FACULDADE DE TECNOLOGIA DE AMERICANA
SEGURANÇA DA INFORMAÇÃO

OTAVIO LOPES MEDEIROS

SEGURANÇA EM APLICAÇÕES WEB

AMERICANA, SP
2016

FACULDADE DE TECNOLOGIA DE AMERICANA
SEGURANÇA DA INFORMAÇÃO

OTAVIO LOPES MEDEIROS

SEGURANÇA EM APLICAÇÕES WEB

Trabalho de conclusão de curso, desenvolvido como critério de avaliação da Disciplina de: Trabalho de Conclusão de Curso, do curso de: Segurança da Informação, na faculdade: Fatec Americana sobre a orientação (a) professor (a) orientador (a): Marcus Vinicius Lahr Giraldi.

Área de concentração: Segurança da Informação.

AMERICANA, SP
2016

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

M44s MEDEIROS, Otávio Lopes
Segurança em aplicações Web / Otávio Lopes Medeiros. – Americana: 2016.
49f.

Monografia (Curso de Tecnologia em Segurança da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Esp. Marcus Vinicius Lahr Giraldi

1. Segurança em sistemas de Informação 2. Web - rede de computadores I. GIRALDI, Marcus Vinicius Lahr II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.518.5

Otávio Lopes Medeiros

SEGURANÇA EM APLICAÇÕES WEB

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Segurança da Informação

Americana, 06 de Dezembro de 2016.

Banca Examinadora:

Marcus Vinícius Lahr Giraldi (Presidente)
Especialista
Fatec Americana

Edson Roberto Gaseta (Membro)
Especialista
Fatec Americana

Benedito Aparecido Cruz (Membro)
Graduado
Fatec Americana

AGRADECIMENTOS

Em primeiro lugar a minha família e namorada que me apoiam quando preciso.
Um agradecimento especial aos professores que nos ensinam e nos desenvolvem nessa grande jornada dos estudos.

DEDICATÓRIA

Aos meus pais pela grande dedicação que dispuseram na minha formação.

RESUMO

O presente texto apresenta conceitos sobre a Segurança da Informação e a sua adequação à Tecnologia da Informação, com foco na Segurança em Aplicações Web. Com a evolução da tecnologia, a tendência é o desenvolvimento de aplicações voltadas para web e acesso remoto. A velocidade da tecnologia cria novas formas de ataques a sistemas, principalmente aplicações *web*. Os desenvolvedores de *softwares* devem se adequar a essa nova dinâmica, com a mudança na proteção ao desenvolver o sistema. As empresas ao disponibilizarem um aplicativo, se o mesmo não foi desenvolvido de maneira segura, deixam suas informações expostas a qualquer tipo de ataque. Dessa forma, apesar de discussões, também apresentadas nesse trabalho, o programador deve seguir as recomendações de segurança, no momento da criação do sistema ou ferramenta, afim de zelar pela segurança da informação da organização que irá utilizá-la.

Palavras Chave: Segurança da Informação; Segurança em Aplicações Web; OWASP.

ABSTRACT

The present text presents concepts about Information Security and its adequacy to Information Technology, with a focus on Web Application Security. With the evolution of technology, the trend is the development of applications focused on the web and remote access. The speed of technology creates new forms of attacks on systems, mainly web applications. Software developers must adapt to this new dynamic, with the change in protection when developing the system. Companies providing an application, if it has not been developed in a secure way, leave their information exposed to any type of attack. Thus, in spite of discussions, also presented in this work, the programmer must follow the safety recommendations, now of creation of the system or tool, in order to ensure the information security of the organization that will use it.

Keywords: Information Security; Security in Web Applications; OWASP.

SUMÁRIO

1. INTRODUÇÃO

Quando se observa uma área de Tecnologia da Informação (TI) é certo que novas pesquisas e novos recursos surgem constantemente, obrigando o profissional desta área a manter-se sempre atualizado. A tecnologia de banco de dados e aplicações Web é um tema de extrema relevância, pois, apesar de sempre surgirem atualizações e recursos com maiores tecnologias de softwares e banco de dados, alguns conceitos utilizados por profissionais no quesito de desenvolvimento, fabricação e segurança de softwares e banco de dados pouco foram alterados.

Como as aplicações Web precisam ter seu código fonte no servidor para que o mesmo consiga executar processamentos retornando o resultado para a página do usuário, muitos desenvolvedores acabam não utilizando criptografia para o código fonte, facilitando e agilizando o desenvolvimento. Assim o profissional consegue fazer alterações no ambiente onde está localizada a aplicação, mas ao mesmo tempo, permite o acesso de qualquer pessoa ao código fonte, possibilitando o roubo de informações, tornando o mesmo vulnerável.

Outro agravante é no desenvolvimento e planejamento da arquitetura de um novo software. O grande erro de qualquer empresa que produza sistemas é não se preocupar com a segurança e uma arquitetura segura.

Este trabalho demonstra como mudanças no desenvolvimento de aplicações web e nos bancos de dados podem trazer benefícios às empresas quando o assunto é segurança da informação, indicando que a tecnologia surgiu para facilitar a acesso à informação e não trazer prejuízos aos usuários e possibilidade de roubo de informações.

É certo que os servidores utilizados atualmente para a hospedagem *online* de *sites* e sistemas são mais seguros, porém nenhum ambiente está totalmente seguro. Tal afirmação pode ser confirmada pela ampla utilização de diversos mecanismos de segurança, tais como o *firewall*, que visa à proteção por meio de barreiras que diminuem ou impedem a possibilidade de invasão.

O grande problema com relação à segurança de informação e riscos de invasão aos sistemas *online* se deve ao fato da inocência e inexperiência de muitos analistas e programadores de software, os quais, ao projetar e desenvolver uma aplicação não levam em conta a necessidade da proteção do seu código fonte, dando prioridade

somente ao seu visual, desempenho e propósito para o qual ele foi projetado. Apesar de a segurança ser, na maioria das vezes irrelevante para esses desenvolvedores, ela é de extrema importância em qualquer atividade praticada em ambientes *online*, pois, na sua ausência, informações podem vazar para pessoas não autorizadas, sendo que algumas delas podem agir com má-fé no uso desses dados.

Existem diversas formas de consultar e conhecer quais os tipos de ataques em aplicações *web* tem sido mais realizados na atualidade. Uma das fontes desse tipo de informação mais respeitadas na área é a OWASP, uma comunidade de desenvolvedores e especialistas em segurança da informação, voltada para a área de sistemas *on-line*.

Para dar ênfase do quão é importante a segurança em aplicações *web*, foi demostrado como é simples realizar um ataque utilizando a técnica de *SQL Injection*, tornando possível a extração qualquer informação da aplicação vulnerável.

2. SEGURANÇA DA INFORMAÇÃO

Este capítulo está organizado da seguinte forma: a Seção 2.1 aborda sobre o tema informação e apresenta um pouco de sua história. A Seção 2.2, descreve sobre a importância da informação e porque se deve preocupar com a sua proteção. Também é introduzido o que é segurança, finalizando sobre quais ferramentas e métodos garantem a proteção da informação.

2.1 INFORMAÇÃO

Informação é qualquer forma ou ato de informar algum tipo de conteúdo ou notícia. Essa informação pode ser armazenada fisicamente ou através de mídias digitais

Atualmente a informação pode ser considerada um dos mais importantes itens de uma empresa ou instituição, nela está contido todos os processos da empresa, controles e até mesmo documentos de vital importância. Mas infelizmente as organizações só se dão conta de quão importante é a informação quando o seu conteúdo é perdido ou interceptado por uma empresa concorrente. “A manipulação indevida da informação pode ocasionar prejuízos, por isso requer alguns cuidados, especialmente em relação ao meio de divulgação e armazenamento utilizado” (KLEINSORGE, 2015, p. 50).

Aborda-se a importância da informação na seguinte afirmação de Dantas (2011, p. 9):

> O mundo moderno tem dedicado especial atenção à informação, devido à sua importância para a manutenção dos negócios e a realização de novos empreendimentos entre pessoas, empresas, povos, nações e blocos econômicos.
> A boa informação abre verdadeiras oportunidades para quem a possui, o que torna o cenário dos negócios mais dinâmico e acirrado em busca de novos mercados, acordos internacionais, poder e qualidade, dentre outros, o que gera a competitividade e transforma a informação no principal elemento motriz desse ambiente altamente competitivo, que requer, assim, proteção especial.
> Em contrapartida, a ausência da informação ou a informação de má qualidade constitui uma grande ameaça, podendo levar empresas à extinção. Tudo isso atribui à informação um importante valor, transformando-a num ativo essencial aos negócios de uma organização, necessitando ser protegida. Para a sua devida proteção, é preciso compreendê-la.

Com essa real necessidade da proteção da informação, existe uma crescente preocupação com *softwares* que proporcionam uma maior segurança para a rede da empresa. “Uma das maiores falhas em segurança da informação é sugerir que apenas a tecnologia garante a solução, deixando assim, as demais áreas desprotegidas, como é o caso do “fator humano” que nem sempre recebe a devida atenção”(SANTOS; GULO, 2012, p. 2). Uma das maiores falhas que existe na área de segurança da informação é acreditar que apenas utilizando proteção de softwares a empresa estará mais segura. A maneira mais explorada por *hackers* é a engenharia social. Conforme visto "A segurança da informação de uma empresa garante, em muitos casos, a continuidade de negócio, incrementa a estabilidade e permite que as pessoas e os bens estejam seguros de ameaças e perigos." (BLUE PHOENIX, 2008).

2.2 SEGURANÇA DA INFORMAÇÃO

Segurança é se sentir mais seguro de algo ou lugar. Em razão disso as pessoas vão fazer compras em um *shopping*. Existem vários motivos para isso, um deles é a segurança. Quando se está em um *shopping* em nenhum momento você acaba se sentindo inseguro de fazer compras, ou de realizar um passeio. Porque nestes lugares tem vigias, câmeras de segurança e é um espaço fechado. A última coisa de que uma pessoa irá se preocupar em um local como esse, é um assalto.

Existem várias maneiras de se proteger uma rede empresarial. Uma das maneiras que se têm mais investimentos é a área de tecnologia. A principal preocupação é com o acesso externo para a rede interna de uma empresa. Ferramentas como o *firewall* desempenham muito bem a função de bloquear acessos indevidos. O *firewall* é um software que gerencia os acessos de uma rede, tanto para saída de pacotes, como de entrada de pacotes na rede. “O objetivo do firewall é basicamente proteger uma rede especifica das ameaças que uma rede pública não confiável pode oferecer” (SOEIRA, SILVA, TABORDA, 2013, p. 13).

Mas existem outros métodos e *softwares* que melhoram a segurança da rede interna. O Sistema Operacional deve estar sempre atualizado, pois é disponibilizado com frequência *patchs* de atualizações, corrigindo vulnerabilidades do sistema, diminuindo dessa maneira os riscos de invasões das redes

Um bom antivírus instalado também é essencial para manter a segurança e o bom funcionamento e desempenho dos computadores. De nada adianta um bom antivírus, se o mesmo não estiver atualizado. Cada vez mais surgem novos antivírus que viabilizam minimizar ou mitigar as vulnerabilidades do sistema que não se tinha conhecimento. É de suma importância deixar sempre o antivírus atualizado, para que o mesmo esteja com seu banco de dados contra vírus atualizado, e ele possa rastrear novos vírus, que possivelmente irá aparecer nos computadores.

Outra maneira de manter a rede segura é utilizar o *proxy*. Essa é uma ferramenta que fica entre o servidor e o cliente (computador do usuário). Toda requisição que o usuário realizar, passará pelo *proxy.* Seu papel é analisar o que o usuário solicitou, e verificar se o mesmo tem permissão para requisitar o conteúdo ao servidor, caso tenha acesso ele será encaminhado ao servidor, caso contrário o pacote será descartado. "*Proxy* é basicamente o intermediador do cliente e do servidor, fazendo um papel fundamental em várias atividades. Neste caso, ele vai desempenhar a função de filtragem dos dados ou bloqueios de sites que indesejados" (MAFIOLETTI, 2012, p. 20).

O foco não é deve ser somente no software ou na ferramenta, existe também o "fator humano" ou a engenharia social. Como definido pelo autor Marciano (2006), "A Tecnologia da informação é capaz de apresentar parte da solução a este problema, mas não é capaz de resolvê-lo integralmente". É necessário o investimento na gerência de pessoas, com treinamentos e conscientização de ataques sociais. Boa parte dos ataques bem-sucedidos não são realizados com uso de tecnologia ou ferramentas sofisticadas, ocorrendo o roubo de informações em razão de descuido ou despreparo de funcionários que passam informações a pessoas não autorizados, através da manipulação desses indivíduos, se passando por alguém que não são.

O livro "A arte de invadir" (Mitinick, 2006) narra várias histórias reais sobre a invasão utilizando-se de engenharia social. A invasão realizada por meio de falha humana é chamada de engenharia social, como por exemplo, um telefonema, um *e-mail*, documentos importantes jogados no lixo, documentos em cima da mesa, mensagem de voz na caixa de entrada. Esses descuidados podem ser explorados pelo atacante para extrair informações que possam ajudar a invasão do alvo desejado.

3 APLICAÇÕES

Este capítulo está organizado da seguinte forma: a Seção 3.1 introduz sobre sistemas e apresenta um pouco da sua história. A Seção 3.2, descreve sobre os Sistemas Operacionais, qual seu papel na prática e suas distribuições disponíveis. Na Seção 3.3 é introduzido o conceito de aplicações.

3.1 SISTEMAS

De acordo com Monteiro (2002, p.3):

> Um sistema pode ser definido de diferentes maneiras. Um sistema pode ser compreendido, por exemplo, como um conjunto de partes que cooperam para atingir-se um objetivo comum. Porém, a que parece mais apropriada para nossas conceituações é a seguinte: “Conjunto de partes coordenadas que concorrem para a realização de um determinado objetivo.”.

Sistema é qualquer forma, procedimento, etapas que ao serem executadas em conjunto, estarão processando dados até chegar a um resultado desejado. Sistema pode ser tanto digital quanto analógico. Sistemas digitais surgiram com a evolução das máquinas e principalmente com o surgimento do primeiro computador. Mas não existem somente sistemas digitais que são operados através de *chips* e processadores, existem máquinas e sistemas que foram inventadas muito antes do surgimento da informática, como por exemplo, o relógio. Através de engrenagens e uma bateria que gera energia e movimentos simétricos, o relógio registra em tempo real o horário.

Mais tarde surgiu uma arquitetura com circuitos integrados que é utilizada até os dias de hoje. São três elementos que formam a base da arquitetura dos computadores: processador, memória principal ou memória RAM e placa mãe que interliga esses dois componentes. A comunicação entre esses componentes é através de uma linguagem de máquina, ou seja, a linguagem binária. Os equipamentos só “entendem” a linguagem de máquina.

Na visão de Machado (2007, p.5) os sistemas,

> Um sistema computacional visto somente sob a ótica do *hardware*, ou seja, como um conjunto de circuitos eletrônicos, placas, cabos e fontes de alimentação, têm pouca utilidade. É através do *software* que serviços são

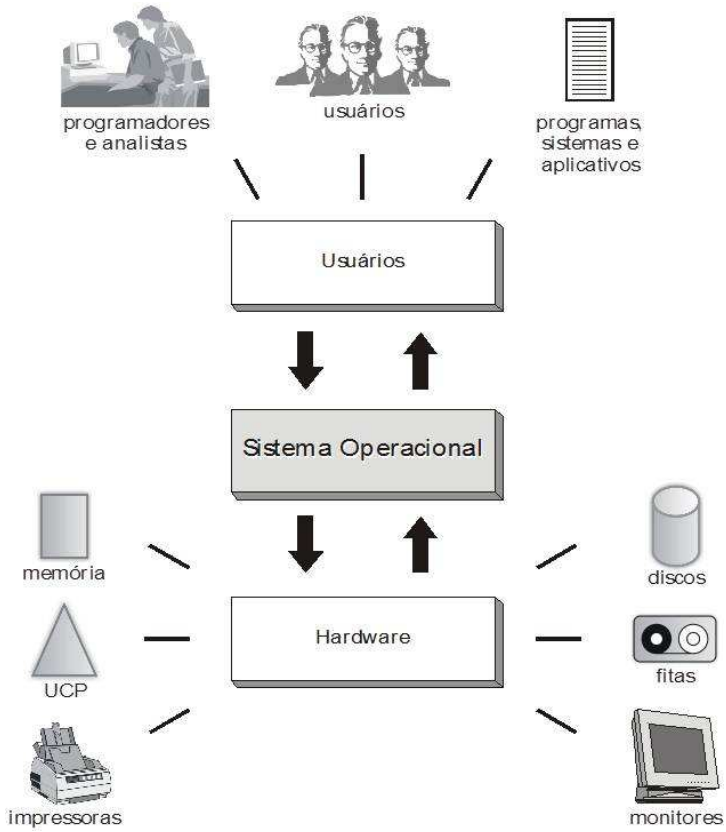
> oferecidos aos usuários, como armazenamento de dados em discos, impressão de relatórios, geração de gráficos, acesso à Internet, entre outras funções.

De nada irá adiantar o *hardware* se não existir um Sistema Operacional, ou seja, somente as peças não irão executar comandos, processamentos, cálculos, ou qualquer que seja a necessidade de quem está operando o computador. As peças fazem o trabalho “braçal”, são elas que vão executar a tarefa, processar os dados e retornar um resultado para o usuário, mas o *hardware* depende de um sistema que informe o que deve ser feito, qual processamento o usuário precisa, qual informação deve ser processada e quais dados devem ser levados em consideração para que o resultado seja obtido. Com essa necessidade surgiu o Sistema Operacional, conforme segue, “Para que o *hardware* tenha utilidade prática, deve existir um conjunto de programas utilizando como interface entre as necessidades do usuário e as capacidades do *hardware*.” (MACHADO, 2007, p.34).

3.2 SISTEMAS OPERACIONAIS

Segundo Machado, “Um *sistema operacional*, por mais complexo que possa parecer, é apenas um conjunto de rotinas executado pelo processador, de forma semelhante aos programas do usuário.” (MACHADO, 2007, p.4). O Sistema Operacional (SO) é uma camada entre as aplicações que o usuário deseja utilizar (como por exemplo: *word, excel* e qualquer ferramenta ou software), e o hardware. Esta arquitetura é demonstrada conforme a Figura 1. Basicamente o SO irá converter comandos que as aplicações executam, pois, o computador só entende linguagem de máquina, ou seja, código binário.

Figura 1 - Visão do Sistema Operacional.



Fonte: MACHADO, 2007, p.4

Outra abordagem sobre sistemas operacionais de acordo com Machado (2007, p.4):

> Sem o sistema operacional, um usuário para interagir com o computador deveria conhecer profundamente diversos detalhes sobre hardware do equipamento, o que tornaria seu trabalho lento e com grandes possibilidades de erros. O Sistema Operacional tem como objetivo funcionar como uma interface entre o usuário e o computador, tornando sua utilização mais simples, rápida e segura.

Por ser um sistema com várias funções, ele é capaz de executar diversos softwares simultaneamente, com arquitetura de execução de processos e rotinas em concorrência. Seu processamento, diferente de uma aplicação comum, não tem começo meio e fim, sendo suas rotinas executadas de maneira linear e concorrentemente de acordo com eventos que podem ocorrer a qualquer momento. Ele controla tanto o uso de *hardware* e recursos do equipamento, quanto as rotinas e eventos a serem executados, previamente solicitados pelas aplicações, de acordo com seu nível de importância. Ainda, Machado (2007, p. 5) alega que:

> Se pensarmos que um computador pessoal nos permite executar diversas tarefas ao mesmo tempo, como imprimir um documento, copiar um arquivo pela Internet ou processar uma planilha, o sistema operacional deve ser capaz de controlar a execução concorrente de todas essas atividades.

Atualmente no mercado existem diversas distribuições de Sistemas Operacionais, tanto sistemas para *desktops*, quanto sistemas *mobile* ou sistemas para *smartphones*. Esses sistemas podem conter uma arquitetura fechada ou uma arquitetura *open source*, ou seja, uma arquitetura livre na qual desenvolvedores podem contribuir com seus conhecimentos e programação de código através de comunidades. Essa distribuição de sistemas operacionais está demonstrada na tabela da Figura 2. Um dos sistemas *open source* mais conhecidos é o Linux ou qualquer sistema baseado em *Unix.* O sistema com arquitetura fechada mais conhecido é o Windows, onde os usuários não conseguem ter acesso ao seu código fonte e não tem o poder de contribuir ou criar novas funcionalidades para a aplicação.

Figura 2 - Tabela de Distribuições de Sistemas Operacionais.

Desktop	
Tipo de Arquitetura	**Distribuições**
Fechada	Windows e MacOs
Open Source	Debian, Ubuntu, Fedora, OpenSuse, RedHat, entre outros sistemas baseados em Unix
Mobile	
Tipo de Arquitetura	**Distribuições**
Fechada	iOS e Windows Phone
Open Source	Android (Sistema baseado em Unix)

Fonte: MACHADO, 2007, p.5

3.3 APLICAÇÕES

Segundo Ferreira, “Um aplicativo (APP) é um software que tem funcionalidades específicas, ou seja, ele é desenvolvido para aperfeiçoar uma função em especial” (FERREIRA, 2013, p.99). O Sistema Operacional, por mais robusto e completo que seja não possui todas as funcionalidades possíveis para gerar um resultado, atingir um objetivo ou até mesmo uma funcionalidade que o usuário desejar. Por exemplo, caso queira escutar música, assistir a um filme, seriado, vídeo, ler uma matéria, livro, revista, criar um documento, entre outras, isso nenhum sistema operacional terá essa função nativamente. Existem programas que o usuário pode instalar que agregam ao computador novas funções, da qual ele não possuía inicialmente. Tendo em vista essa necessidade na qual o SO não possui todas as funções necessárias para cada perfil de usuário, surgiram as aplicações.

Antes de a Internet estar acessível ao público comum, os *softwares* eram sistemas que funcionavam somente em computadores e *off-line*, ou seja, sem qualquer comunicação com outras máquinas remotas. Com isso era muito difícil encaminhar algum documento, relatório, ou qualquer informação compartilhada com outra pessoa. Para isso era necessário exportar esses dados de alguma maneira, seja por impressora, disquete ou qualquer outro tipo de mídia disponível na época.

Como citado por Celso (2011, p.1):

> Desde o início da era computacional, ainda na época dos grandes Mainframes, as ameaças de segurança aos centros de processamento de informações já existiam, mas de modo diferente aos da atualidade, entretanto da mesma forma que os atuais, eram capazes de causar grandes danos aos sistemas. Com o aumento do fluxo de dados e com a necessidade de disponibilizar cada vez mais informações na rede mundial, aumentam assim a exposição de tais dados e também os riscos de segurança relacionados a essa atividade. Um número grande de computadores, armazenando grandes quantidades de dados em uma corporação, por exemplo, traz uma maior área de vulnerabilidade, diferentemente de uma pequena rede doméstica que normalmente não guarda informações valiosas.

Compartilhar informações com outro setor da empresa, uma filial, um cliente, fornecedor, ou qualquer pessoal que desejasse enviar dados, era muito trabalhoso, demandando tempo e recursos da empresa. Graças a essa necessidade e após o surgimento da Internet, com a evolução dos celulares e dispositivos móveis, a tendência foi desenvolver aplicativos que operem nesses dispositivos e principalmente, sistemas que armazenem suas informações na nuvem ou qualquer servidor remoto.

4 VULNERABILIDADES EM APLICAÇÕES WEB

Este capítulo está organizado da seguinte forma: a Seção 4.1 introduz sobre sistemas web e como veem sendo utilizados. A Seção 4.2, descreve sobre as vulnerabilidades de uma aplicação *Web*, e quais são as principais vulnerabilidades utilizadas por atacantes.

4.1 APLICAÇÕES WEB

Segurança em aplicações *Web* de acordo com o Uto (2013, p.1),

> Vulnerabilidades em softwares têm sido amplamente utilizadas por atacantes para roubo de informações confidenciais e invasões de redes corporativas. Prover a segurança de um software, porém, não é um objetivo fácil de ser alcançado, dada a complexidade dos sistemas nos dias de hoje. Facilmente, eles atingem dezenas de milhares de linhas de códigos, que contêm, invariavelmente, quantidade de defeitos significativa. Alguns destes têm impacto direto com segurança, podendo acarretar desde a indisponibilidade do sistema até o controle total do computador por um atacante. Para piorar ainda mais esse cenário, considere que, normalmente, um ciclo de desenvolvimento de software seguro não é adotado, o que resulta, no mínimo, em especificações inseguras e configuração vulnerável das plataformas subjacentes.

Quando as aplicações eram desenvolvidas e disponibilizadas apenas em redes internas corporativas, não existia uma grande preocupação com a segurança dos dados, pois para que o indivíduo pudesse roubar informações, era necessário invadir a segurança física de uma empresa, dado que a esta rede não tinha acesso externo, portanto não era possível realizar uma invasão da rede através do acesso à Internet.

Ainda de acordo com Celso (2011, p1):

> A crescente necessidade de estar acessível a todos, em todos os lugares, através da Internet, implica em garantir proteção contra os ataques virtuais. O grande desafio agora é saber como se proteger e o quanto investir, para evitar invasões e o roubo de informações.

Atualmente é impossível uma organização utilizar alguma aplicação, sem que haja o acesso à rede externa. Com isso, surgiu a preocupação de formas para deixar a rede interna mais segura para que não haja roubo de informações confidenciais da empresa, não somente fisicamente, mas também digitalmente.

Não é fácil desenvolver um sistema de forma segura, dado que atualmente existem sistemas e aplicações que podem chegar a dezenas de milhares de linhas de códigos. Quanto maior a quantidade de códigos escritos, maior é a possibilidade de gerar *bugs* e possíveis vulnerabilidades que o sistema pode apresentar, e isso provavelmente será explorado por um *hacker*.

Para evitar esse tipo de problema, existem formas e ciclos de como desenvolver softwares de maneira segura, principalmente aplicações que são criadas exclusivamente para o usufruto externo. Tal metodologia de desenvolvimento seguro será abordada nos próximos capítulos.

4.2 VULNERABILIDADES

Existem diversas vulnerabilidades em aplicações *Web* bastante conhecidas, e algumas organizações são especializadas em divulgar essas vulnerabilidades, como as que são mais utilizadas por atacantes e quais são as mais perigosas ou que podem oferecer maior risco para o mundo corporativo.

As empresas que tem maior prestigio e são mais conhecidas por alertar quais são as vulnerabilidades exploradas por *hackers*, são: *Common Vulnerabilites and Exposures* e a OWASP.

A *Common Vulnerabilites and Exposures*, é um dicionário público que contém descrições e detalhes de todas as vulnerabilidades já encontradas e relatadas por profissionais da área. Eles relatam também como funcionam os ataques, como foram descobertos e como prevenir-se de cada uma dessas vulnerabilidades.

Complementando dados estatísticos sobre as vulnerabilidades reportadas nesta página pública Uto (2013, p.1) afirma que,

> Para se ter uma ideia mais clara do mundo real, no período de 2001 a 2006 o número de vulnerabilidades em sistemas reportado ao *Common Vulnerabilities and Exposures* simplesmente triplicou. Além disso, houve mudança nos tipos de fraquezas mais comumente encontradas [...]. O extravasamento de buffer, campeão da lista por muitos anos consecutivos, perdeu o lugar a partir de 2005, para vulnerabilidades de injeção de código, como o *Cross-Site Scripting* e Injeção de SQL, por exemplo. Esses tipos de fraquezas afetam, basicamente, sistemas *web* e indicam duas coisas: A recente popularização das interfaces *web* para comércio eletrônico, internet *banking* e configuração de elementos de rede. Os problemas de segurança desse domínio não estão sendo adequadamente considerados durante o processo de desenvolvimento, tanto por ignorância como pela pressão causada por cronogramas de entrega apertados.

Esse dado estatístico vem demonstrando como as aplicações *web* estão mais presentes no mundo corporativo, mas assim como a tecnologia está sempre em constante evolução, os ataques as informações confidenciais de uma empresa também estão evoluindo através da exploração de vulnerabilidades.

O que contribui para maior exposição de vulnerabilidades de um *software* durante o desenvolvimento, além de uma certa ignorância de um desenvolvedor e gerente de projeto de aplicações *web*, são os cronogramas com prazo de entrega muito curto, fazendo com que o programador acabe deixando de lado uma arquitetura mais de segura de um sistema.

Referente à OWASP segundo Uto (2013, p.5):

> O grupo *Open Web Application Security Project* é uma organização mundial, sem fins lucrativos, que visa divulgar aspectos de segurança de aplicações *web*, para que o risco nesses ambientes seja devidamente avaliado por pessoas e empresas. Existem, hoje, 130 capítulos locais, espalhados pelos cinco continentes, todos abertos, gratuitamente, para participação de pessoas interessadas no assunto. A entidade, além de organizar conferências internacionais e encontros sobre o tema, mantém diversos projetos, que variam de guias de implementação segura a ferramentas.

Esse grupo disponibiliza diversas ferramentas, metodologias e arquiteturas seguras para que haja o desenvolvimento com o menor número possível de vulnerabilidades. Dentro do que é disponível na página desse grupo, existem o guia de desenvolvimento escrito por Wiesmann (2005).

O Guia de desenvolvimento tem como objetivo descrever e detalhar quais são as melhores práticas para a construção de um sistema e serviços *web* de forma segura. Esse guia inclui exemplos práticos de implantações e códigos de diversas linguagens como: PHP, JEE e ASP.NET.

Outro guia bastante útil, é o guia de revisão de código, livro escrito por Van der Stock (2008), onde é demonstrado formas de descobrir as vulnerabilidades da aplicação *web* na qual está sendo desenvolvida, através da exploração utilizando inspeção e verificação do código-fonte. No guia contém diversos exemplos práticos para diversas linguagens como: C, C++, ASP e Java.

Para obter uma aplicação *web* mais segura, além de desenvolver através de uma arquitetura mais segura e revisar o código-fonte, é necessário realizar testes de invasão, a fim de encontrar possíveis vulnerabilidades no sistema que foi

desenvolvido. Ademais, para realizar testes de invasões existe o guia de testes escrito por Meucci (2014).

Além desses importantíssimos guias que auxiliam no desenvolvimento de aplicações *web* mais seguras, e indicam como encontrar as vulnerabilidades, existem ferramentas automatizadas de testes de segurança e ferramentas com o intuito de ensinar os conceitos de segurança para uma aplicação.

Uma ferramenta bastante conhecida e utilizada por membros do grupo OWASP e desenvolvedores que se inspiram nesse grupo é o *WebScarab*, que é uma ferramenta escrita utilizando a linguagem Java, que realiza testes automatizados de segurança. Seu principal objetivo é atuar como um *proxy*, no qual captura as requisições feitas pela aplicação *web* até o servidor, permitindo que essa ferramenta altere as respostas. Outra função bastante útil dessa ferramenta é a automatização de testes de injeção de código, permitindo que possa encontrar alguma vulnerabilidade, caso o sistema não esteja protegido contra injeção de códigos feitos pelo atacante.

5 OWASP

Este capítulo está organizado da seguinte forma: a Seção 5.1 aborda sobre o a pesquisa de maiores ataques que acontecem a aplicações web. A Seção 5.2, começa a descrever sobre o ataque de Injeção de SQL. A Seção 5.3 relata sobre o ataque de quebra de autenticação e gerenciamento de sessão. A Seção 5.4 descreve sobre a técnica de *Cross-Site Scripting* (XSS), e como prevenir esse tipo de ataque.

A Seção 5.5 introduz sobre referência insegura e direta a objetos e como proteger a essa vulnerabilidade. Sessão 5.6 aborda sobre as principais consequências da configuração incorreta de segurança, e como pode ser facilmente evitada. Seção 5.7 relata sobre a exposição de dados sensíveis e como proteger essas informações. Seção 5.8 discute sobre a falta de função para controle de nível de acesso e suas principais consequências. A Seção 5.9 descreve sobre o ataque de *Cross-Site Request Forgery* (CSRF) e como prevenir este ataque. A Seção 5.10 informa sobre as principais consequências da utilização de componentes vulneráveis conhecidos. A Seção 5.11 e última, fala sobre os redirecionamentos e encaminhamentos inválidos e como evita-los.

5.1 TOP 10 OWASP

Na comunidade OWASP, os profissionais e colaboradores deste grupo, elegem, de tempos em tempos, uma lista com os 10 maiores tipos de ataques e riscos de segurança mais críticos em aplicações *web.* A última lista atualizada sobre esses ataques foi feita em 2013.

Estas análises que são feitas pela OWASP e seus integrantes, veem demonstrando ao longo do tempo o descaso e despreocupação dos programadores e desenvolvedores de aplicações web, sem levar em consideração a segurança dos *softwares*.

Um comparativo das duas últimas listas (2010 e 2013) realizadas pela comunidade demonstra que ao longo de três anos pouca coisa mudou com relação à segurança dos dados e informações nas aplicações web. Isso foi demonstrado pela OWASP (2013, p5) na Figura 3:

Figura 3 - Tabela comparativa dos ataques mais utilizados nos anos de 2010 e 2013

OWASP Top 10 – 2010 (Anterior)	OWASP Top 10 – 2013 (Novo)
A1 – Injeção de código	A1 – Injeção de código
A3 – Quebra de autenticação e Gerenciamento de Sessão	A2 – Quebra de autenticação e Gerenciamento de Sessão
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Referência Insegura e Direta a Objetos	A4 – Referência Insegura e Direta a Objetos
A6 – Configuração Incorreta de Segurança	A5 – Configuração Incorreta de Segurança
A7 – Armazenamento Criptográfico Inseguro – Agrupado com A9 →	A6 – Exposição de Dados Sensíveis
A8 – Falha na Restrição de Acesso a URL – Ampliado para →	A7 – Falta de Função para Controle do Nível de Acesso
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<Removido do A6: Configuração Incorreta de Segurança>	A9 – Utilização de Componentes Vulneráveis Conhecidos
A10 – Redirecionamentos e Encaminhamentos Inválidos	A10 – Redirecionamentos e Encaminhamentos Inválidos
A9 – Proteção Insuficiente no Nível de Transporte	Agrupado com 2010-A7 criando o 2013-A6

Fonte: OWASP, 2013, p.5

Outras pesquisas enfatizam a falta de interesse dos desenvolvedores de *softwares* com relação a segurança da aplicação. A IBM fez um levantamento e constatou que as vulnerabilidades triplicaram em 2011 (IBM, 2011):

> Outro ponto identificado pelo X-Force foi o crescimento de vulnerabilidades críticas. De janeiro de 2011 até agora, este número triplicou. Existe uma grande variedade de ataques surgindo este ano, como os *whaling*, um tipo de *phishing* dirigido a pessoas do alto nível de uma organização, com acesso a dados críticos.

Se comparado a pesquisa OWASP e IBM, fica comprovado que além de aumentar o número de ataques, os ataques continuam os mesmos. Isso poderia facilmente ser evitado se os programadores de aplicações, se preocupassem com a segurança da ferramenta.

5.2 INJEÇÃO DE SQL

O primeiro item da lista Top 10 segundo a comunidade OWASP é a injeção ou Injeção de SQL (OWASP, 2013, p.7):

> Injeção: As falhas de Injeção, tais como Injeção de SQL, de SO (Sistema Operacional) e de LDAP, ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados

> manipulados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.

Este tipo de ataque é o mais comum de ocorrer e vem se mantendo no top 10 da lista OWASP desde a sua primeira versão do ranking em 2004, no qual ele era o número seis da lista. A partir de 2007 ele foi se tornando mais popular e ficou entre os primeiros lugares da lista, ocupando o segundo lugar. De lá para cá pouco mudou com relação a esse tipo de ataque, e ocupou o primeiro lugar nas duas últimas listas (2010 e 2013). Isto demonstra a falta de preocupação dos desenvolvedores de aplicações *Web,* mesmo em relação aos ataques que ficaram conhecidos ao longo do tempo.

Na Injeção de SQL, o atacante explora a vulnerabilidade do sistema no qual o desenvolvedor não a protegeu contra fontes de dados não confiáveis, ou seja, fontes de dados externas, que não teve origem dentro da própria aplicação. Desta forma o atacante envia uma instrução SQL ou um comando via texto, no qual o sistema irá interpretar esta sintaxe e executar o que o atacante deseja, roubando informações ou até mesmo alterando informações que estão gravadas no banco de dados do software.

Primeiramente é necessário verificar se aplicação está vulnerável. De acordo com a comunidade (OWASP, 2013, p.8):

> A melhor forma para descobrir se uma aplicação está vulnerável à injeção é verificar se todos os usos dos interpretadores separam claramente os dados não-confiáveis do comando ou consulta. Para chamadas SQL, isso significa utilizar variáveis de ligação em todas as instruções preparadas e procedimentos armazenados, e evitar consultas dinâmicas.

Quando o desenvolvedor checa todas as entradas de dados garantindo que essas informações são de fonte confiável e principalmente que partiram da própria aplicação ele garante que os *inputs* são verdadeiros e que não partiram de instruções via comando ou consulta de informações realizados por fontes externas à aplicação.

5.3 QUEBRA DE AUTENTICAÇÃO E GERENCIAMENTO DE SESSÃO

O segundo item da lista segundo a comunidade OWASP é a Quebra de Autenticação e Gerenciamento de Sessão, de acordo com o Top 10 (OWASP, 2013, p.8):

> As funções da aplicação relacionadas com autenticação e gerenciamento de sessão geralmente são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e tokens de sessão ou, ainda, explorem outra falha da implementação para assumir a identidade de outros usuários.

Este tipo de ataque explora a vulnerabilidade do sistema que diz respeito à parte de autenticidade, ou seja, vislumbra usufruir de um acesso privilegiado ao sistema passando por um usuário existente na base dados da aplicação através da clonagem da sessão. Sessão é o período no qual a aplicação considera que o *login* para acesso a aplicação é valido, sendo que, ao final deste período é solicitado ao usuário que digite novamente o usuário e senha.

Ademais, outra vulnerabilidade são as senhas que não estão criptografadas, deixando as mesmas expostas para o atacante interceptá-las, realizando a clonagem de chaves ou *tokens* de acesso a aplicação.

Para evitar ou melhorar a proteção contra esse tipo de vulnerabilidade é necessário de acordo com os membros da OWASP (2013, p.9) verificar se:

> 1. As credenciais de autenticação de usuário não estão protegidas utilizando *hash* ou criptografia, quando armazenadas. 2. As credenciais podem ser descobertas através de fracas funções de gerenciamento de contas (por exemplo, criação de conta, alteração de senha, recuperação de senha, *IDs* de sessão fracos). 3. *IDs* de sessão são expostos na URL (por exemplo, reescrita de URL). 4. *IDs* de sessão são vulneráveis a ataques de fixação de sessão. 5. *IDs* de sessão não expiram, ou sessões de usuário ou *tokens* de autenticação, particularmente aqueles baseados em single *sign-on* (SSO), e não são devidamente invalidados durante o *logout*. 6. *IDs* de sessão não são *rotacionados* após o *login* bem-sucedido. 7. Senhas, *IDs* de sessão, e outras credenciais são enviadas através de conexões não criptografadas.

É certo que, para se obter uma maior proteção ou segurança em senhas e sessões de usuário, é necessário utilizar *hash* (isso garante que não foi alterado a sessão do usuário) e criptografia (dificultando a interceptação das senhas dos usuários). Além de proteger a senha do usuário e a sessão com *hash* e criptografia, o desenvolvedor deve rever toda a sua rotina de cadastro de usuário no sistema, alteração ou recuperação de senhas, pois esses serviços podem estar com a segurança prejudicada.

5.4 CROSS-SITE SCRIPTING (XSS)

O próximo item da lista Top 10 segundo a comunidade OWASP é o *Cross-Site Scripting* (XSS) (OWASP, 2013, p.8)*,*

> Falhas XSS ocorrem sempre que uma aplicação recebe dados não confiáveis e os envia ao navegador sem validação ou filtro adequados. XSS permite aos atacantes executarem scripts no navegador da vítima que podem "sequestrar" sessões do usuário, desfigurar sites, ou redirecionar o usuário para sites maliciosos.

Este tipo de ataque normalmente é utilizado para alterar sites, trocando textos, figuras ou até mesmo *links* de acesso. Com isso o *hacker* consegue direcionar o usuário para uma página falsa ou sites maliciosos. Muitos usuários já sofreram esse tipo de ataque ao acessar um *e-commerce* (Loja virtual), clicando em um produto que deseja comprar, e ao clicar sua *url* (endereço da página), foi direcionada para um site fraudulento.

Além desse tipo de ataque, também é possível interceptar a sessão de um *login* (usuário), passando a ter um acesso privilegiado e até mesmo se passar por outra pessoa.

Para prevenir essa vulnerabilidade, de acordo com a recomendação da OWASP (2013, p.10):

> Você está vulnerável se não garantir que todas as entradas fornecidas pelos usuários sejam apropriadamente filtradas, ou você não verifica que elas sejam seguras via validação de entrada, antes de incluir essa entrada na página de saída. Sem o adequado filtro ou validação da saída, tal entrada será tratada como conteúdo ativo no navegador.

A segurança em aplicações *web* não se baseia apenas em proteger as entradas dos dados, mas também a saída dos dados antes que a pagina seja exibida ao usuário, pois este tipo de ataque utilizando o XSS, visa alterar a saída dos dados no momento em que a página ou site está sendo carregado no navegador do usuário.

5.5 REFERÊNCIA INSEGURA E DIRETA A OBJETOS

O quarto item da lista segundo a comunidade OWASP é o Referência Insegura e Direta a Objetos, de acordo com o Top 10 (OWASP, 2013, p.8):

> Uma referência insegura e direta a um objeto ocorre quando um programador expõe uma referência à implementação interna de um objeto, como um

> arquivo, diretório, ou registro da base de dados. Sem a verificação do controle de acesso ou outra proteção, os atacantes podem manipular estas referências para acessar dados não-autorizados.

É muito comum desenvolvedores deixarem comentários em um código fonte que está exposto no *front-end* (aplicação que é exibida ao usuário final), isso facilita a documentação de trechos de códigos para que outro desenvolvedor possa entender aquele trecho de código. Ocorre que, em relação à segurança, isso é muito ruim, pois atacantes podem utilizar dessas informações para entender como está estruturado o *software* alvo.

Outro ponto de extrema importância é verificar os acessos privilegiados de cada arquivo, diretório ou registro do bando de dados. Todo e qualquer acesso mínimo que seja feito, deve ser verificado qual é o grau de acesso que o usuário pode ter, deixando apenas exposto o que ele realmente tem direito a acessar, evitando expor informações confidencias.

Uma das dicas apresentadas pelo OWASP (2013, p.11) é:

> Uso de referência indiretas a objetos por usuário ou sessão. Isso impede que o atacante atinja diretamente os recursos não autorizados. Por exemplo, em vez de utilizar a chave de banco de dados do recurso, uma lista de seis recursos autorizados para o usuário atual poderia utilizar os números de 1 a 6 para indicar qual valor o usuário selecionou. A aplicação tem que mapear as referências indiretas por usuário de volta para a chave do banco de dados real no servidor. Verificar o acesso. Cada utilização de uma referência direta a objeto de uma origem não confiável deve incluir uma verificação de controle de acesso para garantir que o usuário está autorizado para o objeto requisitado.

Os serviços que expõem os elementos da tela, não devem publicar os *ids* do registro que está gravado no banco de dados. O correto seria fazer uma tabela de identificação, “de para”, dos *ids* indicando que o item “x” é igual ao item “y”. Assim como no item 3 da lista Top 10 da OWASP (2013), o desenvolvedor deve sempre verificar o controle de acesso do usuário, e se a sua origem é de fonte confiável.

5.6 CONFIGURAÇÃO INCORRETA DE SEGURANÇA

O quinto item da lista segundo a comunidade OWASP é a Configuração Incorreta de Segurança, de acordo com o Top 10 (OWASP, 2013, p.8):

> Uma boa segurança exige a definição de uma configuração segura e implementada na aplicação, frameworks, servidor de aplicação, servidor web,

> banco de dados e plataforma. Todas essas configurações devem ser definidas, implementadas e mantidas, já que geralmente a configuração padrão é insegura. Adicionalmente, o software deve ser mantido atualizado.

Outro erro comum dos desenvolvedores é a configuração incorreta de aplicações, *frameworks*, banco de dados ou qualquer tipo de recurso que irá ser utilizado pela aplicação desenvolvida. Manter senha padrão de um recurso, configurações *default*, ou até mesmo porta padrão utilizada pelo recurso, tudo isso facilita ao *hacker* que deseja realizar um ataque, pois o mesmo tem o conhecimento dessas configurações que vêm de fábrica.

Além desse erro comum, muitos programadores acabam adotando a mesma senha para o controle de acesso em diversas camadas da aplicação. Com isso se o atacante descobrir qual é a senha utilizada para acessar o servidor de aplicação, consequentemente o *hacker* conseguirá ter acesso a qualquer recurso que desejar acessar.

Segundo a OWASP (2013, p.12), para evitar este tipo de ataque é necessário:

> Um processo de *hardening* recorrente que torne fácil e rápido de implantar outro ambiente que está devidamente blindado. Ambientes de desenvolvimento, controle de qualidade e produção devem ser todos configurados de forma idêntica (com senhas diferentes usadas em cada ambiente). Este processo deve ser automatizado para minimizar o esforço necessário para configurar um novo ambiente seguro. Um processo para se manter a par e implantar todas as novas atualizações e correções de software em tempo hábil e em para cada ambiente. Uma arquitetura de aplicação forte que forneça a separação segura e eficaz entre os componentes. Considere executar varreduras e fazer auditorias periodicamente para ajudar a detectar erros futuros de configuração ou correções ausentes.

Portanto, neste caso, o principal erro cometido é deixar configurações de fábrica, de *frameworks*, aplicações ou qualquer recurso utilizado no servidor da aplicação. Isso incluí: senhas, configurações e porta utilizada pela aplicação. Deve-se manter sempre atualizado os recursos, pois nessas atualizações são feitas as correções de vulnerabilidades para manter uma arquitetura segura e eficaz entre as camadas da aplicação.

Também é muito recomendado a execução de varreduras, em busca de vulnerabilidades, atualizações e até mesmo detectar erros de configurações que estão implementadas no servidor.

5.7 EXPOSIÇÃO DE DADOS SENSÍVEIS

O sexto item da lista segundo a comunidade OWASP é a Exposição de Dados Sensíveis, de acordo com o Top 10 (OWASP, 2013, p.8):

> Muitas aplicações web não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação. Os atacantes podem roubar ou modificar esses dados desprotegidos com o propósito de realizar fraudes de cartões de crédito, roubo de identidade, ou outros crimes. Os dados sensíveis merecem proteção extra como criptografia no armazenamento ou em trânsito, bem como precauções especiais quando trafegadas pelo navegador.

Os dados que são considerados sensíveis devem sempre ter uma proteção ou uma cautela extra. Informações que são consideradas sensíveis são informações que, caso roubadas ou violadas, podem causar um grande estrago ou prejuízo a organização. Tais quais dados de cartão de crédito, documentos, credenciais de acesso, entre outros. Essas informações normalmente são as mais visadas pelos atacantes, pois podem lhe causar um certo lucro com essas informações, como por exemplo: uso para fraudes, roubo de identidade, acesso privilegiado com as credenciais, ou outros tipos de crimes que podem ser praticados.

Para evitar este tipo de ataque, é necessário de acordo com a OWASP (2013, p13):

> Considerando que você pretende proteger os dados de ameaças (como por exemplo, ataque interno ou de usuário externo), tenha a certeza de criptografar todos os dados sensíveis em repouso e em trânsito de uma forma que iniba estas ameaças. Não armazene dados sensíveis desnecessariamente. Descarte-os o mais rápido possível. Dados que você não tem não podem ser roubados. Certifique-se que o nível utilizado nos algoritmos e chaves são fortes, e que o gerenciamento de chaves está aplicado adequadamente. Considere utilizar os módulos criptográficos validados do FIPS-140. Certifique-se que as senhas são armazenadas com um algoritmo projetado especialmente para a proteção de senhas, como o bcrypt, PBKDF2 ou scrypt. Desabilite o auto completar em formulários de coleta de dados sensíveis e desabilite o cache em páginas que contenham dados sensíveis.

A criptografia de dados considerados sensíveis é algo indispensável para a segurança e o bom funcionamento da aplicação *web*. Deve-se sempre criptografar essas informações quando estiverem armazenadas ou em trânsito. Na maioria das vezes os atacantes irão interceptar o trafego de dados em busca desse tipo de informação.

Sempre que for armazenar informações sensíveis e confidenciais certifique-se se realmente é necessário o armazenamento dessas informações, pois, guardar informações desnecessariamente só irá aumentar o grau de segurança que se deve ter com essas informações, sem que haja realmente a necessidade de guardá-las.

É necessário verificar se a criptografia utilizada para trafegar ou salvar essas informações estão atualizadas ou se não são arquitetura de criptografia defasada, na qual não recebem melhorias ou atualizações, e, sempre utilizar as melhores criptografias disponíveis no mercado, elevando assim, o grau de segurança da aplicação.

Além de verificar se realmente é necessário guardar a informação, sempre desabilite o auto completar dos formulários utilizados na aplicação. Se realmente o usuário deve preencher essas informações, garanta que ele os digite novamente, e não armazene em *cache* ou arquivos temporários. Armazenar informações nesse recurso do navegador, foge da proteção utilizada nos servidores, portanto ficará mais vulnerável a extração dessas informações pelo atacante.

5.8 FALTA DE FUNÇÃO PARA CONTROLE DO NÍVEL DE ACESSO

O sétimo item da lista segundo a comunidade OWASP é a Falta de Função para Controle do Nível de Acesso, de acordo com o Top 10 (OWASP, 2013, p.8):

> A maioria das aplicações web verificam os direitos de acesso em nível de função antes de tornar-se essa funcionalidade visível na interface do usuário. No entanto, as aplicações precisam executar as mesmas verificações de controle de acesso no servidor quando cada função é invocada. Se estas requisições não forem verificadas, os atacantes serão capazes de forjar as requisições, com o propósito de acessar a funcionalidade sem autorização adequada.

É comum aplicações *web* verificar se o usuário tem acesso a determinada funcionalidade do sistema, garantindo desta forma que o funcionário não tenha acesso a folha de pagamento se o mesmo não tiver está permissão ativa. Porém está verificação normalmente é feita apenas no formulário ou tela, e não é feita no servidor. Isso protege apenas que usuários não tenham acesso a esse tipo de informação, e não protege contra os *hackers.*

De acordo com a OWASP (2013, p.14):

> Pense sobre o processo para gerenciar os direitos e garantir que você possa atualizar e auditar facilmente. Não codifique diretamente. A execução de mecanismos deve negar todo o acesso por padrão, exigindo direitos explícitos para papéis específicos no acesso a todas as funções. Se a função está envolvida em um fluxo de trabalho, verifique, para ter certeza, se as condições estão em estado adequado para permitir acesso.

Para garantir o direito de acesso é necessário não garantir apenas na tela ou na apresentação do sistema, mas sim garantir o grau de acesso no servidor e nos seus controladores, pois, são eles que realmente devem garantir o acesso que deve ser permitido a cada usuário. Com isso, o programador não deve deixar fixo o acesso de cada usuário, mas sim deixar de forma dinâmica, na qual fique fácil a manutenção de acessos, atualizações e até mesmo a auditória.

Visando uma maior segurança de acesso, por padrão, qualquer tipo de acesso deve ser negado, permitindo o acesso apenas quando realmente o usuário tenha a permissão concedida.

5.9 *CROSS-SITE REQUEST FORGERY* (CSRF)

O oitavo item da lista segundo a comunidade OWASP é o *Cross-Site Request Forgery* (CSRF), de acordo com o Top 10 (OWASP, 2013, p.8):

> Um ataque CSRF força a vítima que possui uma sessão ativa em um navegador a enviar uma requisição HTTP forjada, incluindo o cookie da sessão da vítima e qualquer outra informação de autenticação incluída na sessão, a uma aplicação web vulnerável. Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas realizadas pela vítima.

Mesmo após fazer todas as verificações anteriores, ainda assim, é possível invadir uma aplicação *web*. O *hacker* explora a vulnerabilidade na aplicação que não verifica se a requisição partiu de dentro da própria aplicação ou de uma fonte confiável. Com isso, o atacante acaba enviando uma requisição ao servidor da aplicação *web*, utilizando o *cookie* da sessão forjando uma requisição para a aplicação.

Para proteger a aplicação contra esse tipo de ataque, de acordo com a OWASP (2013, p.15) deve-se:

> A prevenção de um CSRF geralmente requer a inclusão de um *token* imprevisível em cada requisição HTTP. Tais *tokens* devem, no mínimo, ser únicos por sessão de usuário. A opção preferida consiste em incluir um *token* único em um campo oculto. Isso faz com que o valor seja enviado no corpo

> da requisição HTTP, evitando-se a sua inserção na URL, que é mais propensa a exposição. O *token* único pode ser incluído na própria URL, ou em parâmetros da URL. Contudo, tal posicionamento corre um risco maior já que a URL será exposta ao atacante, comprometendo assim o *token* secreto.

O *token* (chave de segurança para acesso a aplicação, que garante a sessão do usuário), deve ser sempre bem protegido com criptografia e de preferência fazer uma verificação com *hash* (algoritmo que garante a integridade do conteúdo, caso o conteúdo seja alterado o código de verificação mudará). Desta maneira é garantido uma segurança maior.

Um erro muito comum cometido por programadores para tentar proteger esse *token* é ocultar o seu conteúdo em um campo oculto, desta forma a chave de acesso está totalmente vulnerável, pois um usuário comum não saberá encontrar esse conteúdo, mas com certeza um atacante terá acesso a essa chave de segurança.

Outra técnica bastante praticada por desenvolvedores de *software* é o envio dessa chave de acesso na própria *URL* da aplicação web. Esse método é o mais vulnerável, pois qualquer indivíduo mal-intencionado, poderá usufruir de maneira ilícita do acesso a aplicação.

5.10 UTILIZAÇÃO DE COMPONENTES VULNERÁVEIS CONHECIDOS

O penúltimo item da lista segundo a comunidade OWASP é a Utilização de Componentes Vulneráveis Conhecidos, de acordo com o Top 10 (OWASP, 2013, p.8):

> Componentes, tais como bibliotecas, *frameworks*, e outros módulos de software quase sempre são executados com privilégios elevados. Se um componente vulnerável é explorado, um ataque pode causar sérias perdas de dados ou o comprometimento do servidor. As aplicações que utilizam componentes com vulnerabilidades conhecidas podem minar as suas defesas e permitir uma gama de possíveis ataques e impactos.

Este tipo de ataque explora vulnerabilidades de aplicações web que estão com seus componentes, bibliotecas, *frameworks* (componentes que têm a função de realizar consultas no banco de dados, ou até mesmo criar relatórios), ou qualquer tipo de módulo de uma aplicação.

Estas vulnerabilidades podem ser privilégio elevado que um componente tem sem necessidade ou sem proteção alguma, como a utilização de um *framework* desatualizado, principalmente no quesito de atualizações de segurança. Também

pode ocorrer que uma determinada biblioteca utilizada na aplicação foi descontinuada por seus criadores.

Para evitar este tipo de ataque, deve ser feito os seguintes ajustes, de acordo com a OWASP (2013, p16):

> Identificar todos os componentes e as versões que você está utilizando, incluindo todas as dependências. (Ex., versões dos *plugins*). Monitorar a segurança desses componentes em banco de dados públicos, listas de e-mail de projetos e segurança, e mantê-los atualizados. Estabelecer políticas de segurança que definam o uso do componente, assim como exigir certas práticas de desenvolvimento de software, passando em testes de segurança, e licenças aceitáveis. Quando apropriado, considere a adição de invólucros de segurança em torno dos componentes para desabilitar funcionalidades não utilizadas e/ou proteger falhas ou aspectos vulneráveis do componente.

O primeiro passo é a identificação de todos os componentes e as respectivas versões que estão sendo utilizadas no desenvolvimento do *software*. Após ter sido realizado este levantamento, deve-se monitorar esses *frameworks* no quesito de segurança, verificando qual é o comportamento dessas bibliotecas com relação a banco de dados público e sempre manter os mesmos atualizados. Caso um *plugin* foi descontinuado, encontrar o mais rápido possível um substituto.

Deve ser definido um procedimento de segurança no qual se estabelece recomendações e práticas de segurança na utilização de componentes de terceiros. Essas definições devem ser utilizadas pelos programadores no momento do desenvolvimento de uma aplicação web. Após ter sido desenvolvido é recomendado que sejam realizados testes de segurança no aplicativo.

Também é de extrema importância sempre deixar desabilitado as funcionalidades que não são utilizadas pela aplicação, evitando que possíveis falhas não identificadas possam ser utilizadas por um atacante.

5.11 REDIRECIONAMENTOS E ENCAMINHAMENTOS INVÁLIDOS

O último item da lista segundo a comunidade OWASP é o Redirecionamentos e Encaminhamentos Inválidos, de acordo com o Top 10 (OWASP, 2013, p.8):

> Aplicações web frequentemente redirecionam e encaminham usuários para outras páginas e sites, e usam dados não confiáveis para determinar as páginas de destino. Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sites de phishing ou malware, ou usar encaminhamentos para acessar páginas não autorizadas.

Uma prática bastante utilizada recentemente é o *phishing*. Esta prática consiste em tentar extrair algum tipo de informação do usuário. Sua maior característica é a criação de páginas *web* falsas, na qual o usuário é atraído por algum tipo de anúncio, e-mail ou até mesmo a exploração de uma vulnerabilidade de um determinado site. O *hacker* usa desta página falsa para que o usuário acredite que está realizando alguma compra, acessando um sistema financeiro ou bancário, assim, o usuário irá fornecer qualquer tipo de informação que o atacante solicitar.

Esta prática irá explorar a vulnerabilidade do sistema web que realiza grandes redirecionamentos à sites externos ou de terceiros. Sempre que realizar este tipo de prática os programadores devem verificar a origem da solicitação e dos dados, e, se são confiáveis para realizar o encaminhamento do usuário para outra página web.

A recomendação da OWASP (2013, p17) com relação desse tipo de ataque é:

> Uso seguro de redirecionamentos e encaminhamentos pode ser feito de várias formas: simplesmente evitar usá-los. Se forem usados, não envolva parâmetros do usuário no cálculo do destino. Normalmente, isto pode ser feito. Se os parâmetros de destino não podem ser evitados, tenha certeza que o valor fornecido é válido, e autorizado para o usuário. Recomenda-se que qualquer parâmetro de destino seja um valor mapeado, e não a URL real ou parte dela, e que o código do lado do servidor traduza este mapeamento para a URL de destino.

O redirecionamento e encaminhamento de página é um método muito inseguro e requer várias verificações para aumentar a segurança. Ainda que ocorram diversas verificações, é uma prática bastante insegura, por tanto deve ser evitada ao máximo. Caso não seja possível evitar este tipo de técnica, não utilizar parâmetros enviados pelo usuário para calcular a rota de destino ou qual é a página que será redirecionada.

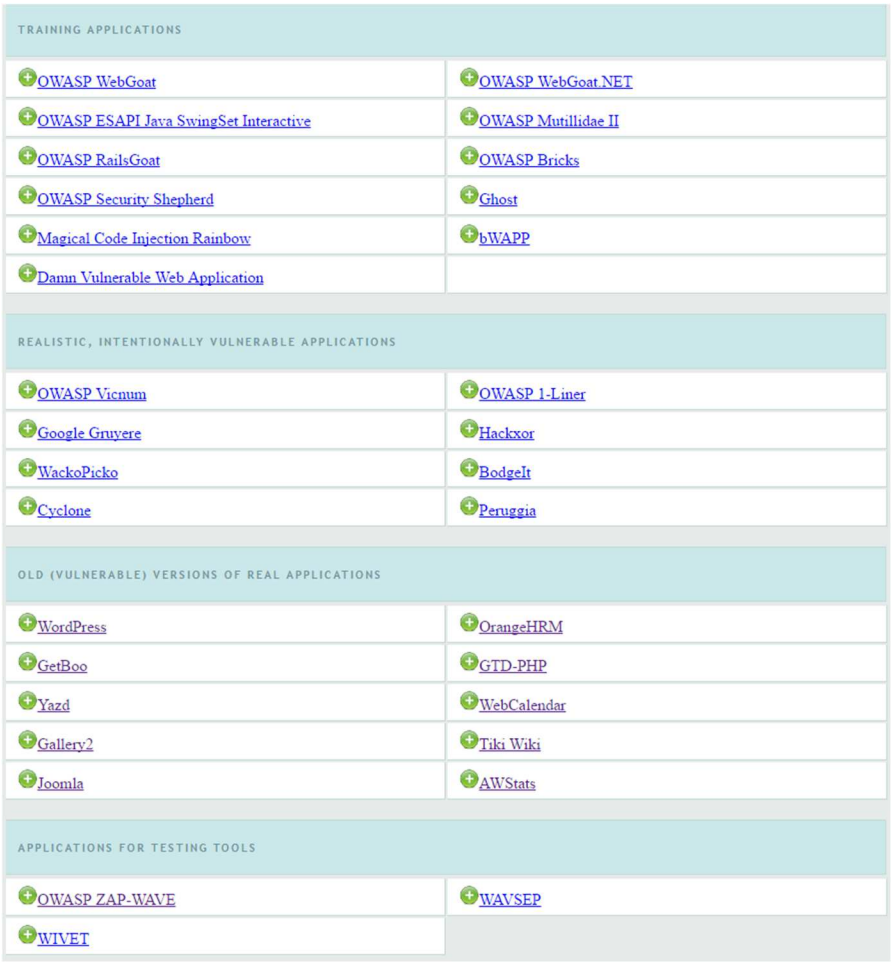
Se realmente for necessário a utilização de parâmetros deve ser certificado que a informação enviada é realmente de fonte confiável e se a mesma é válida. Também deve se certificar de que o acesso é realmente autorizado para o usuário pertencente a sessão. Nesta prática de redirecionamento é recomendado que seja feita através da utilização de valores mapeados, e não utilizar uma *URL* válida e real. Esta tradução deve ser feita no servidor evitando qualquer tipo de interação do atacante.

6 ESTUDO DE CASO

Para a realização do estudo de caso foi utilizado uma VM (*Virtual Machine*, ou seja, máquina virtual, na qual é criado um ambiente controlado para a realização dos testes de invasão) e a implementação de um dos ataques mais usados por *hackers* de acordo com a lista “Top 10 OWASP”, abordado no capítulo anterior, o *SQL Injection*.

A OWASP fornece uma VM (OWASP, 2013), com várias ferramentas de escaneamento de vulnerabilidades de segurança, aplicações para treinamento, versões de aplicações vulneráveis e ferramentas para testes de segurança. Esta máquina virtual tem um servidor apache, com isso é possível ter acesso às ferramentas e aplicações através de uma interface *web*, conforme demonstrado na Figura 4.

Figura 4 – Página inicial OWASP: *Broken Web Applications Project*



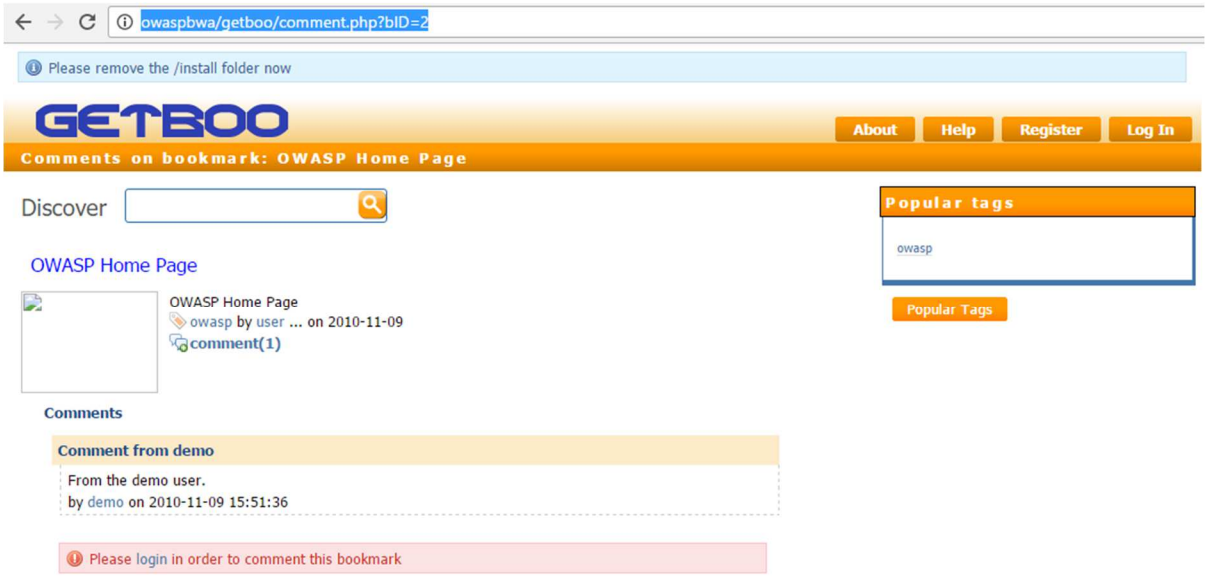
Fonte: *print screen* da aplicação Máquina Virtual OWASP

Foi utilizada a aplicação *GetBoo* como alvo para a realização do ataque utilizando o método *SQL Injection*. A falha para realização de um *SQL Injection* caracteriza-se principalmente em ter uma página que utilize parâmetros na URL. É possível utilizar parâmetros de maneira segura, mas, nesta aplicação não estava sendo utilizada segurança nenhuma contra esse tipo de ataque.

Para realizar este tipo de ataque foi utilizada uma ferramenta, chamada *Havij*, que realiza diversas requisições em busca da vulnerabilidade. Assim, foi possível extrair qualquer tipo de informação armazenada no banco de dados da aplicação alvo.

Identificar se a página alvo tem proteção contra este tipo de vulnerabilidade é simples, primeiro foi localizada a página que utiliza parâmetros na URL, conforme demonstrado na Figura 5.

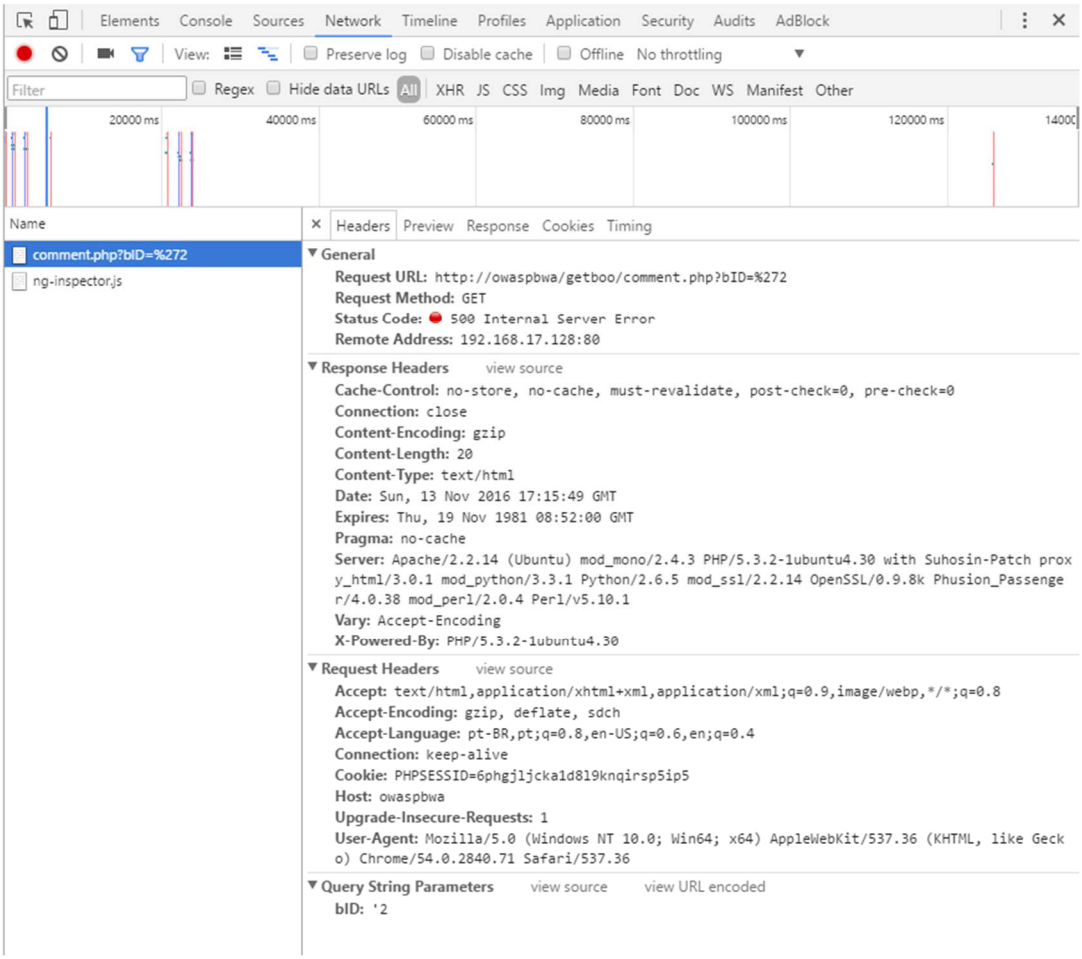
Figura 5 - Página que utiliza parâmetro na URL



Fonte: *print screen* da aplicação Máquina Virtual OWASP

Foi identificado que a página contém a falha de segurança contra este tipo de ataque. Para realizar esta identificação, basta colocar aspas simples antes do sinal de igual na URL alvo. Ao inspecionar a página web, utilizando o *browser* para teste Google Chrome, foi possível identificar que a requisição *HTTP* do tipo *GET* apresentou um erro interno no servidor, conforme demonstrado na Figura 6.

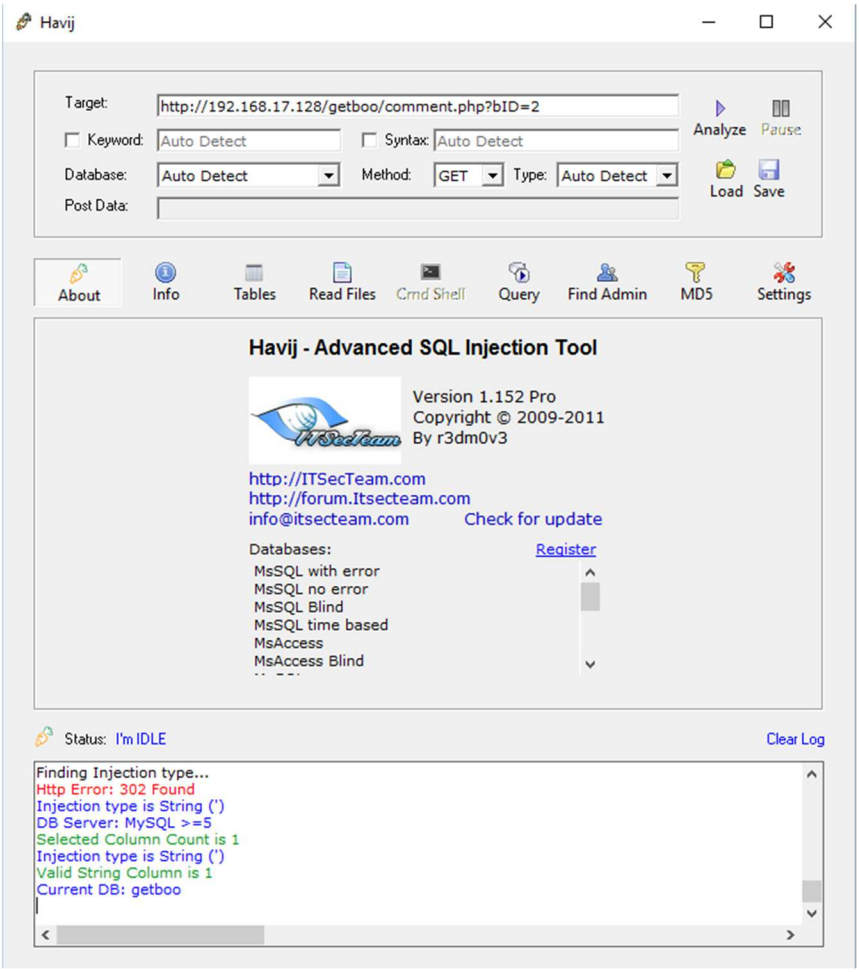
Figura 6 - Erro da requisição HTTP para o servidor



Fonte: *print screen* da aplicação Máquina Virtual OWASP

Após a identificação da página alvo ser uma aplicação vulnerável contra este tipo de ataque, foi feito uma análise da URL, através da ferramenta *Havij*. Ao clicar no botão “*Analyze”* conforme demonstrado na Figura 7, a ferramenta realiza várias requisições, até encontrar o banco de dados. Desta forma é possível extrair qualquer tipo de informação do banco de dados. Nesta figura é possível identificar que depois de realizar toda a varredura, foi identificado um banco de dados com nome “getboo”.

Figura 7 - Análise da URL através da ferramenta *Havij*



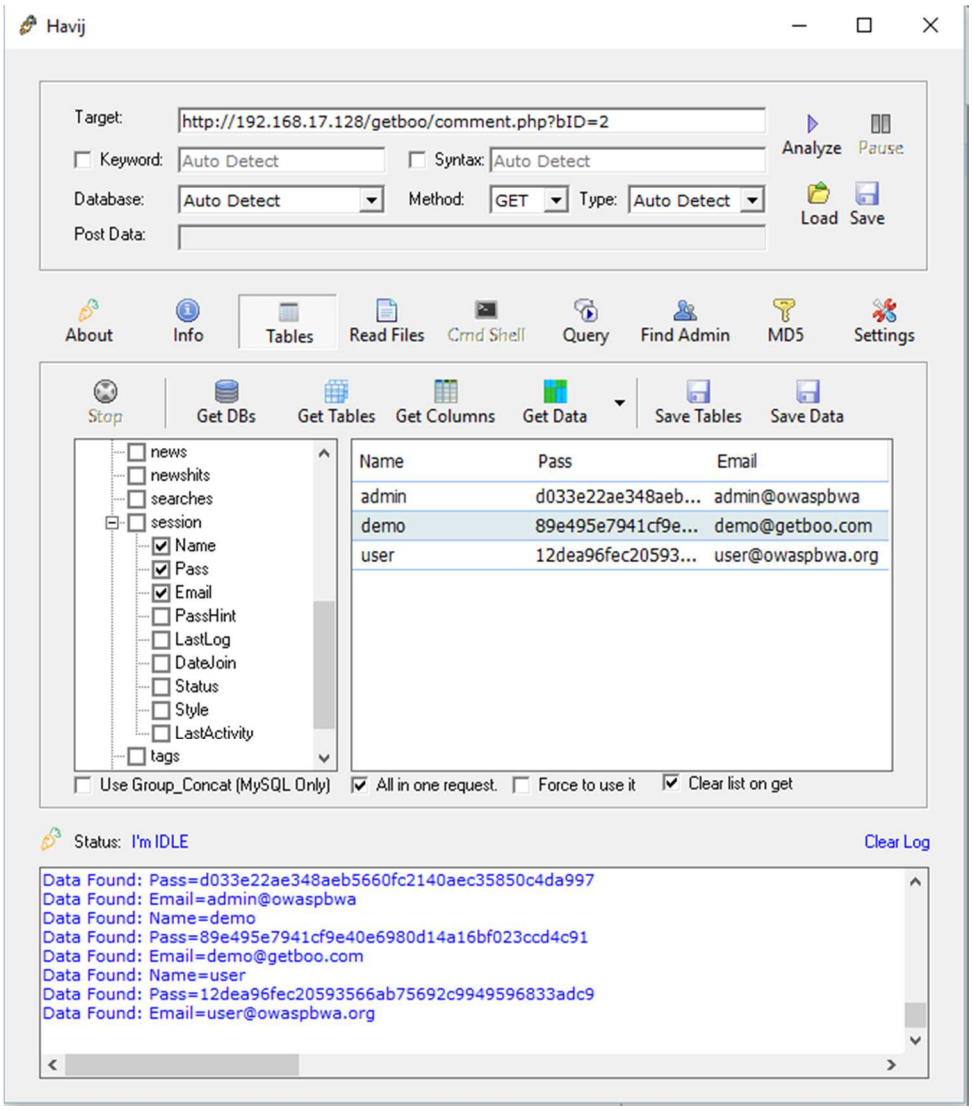
Fonte: *print screen* da aplicação Ferramenta *Havij*

Quando é encontrado o banco de dados a ferramenta disponibiliza todas as informações do sistema, através da opção "*Tables"* conforme demonstrado na Figura 8. Nesta aba da ferramenta é disponibilizado toda a estrutura do banco de dados. Primeiro é necessário localizar o banco de dados, isto é possível através da opção "*GetDBs*". Desta forma é listado todos os bancos utilizados pela aplicação, neste caso é o "*getboo*".

Após listar o banco de dados selecionar o mesmo e clicar no botão "*GetTables*". Com isso será listado todas as tabelas disponíveis no banco de dados. Foi escolhido a tabela "*session",* no qual está salvo todos os usuários, com seus e-mails e senhas para acesso da aplicação. Ao clicar na opção "*GetColumns",* é listado todas as colunas que existe nesta tabela.

Para realizar o ataque foi escolhido as principais informações para acesso da aplicação: "*name", "pass", "email". Name ou email,* que normalmente são utilizados como o acesso da aplicação, "*pass"* neste caso é o *password* ou senha.

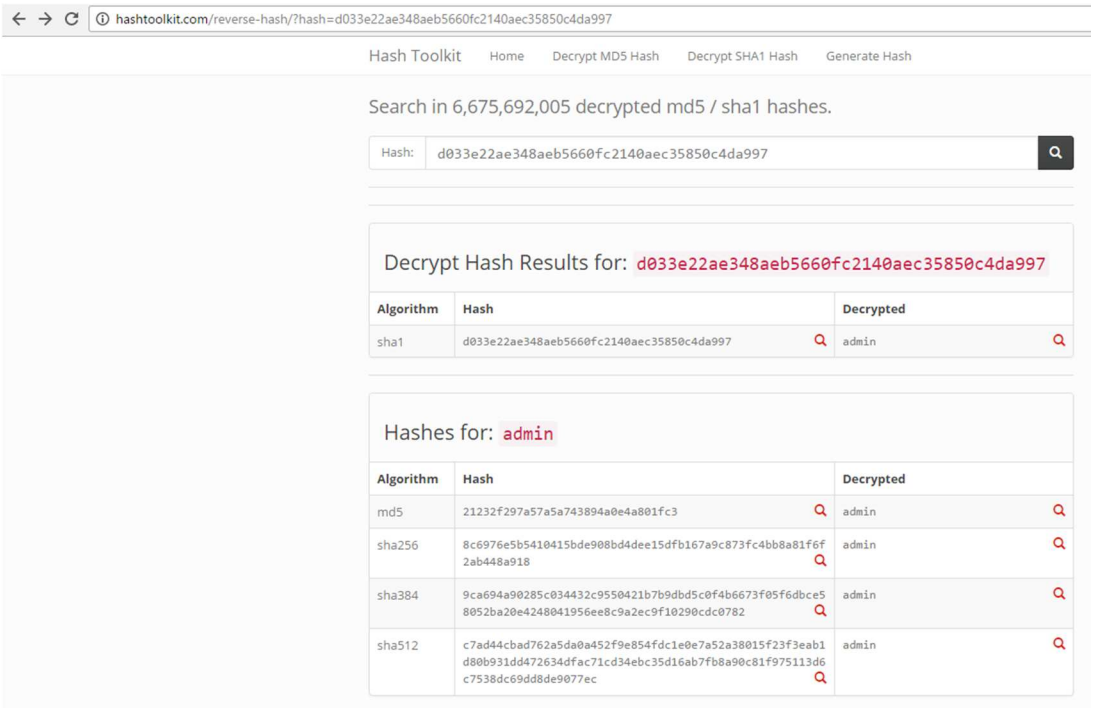
Figura 8 - Extraindo informações do sistema



Fonte: *print screen* da aplicação Ferramenta *Havij*

Para criptografar a senha foi utilizado o algoritmo SHA1. Caso fosse utilizado o algoritmo MD5, a própria ferramenta *Havij* teria apresentado a senha de forma legível, através do botão “MD5”. Para desvendar a senha utilizada no cadastro deste usuário, foi utilizado uma ferramenta *online* chamada *HashToolKit*. Para quebrar este *hash* basta colocar o mesmo na caixa de pesquisa, e a ferramenta irá exibir a senha conforme demostrado na Figura 9.

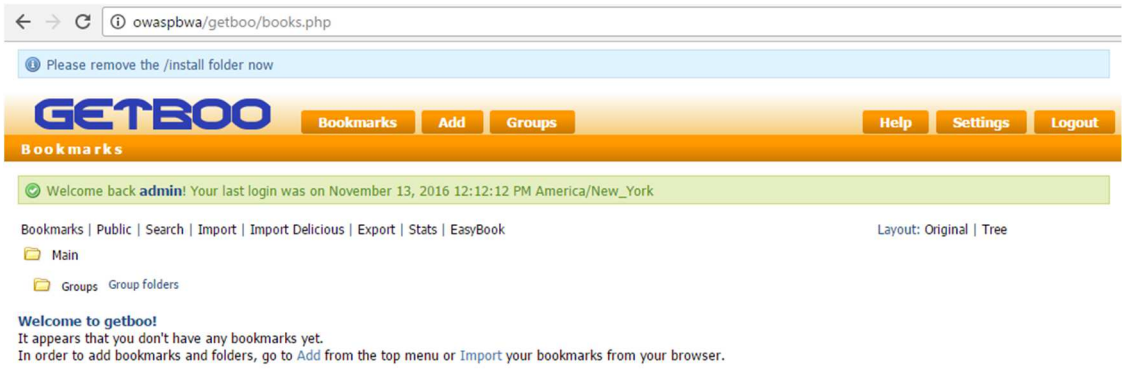
Figura 9: Senha do usuário



Fonte: *print screen* da aplicação Ferramenta *HashToolKit*

Após desvendar a senha do usuário Admin, no caso a senha era o mesmo nome do usuário, foi possível comprovar que o mesmo estava correto, entrando o usuário e senha no sistema web, conforme demonstrado na Figura 10.

Figura 10 - Entrando na aplicação



Fonte: *print screen* da aplicação Máquina Virtual OWASP

Este tipo de ataque causou um grande estrago, pois todo o banco de dados foi interceptado e qualquer informação o atacante poderia usufruir. Isso poderia ter sido facilmente evitado, se o programador que desenvolveu este aplicativo tivesse se atentado as recomendações de segurança quando criou o aplicativo.

Como demonstrado e recomendado pela OWASP no capítulo 5.2 Injeção de SQL, para evitar este tipo de ataque é necessário validar os parâmetros de entrada da requisição, sessão do usuário e se a requisição partiu de uma origem confiável.

7 CONSIDERAÇÕES FINAIS

Após a era computacional a informação é um dos bens mais valiosos de uma empresa ou pessoa física. Nela estão contidos dados valiosos e sigilosos que podem ser utilizados com má intenção, por *hackers*. Esses atacantes buscam obter qualquer tipo de informação ou até mesmo deixar um serviço indisponível. Existem vários motivos que levam os *hackers* a realizar esses ataques. Pode ser um concorrente que deseja obter uma informação privilegiada, atrapalhar os negócios, ou até mesmo um ataque a uma instituição governamental a fim de realizar algum tipo de protesto.

Segurança da Informação é primordial para manter a saúde da empresa ou pessoal, com relação aos dados e conhecimentos que são armazenados ou informações privilegiadas existentes. Atualmente existem várias formas de proteger essas informações, com uma infraestrutura segura utilizando *Firewall*, antivírus, *proxy*, entre outros tipos de proteções disponíveis no mercado.

De nada irá adiantar investir na infraestrutura da empresa e proteção se o *software* utilizado pela empresa pode conter falhas graves de segurança. Esses sistemas podem ser a porta de entrada para roubo de informações, alteração de dados, ou até mesmo ataques para deixar algum serviço primordial da empresa indisponível.

Como foi possível constatar, existe uma estatística histórica de que os programadores de aplicações *web*, não se preocupam com o quesito Segurança da Informação no momento de desenvolver um *software.* Nas duas últimas avaliações da OWASP ficou comprovado que sete dos dez itens da lista “Top 10 OWASP” continuaram os mesmos.

O grande problema desses ataques é que são fáceis de serem efetuados e normalmente fazem um grande estrago para a empresa ou pessoa alvo. Existem ferramentas especializadas para cada tipo de ataque feito pelos *hackers*. Essas ferramentas efetuam várias requisições, atingindo o seu objetivo em pouco tempo de execução.

Uma das formas de proteger uma aplicação contra qualquer tipo de ataque é seguir as recomendações da OWASP no momento do desenvolvimento do *software*. Garantindo assim uma maior proteção das informações confidenciais. Ademais, o programador deve sempre atualizar seus conhecimentos na área de segurança da informação, pois, os ataques estão sempre sendo inovados. Por tanto é de suma

importância sempre pesquisar novas formas de ataques, e como se proteger contra os mesmos.

A OWASP é uma das mais conceituadas entidades que realizam esse tipo de pesquisa e análise, e indica índices sobre quais os ataques amplamente utilizados, Divulgando uma lista sobre os principais ataques a cada três anos. Esta é uma das formas do desenvolvedor se manter atualizado com relação à Segurança em Aplicações *Web,* e realizar práticas para proteção de suas aplicações*.*

Através do estudo de caso foi possível demonstrar a facilidade de ataque à uma aplicação web desprotegida. Isto ficou comprovado com o ataque a uma aplicação vulnerável, permitindo a extração de qualquer tipo de informação deste sistema. Desta forma um hacker poderia obter qualquer informação que precisasse e utilizá-la de forma a ganhar alguma vantagem ou causar prejuízo à organização. É muito importante que o programador se preocupe com a segurança da aplicação durante o seu desenvolvimento. Se a aplicação tivesse sido criada seguindo as recomendações de segurança adequadas este ataque poderia ter sido evitado.

Vale ressaltar e salientar a importância e necessidade do desenvolvimento de aplicações seguras, com técnicas de segurança próprias, e demasiado cuidado na sua criação. Diferente do que prega o senso comum, uma aplicação segura é muito mais valiosa do que uma aplicação com seu *front-end* elegante e bonito. Todo desenvolvedor deve se atualizar e despender parte do seu prazo de desenvolvimento na elaboração de códigos seguros e protegidos, a fim de evitar possíveis exposições dos dados.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BLUE PHOENIX. **Boas práticas de segurança**. Disponível em: <http://www.bluephoenix.pt>. Acessado em: 17 out. 2015.

CELSO, Anderson. **Segurança da Informação em Empresas.** Disponível em: <http://www.administradores.com.br/mobile/artigos/carreira/seguranca-da-informacao-em-empresas/58918/>. Acessado em: 16 nov. 2016.

DANTAS, Marcos L. **Segurança da informação**: uma abordagem focada em gestão de riscos. Olinda: Livro Rápido, 2011.

IBM. **Relatório IBM X-Force revela que ataques a redes móveis dobrarão em 2011**. Disponível em: <https://www-03.ibm.com/press/br/pt/press release/ 35818.wss>. Acessado em: 08 dez. 2016.

KEARY, Eoin. ***Authentication Cheat Sheet***. Disponível em: <https://www.owasp.org/index.php/Authentication_Cheat_Sheet>. Acessado em: 03 out. 2016.

KLEINSORGE, Cláudia R P. **Efetividade dos sistemas de informação nas organizações**. Belo Horizonte, 2015. Disponível em: <http://www.fumec.br/revistas/sigc/article/viewFile/3320/1897>. Acesso em: 15 out. 2015.

MACHADO, Francis B. **Sistemas Operacionais.** 4 ed, Rio de Janeiro: Grupo Editorial Nacional, 2007.

MARCIANO, João L P. **Segurança da informação**: uma abordagem social. Brasília, 2006. Disponível em: <http://repositorio.unb.br/bitstream/10482/1943/1/Jo%C3%A3o%20Luiz%20Pereira%20Marciano.pdf>. Acesso em: 11 out. 2015.

MAFIOLETTI, Rafael**. Uso do PFSENSE para o controle de acesso em uma rede local**. Lages, 2012. Disponível em: <https://revista.uniplac.net/ojs/index.php/tc_si/article/viewFile/917/627>. Acesso em 19 out. 2015.

MEUCCI, Matteo. **OWASP Testing Project**. 2014. Disponível em: <https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents. Acessado em: 18 out. 2015

MITNICK, Kevin; Simon, Willian L. **A Arte de invadir**, Prentice Hall, 2006

MONTEIRO, Mario A. **Introdução à Organização de Computadores**. 4 ed, Rio de Janeiro: Grupo Editorial Nacional, 2002.

OWASP. **The free and open software security community**. Disponível em: <https://www.owasp.org/index.php/Main_Page>. Acessado em: 01 mai. 2016

OWASP. **Top 10 – 2004**: Os Dez riscos de segurança mais críticos em aplicações *web.* Disponível em: <https://www.owasp.org/index.php/Top_10_2004>. Acessado em: 03 out. 2016

OWASP. **Top 10 – 2007:** Os Dez riscos de segurança mais críticos em aplicações *web.* Disponível em: <https://www.owasp.org/index.php/Top_10_2007>. Acessado em: 03 out. 2016

OWASP. **Top 10 – 2010**: Os Dez riscos de segurança mais críticos em aplicações *web.* Disponível em: <https://www.owasp.org/index.php/Top_10_2010-Main>. Acessado em: 03 out. 2016

OWASP. **Top 10 – 2013:** Os Dez riscos de segurança mais críticos em aplicações *web*. Disponível em:https://www.owasp.org/images/9/9c/OWASP_Top_10_2013_PT-BR.pdf>. Acessado em: 02 mai. 2016

OWASP. **Broken Web Applications Project**. Disponível em:

<https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project#tab=Main>. Acessado em: 12 nov. 2016

OWASP. ***SQL Injection Prevention CheatSheet*.** Disponível em: <https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet>. Acessado em: 03 out. 2016

SOEIRA, Bárbara V; SILVA, Fernando C, TABORDA, Letícia B. **Implementação do *Firewall* CISCO ASA 5500**. Curitiba, 2013. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2063/1/CT_COTEL_2013_1_09.pdf>. Acessado em: 17 nov. 2015

STOCK, Andrew. **OWASP code review guide**, 1 ed, 2008. Disponível em: <https://www.owasp.org/images/2/2e/OWASP_Code_Review_Guide-V1_1.pdf>. Acessado em: 17 nov. 2016.

UTO, Nelson. **Teste de Invasão de Aplicações Web**. 1 ed. Rio de Janeiro: Escola de Redes, 2013.

WIESMANN, Adrian; STOCK, Andrew; CURPHEY, Mark; STIRBEI, Rray. **A Guide to Building Secure Web Applications and Web Services**. 2005. Disponível em: <https://www.um.es/atica/documentos/OWASPGuide2.0.1.pdf>. Acessado em: 11 dez. 2015