostos como, dados pessoais, cartões de crédito, registro de saúde, credenciais, etc. Nesse tipo de vulnerabilidade, os atacantes não quebram a criptografia diretamente, eles procuram outros meios, tais como roubar chaves, fazer ataques do tipo *man in the middle*, roubar dados ou textos apagados do servidor, enquanto estão navegando ou a partir do navegador do usuário. A falha mais comum nesse tipo de vulnerabilidade é simplesmente não criptografar dados sensíveis ou quando a criptografia é utilizada a geração de chaves e o algoritmo utilizado são fracos.

- Falta de Função para Controle do Nível de Acesso

A aplicação não executa, ou executa incorretamente a verificação de autorização quando um usuário tenta acessar algum recurso. Assim o atacante, um

usuário autorizado do sistema, muda a URL ou um parâmetro de uma função privilegiada, com a intenção de obter acesso. Com isso usuários anônimos podem acessar funções privadas não protegidas. Aplicações *Web* nem sempre protegem adequadamente as funções do aplicativo. Muitas vezes o nível de proteção da função é gerenciado através de configuração, e o sistema pode estar configurado incorretamente.

- *Cross-Site Request Forgery*

O *Cross-Site Request Forgery* (XSRF) é um ataque que ocorre quando um *site* malicioso, *e-mail*, mensagem instantânea ou programa faz com que o navegador *Web* da vítima execute uma ação indesejada em um *site* confiável que o usuário esteja autenticado. O impacto desse ataque está diretamente ligado aos recursos expostos pela aplicação vulnerável. Em alguns casos, pode resultar em transferências de fundos, alteração de senhas ou o atacante pode comprar algum item no contexto do usuário.

- Utilização de Componentes Vulneráveis Conhecidos

Alguns componentes vulneráveis, como por exemplo, bibliotecas ou *frameworks*, podem ser identificadas por atacantes através da análise manual. Ele personaliza um *exploit*, que é um *software*, um pacote ou sequência de dados que se aproveita das vulnerabilidades da aplicação para invadi-la (Prada, 2008). Esse *exploit* é personalizado de acordo com a necessidade do atacante.

Quase todas as aplicações possuem essa vulnerabilidade, devido a maioria das equipes de desenvolvimento não se preocuparem em garantir que seus componentes e bibliotecas estejam atualizadas.

- Redirecionamentos e Encaminhamentos Inválidos

Uma falha de redirecionamento é quando um aplicativo utiliza um parâmetro e redireciona o usuário para o valor do parâmetro sem qualquer validação. Essa vulnerabilidade é usada em ataques de *phishing* para forçar usuários a visitar *sites* maliciosos sem perceber. Atacantes utilizam *links* com redirecionamento não validado e truques para que as vítimas o acessem.

3 MATERIAL E MÉTODOS

Primeiramente, será feito pesquisas em livros, revistas e *Internet* para o levantamento de informações como: elementos da segurança computacional,

mecanismos de segurança e os principais ataques a sistemas e redes, posteriormente uma pesquisa mais avançada nos permitirá obter informações detalhadas de ataques focados em aplicações *Web*.

A documentação será feita utilizando o *Microsoft Word 2013*, onde iremos descrever todas as informações obtidas com pesquisas e estudos.

A partir destas informações pretende-se escanear uma página *Web* com ao menos três *scanners open sources* para uma análise sobre os resultados obtidos por eles. Posteriormente a página *Web* terá suas vulnerabilidades corrigidas e um novo escaneamento será feito.

Os *scanners* utilizados serão: (1) *Skipfish*, (2) *Zed Attack Proxy* (ZAP) e (3) *Iron Web Application Advanced Security Testing Platform* (IronWASP).

4 DESENVOLVIMENTO

4.1 *Scanners* de Vulnerabilidades

Scanners de vulnerabilidades são ferramentas automatizadas que fazem varreduras em aplicações *Web* a fim de detectar possíveis falhas de segurança, como XSS, SQL *Injection*, configurações inseguras, etc. Até 2012, ao menos 60 *scanners* de vulnerabilidades foram desenvolvidos, dentre as quais, 80% são *open sources* e de código aberto, onde cada ferramenta possui suas vantagens e desvantagens.

Em 2010, surgiu o projeto *Web Application Vulnerability Scanner Evaluation Project* (WAVSEP), esse projeto consiste em uma aplicação vulnerável destinada a avaliar os recursos e qualidade dos *scanners* de vulnerabilidades. Esse projeto possui diversos casos de testes e alguns falsos positivos para que se possa avaliar a precisão das ferramentas. Através desse projeto Chen (2012), realizou testes em cerca de 60 aplicações, comerciais e *open sources*.

Analisando os resultados obtidos pelos testes de Chen[1] (2012), e do projeto WAVSEP[2], foram selecionadas três ferramentas *open sources* para avaliação. A escolha dessas ferramentas levou em consideração a qualidade dos resultados obtidos nos testes e facilidade de uso. As ferramentas escolhidas foram: (1) *Skipfish*, (2) *Zed Attack Proxy* (ZAP) e (3) *Iron Web Application Advanced Security Testing Platform* (IronWASP).

---

[1] *Top 10 - O Web Application Vulnerability Scanners Benchmark*: http://sectooladdict.blogspot.co.il/2012/07/2012-web-application-scanner-benchmark.html

[2] Projeto WAVSEP: https://code.google.com/p/wavsep/

- Skipfish

Skipfish é uma ferramenta de segurança desenvolvida por Michal Zalewski, Niels Heinen e Sebastian Roschke, com licença de código *Apache* versão 2.0.

É destinada para fornecer resultados precisos e significativos. Ela possui suporte para sistemas operacionais como *Linux*, *FreeBSD*, *MacOS X* e ambientes *Windows*. Escrita em código C puro, possui apoio para uma grande variedade de *frameworks Web* e *sites* de tecnologia mista, além da capacidade de aprendizagem automática.

- Zed Attack Proxy

A *Zed Attack Proxy* (ZAP) é uma ferramenta de testes de penetração com o objetivo de encontrar vulnerabilidades *Web*. Ela oferece um conjunto de ferramentas que permitem encontrar as vulnerabilidades de forma manual ou automatizada. Foi desenvolvida por uma equipe mundial de voluntários, porém patrocinada, direta ou indiretamente, por diversas organizações, como: OWASP, *Mozilla*, *Google*, *Microsoft*, *Sálvia*, etc.

- IronWASP

A *Iron Web Application Advanced Security Testing Platform* (IronWASP), é um sistema de código aberto para testes de vulnerabilidades em aplicações *Web*. Criado por Lavakumar Kuppan, teve sua primeira versão lançada em fevereiro de 2012.

Projetado para rodar no *Windows*, ela pode ser executada em sistemas *Linux* através do *Wine* e em sistemas *MAC* através do *CrossOver*, ambos são aplicativos que possibilitam a instalação de *softwares* do *Windows* em outros Sistemas Operacionais. Dentre sua funcionalidades, a IronWASP é uma ferramenta customizável, isto é, o usuário pode criar seus próprios *scanners* de segurança.

4.2 Escaneamento

Para que fosse feito o escaneamento com as ferramentas, foi utilizado uma aplicação simples de uma locadora de filmes. Essa aplicação foi hospedada em dois servidores diferentes. A primeira versão é a aplicação original, consequentemente, com diversos erros, e foi hospedada no servidor UOL, com a seguinte URL: www.computacaounifenas**.com.br**. A segunda versão é a aplicação que foi corrigida

após o escaneamento, com grande parte dos erros removidos, e foi hospedada no servidor *Hostinger*, com a seguinte URL: www.computacaounifenas**.bl.ee**.

- Skipfish

O escaneamento da primeira versão do site feito pela *Skipfish* demorou cerca de 45 minutos, e apresentou um total de 66 erros.

Após as devidas correções, foi feito um novo escaneamento, resultando um total de apenas 9 erros.

- ZAP

Com a ferramenta ZAP, o escaneamento demorou cerca de 30 minutos, relatando um total de 30 erros.

Após a correção da aplicação a ferramenta apresentou um total de 6 erros.

É importante ressaltar que a ferramenta não foi capaz de detectar erros no servidor.

- IronWASP

O escaneamento da ferramenta IronWASP foi feito em aproximadamente 37 minutos e relatou um total de 52 erros.

O resultado após a correção dos erros foi um total de dois erros.

5 RESULTADOS

Após o escaneamento da aplicação com as ferramentas selecionadas, na primeira versão da aplicação, desenvolvida sem o cuidado com as vulnerabilidades, foram relatados um total de 142 erros. Esses erros foram analisados e, para maior parte deles, foram encontradas soluções. A aplicação foi corrigida para que um novo escaneamento pudesse ser feito. Feitas as devidas correções a aplicação foi hospedada em outro servidor. Após o escaneamento na nova versão da aplicação, versão está com o tratamento dos erros encontrados na primeira versão, foram relatados um total de 17 erros.

Com a análise dos resultados obtidos nos respectivos escaneamentos, chegou-se a um percentual de 90% de erros corrigidos. Conforme ilustra o Gráfico 1.

Gráfico 1 - Percentual de erros corrigidos.

Diante das análises das ferramentas, temos como resultado a IronWASP como melhor ferramenta de escaneamento entre as testadas. Com um número considerável de vulnerabilidades encontradas e um tempo satisfatório, além da facilidade de utilização e na análise dos resultados. A ferramenta é simples e flexível sem deixar a desejar na eficácia da mesma.

Apesar do alto desempenho obtido pela ferramenta IronWASP, conseguimos perceber que nem todos os erros foram detectados por ela, diante desse fato, constatamos que o uso conjunto de mais de uma ferramenta, consegue prover maior segurança e benefícios como um maior número de erros identificados e também uma diversidade maior deles.

6 CONCLUSÃO

A falta de segurança em aplicações *Web* não pode mais ser tratada como consequência do grande e acessível meio em que se encontram, a *Internet*. Atualmente existem diversas organizações que tratam esse assunto, auxiliando desenvolvedores e empresas com palestras e infinitas documentações, trazendo todo suporte e meios para manter uma aplicação com o menor número de riscos possíveis.

Nesse trabalho foram utilizadas três ferramentas de escaneamento, porém existem muitas outras disponíveis, tanto *open soucers*, como comerciais. A diversidade de ferramentas encontradas deixa a critério que cada desenvolvedor ou empresa possa

utilizar a ferramenta que mais se adapta a sua forma de trabalho, não havendo motivos para deixar de usa-las em seus projetos.

A correção das vulnerabilidades teve uma dificuldade média, já que existem diversos materiais disponíveis para estudos e correções desses erros, inclusive arquivos disponibilizados pelas organizações citadas nesse trabalho, o que comprova que a grande quantidade de vulnerabilidades encontradas em aplicações é resultado da falta de atenção, a despreocupação de seus desenvolvedores e até mesmo a falta de incentivo de empresas, que não auxiliam seus colaboradores e não exigem que eles construam códigos mais seguros.

Após a utilização de três *scanners open sources* foi possível compreender as principais funcionalidades e seus benefícios, definindo a melhor ferramenta; porém vale ressaltar que o uso de duas ou mais ferramentas traz diversos benefícios inclusive maior segurança. Não foi possível fazer a correção de todos os erros da aplicação, devido principalmente ao uso de um servidor externo, contudo a maioria e os de maiores riscos foram corrigidos.

Segurança em aplicações *Web* é uma área muito complexa, pois envolve desde o usuário e seu *browser*, até o extenso e vulnerável meio de comunicação que é a *Internet*. Entretanto, é indispensável que aplicações *Web* sejam bem projetadas e focadas na segurança, isto é, aplicações que realizam validações de entradas, tratamentos de erros, acessos limitados, utilização de APIs de segurança, uso correto das tecnologias, entre outros métodos descritos nesse trabalho. Um desenvolvedor ou empresa com boas políticas de segurança é essencial para evitar os prejuízos e as consequências irreversíveis.

REFERÊNCIAS

CHEN, Shay. ***Top 10: O Web Application Vulnerability Scanners Benchmark***. jul. 2012. Disponível em: <http://sectooladdict.blogspot.co.il/2012/07/2012-web-application-scanner-benchmark.html>. Acesso em: 25 set. 2013.

CHESWICK, Willian R.; BELLOVIN, Steven M.; RUBIN, Aviel D.. ***Firewall*s e segurança na *Internet*:** repelindo o hacker ardiloso. 2. ed. Porto Alegre: Bookman, 2005.

COMER, Douglas E. **Redes de computadores e *Internet***. Porto Alegre: Bookman, 2007.

COMMON WEAKNESS ENUMERATION. Disponível em: <http://cwe.mitre.org/>. Acesso em: 30 jul. 2013.

FRANZINI, Fernando. **Vulnerabilidades de Aplicativos *Web***. iMasters. 26 mar. 2009. Disponível em: <http://imasters.com.br/artigo/12132/>. Acesso em: 5 nov. 2013.

NAKAMURA, Emilio Tissato; GEUS, Paulo Licio de. **Segurança de redes em ambientes cooperativos.** São Paulo: Novatec Editora, 2007.

NOBRE, J. C. A.. **Ameaças e Ataques aos Sistemas de Informação:** Previnir e Antecipar. Cadernos UniFOA, Volta Redonda, ano 2. n. 5, dez. 2007. Disponivel em: < http://www.foa.org.br/cadernos/edicao/05/11.pdf >. Acesso em: 01 mai. 2013.

PRADA, Rodrigo. **O que é *exploit*.** 22 dez. 2008. Disponível em: <http://www.tecmundo.com.br/seguranca/1218-o-que-e-exploit-.htm>. Acesso em: 11 ago. 2013.

RESS, Weber. Começando em segurança. **MSDN.** Set. 2011. Disponível em: <http://msdn.microsoft.com/pt-br/library/ff716605.aspx>. Acesso em: 01 mai. 2013.

STALLINGS, Willian. **Criptografia e segurança de redes.** Colaboração de: Daniel Vieira, Graça Bressan, Ákio Barbosa e Marcelo Succi. 4. ed. São Paulo: Pearson Prentice Hall, 2008.

WEB APPLICATION SECURITY CONSORTIUM . Disponível em: <http://www.webappsec.org/>. Acesso em: 30 jul. 2013.