



Fundação Educacional do Município de Assis
IMESA - Instituto Municipal de Ensino Superior de Assis

ANTONIO CARLOS GONÇALVES JUNIOR

**UM ESTUDO DE SEGURANÇA DA INFORMAÇÃO:
INJEÇÃO DE SQL**

Assis

2013

ANTONIO CARLOS GONÇALVES JUNIOR

**UM ESTUDO DE SEGURANÇA DA INFORMAÇÃO:
INJEÇÃO DE SQL**

Trabalho de Conclusão de Curso apresentado ao Instituto
Municipal de Ensino Superior de Assis, como requisito do
Curso de Análise e Desenvolvimento de Sistemas.

Orientador: Msc. Fábio Eder Cardoso

Área de Concentração: Web Site

Assis

2013

FICHA CATALOGRÁFICA

GONÇALVES JUNIOR, Antonio Carlos.

Um estudo de segurança da informação: Injeção de sql / Antonio Carlos Gonçalves Junior.
Fundação Educacional do Município de Assis – FEMA – Assis,2013.

PG

Orientador: Fábio Eder Cardoso

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

1. Injeção de SQL. 2. Segurança na Web.

CDD:001.61
Biblioteca da FEMA

UM ESTUDO DE SEGURANÇA DA INFORMAÇÃO: INJEÇÃO DE SQL

ANTONIO CARLOS GONÇALVES JUNIOR

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Prof^o Msc. Fábio Eder Cardoso

Analisador: Prof^o Dr. Almir Rogério Camolesi

Assis

2013

AGRADECIMENTOS

Primeiramente à Deus,

Por nunca me desamparar, e ajudar a superar cada obstáculo em minha vida.

Ao Prof. Fábio Eder Cardoso,

Pela orientação e pelo constante estímulo transmitido durante o trabalho.

Aos meus pais Andrelisa e Antonio Carlos,

Por me propiciarem a vida e os recursos necessários para trilhar o caminho do conhecimento e da educação.

À minha namorada Ana Carolina,

Ofereço um agradecimento mais do que especial, por ter vivenciado comigo passo a passo todos os detalhes deste trabalho, por ter me oferecido todo apoio que necessitava nos momentos difíceis, todo carinho, respeito, por ter me aturado nos momentos de estresse, e por tornar minha vida cada dia mais feliz.

RESUMO

Com o avanço tecnológico, o acesso às redes é primordial. Atualmente é indispensável o uso da internet. A cada dia nos tornamos cada vez mais dependentes desse mundo online. Fazemos, por exemplo, inscrições de vestibulares, compras, agendamento de consultas médicas, pesquisas e, podemos até mesmo acessarmos nossas próprias contas bancárias. Por desfrutar dessa comodidade precisa-se depositar total confiança nesses sistemas, portanto os métodos de segurança na *web* nem sempre estão aptos a deter ameaças. Em vista desse cenário esse estudo tem como objetivo, práticas para inibir esses ataques, o tornando mais confiável para utilização.

No presente estudo, foi utilizado um código PHP vulnerável a Injeção de SQL, para demonstração de forma didática em um site anônimo, e o software *Acunetix* com finalidade de demonstrar as vulnerabilidades à Injeção SQL e outras, com o propósito de tornar o site mais seguro e confiável.

Palavras-chave: Injeção SQL; Segurança na web.

ABSTRACT

With the technological advancement, access to networks is primordial. Nowadays it is indispensable the usage of the internet. Day by day we become more and more dependent of this online world. We do, for example, registrations for vestibulares, shopping, schedule medical consultations, researches and, we can even access our own bank accounts. By enjoying this convenience we need to deposit total confidence in these systems, although the methods of security on the web do not always know how to stop threats. Looking forward this scenario this study has as objective, practices to inhibit these attacks, making it most reliable for use. In this study, it was used a code PHP vulnerable to SQL Injection, for demonstration of how didactics in an anonymous site and software *Acunetix* with the purpose of demonstrating the vulnerabilities SQL Injection and other, with the purpose of making the site more secure and reliable.

Keywords: SQL Injection; Web Security.

LISTA DE ILUSTRAÇÕES

Figura 1 - Incidentes Reportados ao CERT.br.....	24
Figura 2 - Formulário de Login.....	26
Figura 3 - Código PHP.....	25
Figura 4 - Injeção SQL.....	27
Figura 5 - Resultado da Injeção SQL.....	27
Figura 6 - Deletando informações do banco de dados.....	28
Figura 7 - Código PHP corrigido.....	29
Figura 8 - Injeção de SQL sem êxito.....	29
Figura 9 - Nível de segurança do website.....	31
Figura 10- Alertas encontrados.....	32

LISTA DE TABELAS

Tabela 1 - Comandos Básicos Da Linguagem SQL	17
Tabela 2 - Cláusulas da Linguagem SQL no comando <i>SELECT</i>	26
Tabela 3 - Operadores da Linguagem SQL.....	28
Tabela 4 - Informações do <i>Web Site</i>	29

LISTA DE ABREVIATURAS E SIGLAS

SQL	Structure Query Language
URLs	Uniform Resource Locator
Dos	Denial of Service
CSRF	Cross-Site Request Forgery
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
SSH	Secure Shell
HTTPD	Hypertext Transfer Protocol Daemon
CPU	Central Processing Unit
RAM	Random Access Memory
IP	Internet Protocol

SUMÁRIO

1.	INTRODUÇÃO.....	12
1.1	OBJETIVOS.....	12
1.2	JUSTIFICATIVA.....	13
1.3	ESTRUTURA DO TRABALHO.....	13
2.	LINGUAGEM SQL.....	15
3.	O QUE É SQL INJECTION.....	17
4.	SEGURANÇA NA WEB.....	18
4.1	WEB SERVICES.....	18
4.2	ATACANTES.....	19
5.	VULNERABILIDADES.....	21
5.1	TIPOS DE VULNERABILIDADES NA WEB.....	21
5.2	SOFTWARES PARA PENTEST.....	23
6.	SEGURANÇA.....	25
7.	DEMONSTRAÇÃO DE SQL INJECTION.....	26
7.1	EVITANDO ATAQUE SQL INJECTION.....	29
7.2	RASTREANDO VULNERABILIDADES.....	30
8.	MATERIAIS E MÉTODOS.....	40
9.	RESULTADO.....	41
10.	CONSIDERAÇÕES FINAIS.....	42
	REFERÊNCIAS.....	43

1. INTRODUÇÃO

O presente trabalho trata de uma pesquisa bibliográfica sobre injeção de SQL. Com o acesso massivo às estruturas organizadas de dados, principalmente pela *Internet*, o fator segurança torna-se primordial, visto que sem ela, os sistemas apresentam-se vulneráveis e a mercê de pessoas maldosas. Uma das modalidades de exploração destas vulnerabilidades é o ataque por injeção de SQL, ou *SQL Injection*, tema deste trabalho.

A linguagem SQL (*Structured Query Language*) define-se como uma linguagem declarativa em oposição as outras linguagens procedurais, uma vez que apresenta maior rapidez e usabilidade em relação às outras.

Segundo Ferreira *et al.* (2005):

SQL é formada basicamente por duas sub-linguagens: • Linguagem de definição de dados (SQL DDL): fornece comandos para definir e modificar esquemas de tabelas, remover tabelas, criar índice e definir restrições de integridade. • Linguagem de manipulação de dados (SQL DML): fornece comandos para consultar, inserir, modificar e remover dados no banco de dados.

A injeção de SQL, mais conhecida por *SqllInjection*, é uma forma de teste de vulnerabilidade que se aproveita de falhas de segurança presentes nas bases de dados. É um ataque que visa enviar comandos até base de dados por meio de campos de formulários ou através de URLs (*Uniform Resource Locator*). Um ataque bem sucedido pode apagar uma tabela do banco de dados, apagar todos os dados da tabela ou até subtrair, de forma indevida, senhas que estejam cadastradas.

1.1 OBJETIVO

O presente trabalho tem como objetivo, reconhecer as vulnerabilidades, analisar como impedir os ataques, demonstrar, de forma didática e objetiva, um ataque por meio de um site vulnerável e por fim aplicar as formas de segurança para inibir os mesmos resultando na melhorada segurança de todos os sites na rede. Comprovar para todos os usuários as falhas que podem ser exploradas quando certos cuidados, como má codificação da linguagem SQL, são ignorados. A pretensão desse trabalho é comprovar que há necessidade de obter mais estudos sobre segurança na web.

1.2 JUSTIFICATIVA

Segundo a convivência do autor em acessos em seu cotidiano foi identificado vários sites vulneráveis, correndo risco de sofrer vários tipos de ataques. Mesmo com o avanço tecnológico, ainda existem inúmeras vulnerabilidades na web, tornando sistemas que utilizam a linguagem SQL inseguras. Portanto, é de extrema importância que haja conhecimento dos desenvolvedores, no tocante à segurança, pois muitos sistemas correm risco de ser reconfigurados. Além desse motivo e de outras experiências, dados sobre vulnerabilidades e à citação abaixo contribui para realização desta pesquisa para conclusão do curso de Análise e Desenvolvimento de Sistemas na Fundação Educacional do Município de Assis.

De acordo com Teixeira (2007), ano após ano as vulnerabilidades crescem, entretanto é necessário lançar técnicas que possibilitam maneiras para contribuir na diminuição dos erros nas aplicações.

1.3 ESTRUTURA DO TRABALHO

Este trabalho foi subdividido em sete capítulos a serem explicados a seguir.

No primeiro capítulo foi apresentada a contextualização, o objetivo e a justificativa para o desenvolvimento do trabalho.

Já no segundo capítulo serão abordados os conceitos teóricos da Linguagem SQL e seus comandos básicos.

O terceiro capítulo apresenta a definição de SQL Injection.

O quarto capítulo descreve a segurança na web, sequenciamento dos ataques existentes.

No quinto capítulo será apresentado o conceito de vulnerabilidades, seguida dos tipos de vulnerabilidades na *web*, e *softwares* para *pentest*.

O sexto capítulo tratará sobre segurança desses ataques.

No sétimo capítulo será demonstrado um ataque de SQL Injection de forma didática, as maneiras de se evitar tais ataques, e um rastreamento de vulnerabilidades.

Já no oitavo capítulo descreve os materiais e métodos para desenvolver esse trabalho.

No nono capítulo, uma breve discussão sobre as vulnerabilidades de um site real e os resultados obtidos.

No décimo e último capítulo será apresentada a conclusão dessa pesquisa.

2. LINGUAGEM SQL

A linguagem SQL foi criada nos anos 70 nos laboratórios da IBM, a linguagem é um grande padrão de banco de dados, por ser simples de fácil uso. A forma que o SQL utiliza a consulta é específica ao resultado, ao contrário das outras linguagem que aponta o caminho para chegar até o mesmo.

Para se entender como funciona um ataque por *SQL Injection*, é indispensável ter a ciência de pelo menos um pouco dos comandos básicos da linguagem SQL padrão (RACCIATTI, 2002, p.7).

Comando	Definição
<i>ALTER</i>	Utilizado para alterar tabelas agregando campos e trocar a definição de campos.
<i>CREATE</i>	Utilizado para criar novas tabelas, campos e índices.
<i>DELETE</i>	Utilizado para apagar registros de uma tabela da base de dados.
<i>DROP</i>	Empregado para eliminar tabelas e índices.
<i>INSERT</i>	Utilizado para inserir dados em uma tabela.
<i>SELECT</i>	Utilizado para consultar registros de uma base de dados que satisfaçam um determinado critério.
<i>UPDATE</i>	Utilizado para modificar os valores dos campos e registros especificados.

Tabela 1 - Comandos básicos da linguagem SQL (In: Técnicas de SQL Injection: Un Repaso, 2002)

Comando	Definição
<i>FROM</i>	Utilizado para especificar a tabela a qual serão selecionados os objetos.
<i>WHERE</i>	Utilizado para especificar as condições que devem reunir os registros que serão selecionados.
<i>GROUP BY</i>	Utilizado para separar os registros escolhidos em grupos específicos.
<i>HAVING</i>	Utilizado para especificar a condição que deve satisfazer cada grupo.
<i>ORDER BY</i>	Utilizado para selecionar os registros selecionados de acordo com uma ordem específica.
<i>UNION</i>	Utilizado para unir o resultado de duas consultas em um só resultado, este comando gera a união de conjuntos.

Tabela 2 - Cláusulas da linguagem SQL no comando *SELECT* (In: Técnicas de SQL Injection: Un Repaso, 2002)

Além dos comandos e cláusulas, são indispensáveis operadores (Tabela 3) para efetuar operações matemáticas, comparações e expressões booleanas.

Comando	Definição
>	Maior que
<	Menor que
<>	Diferente de
>=	Maior ou igual a
<=	Menor ou igual a
=	Igual a
<i>AND</i>	Operador booleano para testar se mais de uma condição é verdadeira ou falsa
<i>OR</i>	Operador booleano para testar se pelo menos uma das condições é verdadeira ou falsa.
<i>BETWEEN</i>	Utilizado para especificar um intervalo de valores.
<i>LIKE</i>	Utilizado para especificar um intervalo de valores.
<i>IN</i>	Utilizado para especificar registros de uma base de dados.

Tabela 3 - Operadores da linguagem SQL (In: Técnicas de SQL Injection: Un Repaso, 2002)

3. O QUE É SQL INJECTION?

Sqlinjetcion e uma forma de ataque que explora as falhas de segurança que se interagem no banco de dados. O desconhecimento, por parte dos programadores, sobre práticas de implementações, resulta na vulnerabilidade do sistema, permitindo aos Hackers, ataques via *SQLInjection*.

Segundo Microsoft (2013),

Injeção SQL é um ataque no qual um código mal-intencionado é inserido em cadeias de caracteres que são passadas posteriormente para uma instância do SQL Server para análise e execução. Qualquer procedimento que construa instruções SQL deve ser verificado quanto a vulnerabilidades de injeção porque o SQL Server executará todas as consultas sintaticamente válidas que receber. Mesmo dados com parâmetros podem ser manipulados por um invasor qualificado e determinado.

A forma principal de injeção SQL consiste em inserção direta de código em variáveis de entrada de usuário, concatenadas com comandos SQL e executadas. Um ataque menos direto injeta código mal-intencionado em cadeias de caracteres destinadas a armazenamento em uma tabela ou como metadados. Quando as cadeias de caracteres armazenadas são concatenadas subseqüentemente em um comando SQL dinâmico, o código mal-intencionado é executado.

O processo de injeção funciona encerrando prematuramente uma cadeia de caracteres de texto e anexando um novo comando. Como o comando inserido pode ter cadeias de caracteres adicionais anexadas a ele antes de ser executado, o malfeitor encerra a cadeia de caracteres injetada com uma marca de comentário "--". O texto subseqüente é ignorado no momento da execução.

4. SEGURANÇA NA WEB

Segundo ABNT NBR ISO (2005):

A informação é um ativo que, como qualquer outro ativo importante, é essencial para os negócios de uma organização e conseqüentemente necessita ser adequadamente protegida. Isto é especialmente importante no ambiente dos negócios, cada vez mais interconectado. Como um resultado deste incrível aumento da interconectividade, a informação está agora exposta a um crescente número e a uma grande variedade de ameaças e vulnerabilidades.

Hoje em dia a segurança das informações são primordiais para qualquer tipo de sistemas web, seja ela em qualquer linguagem programáveis.

Pois muitas informações hoje em dia são extremamente secretas, exemplo: dados bancários, cartão de créditos e etc. E com isso muitos sistemas tentam cada vez mais deixar o ambiente totalmente seguro, mas ainda assim os Hackers ainda acham vulnerabilidades para realizar os ataques.

4.1 *WEB SERVICES*

Segundo Wangham, (2006):

Há tempos que a Internet se consolidou como um importante veículo de comunicação e não demorou muito para se tornar uns dos principais meios para a realização de negócios. A Internet também é conhecida por agregar os mais diversos sistemas computacionais que variam desde a arquitetura de máquina, sistema operacional até os aplicativos finais aos usuários. O sucesso deste ambiente tão heterogêneo foi possível devido ao uso de protocolos padronizados, que garantem a interoperabilidade entre as aplicações, não importando em qual sistema operacional ou arquitetura de máquina esta esteja rodando.

As *Webs Services* formam um novo padrão de compartilhamento de dados que recebe publicação de usos e técnicas acessíveis pela Internet. Com uma interconexão simples para o cliente e fácil integração de aplicações diferentes, os Web Services são uma importante ferramenta para causar a conexão dos dados em plataformas distintas, incluindo os dispositivos móveis.

Antes da existência da Internet as empresas já utilizavam os sistemas computacionais, portanto, foram desenvolvidos afim de não implementação de sistemas distintos, exemplo, os sistemas computacionais com seus clientes fornecedores, etc. Perante a necessidade da ligação entre as aplicações lançadas de organizações desiguais, surgiu uma nova caracterização de sistemas permitindo a troca de informações e relação com os sistemas existentes (WANGHAM, 2006).

4.2 ATACANTES

Os atacantes, mais conhecidos como “*Hackers*” são responsáveis por explorar vulnerabilidades para benefício próprio ou não. Dentre os *hackers* existem vários, alguns deles são: *Preackers*, *Script Kiddies* e *Crackers*.

Para Souza (2007) *Preackers* é responsável pelas fraudes nas telefonias, ultimamente o alvo mais comum é a telefonia móvel; *Script Kiddies*: maior índice de ataques, portanto com baixo nível de conhecimento, que usam instrumentos de fácil acesso na internet; *Crackers*: possuem um grande conhecimento, com capacidade de anular segurança de informações, apagando sistemas e subtraindo bancos e instituições financeiras. *Carders*: responsáveis por fazer compras pela internet utilizando cartões de créditos roubados, ou mesmo clonando números de cartões através de softwares; *Insiders*: colaboradores ou ex-colaboradores insatisfeitos de empresas que têm como objetivo roubar informações configurando espionagem ou mesmo agindo de forma a destruir sistemas e processos dos quais conhece.

De acordo com Souza (2007), a lista abaixo apresenta os ataques mais comuns:

Ataque ao nível da aplicação: explora as vulnerabilidades dos *softwares*;

Ataque a servidor web: caracteriza-se pela exploração de vulnerabilidades em *softwares* para publicação de páginas web como o *Internet Information Server*

Buffer-overflow: falha no controle da área de armazenamento temporário em memória RAM, também conhecidos como explosão de pilha com efeitos de falhas em aplicações deixando-as indisponíveis;

Exploit: *software* de código maldoso que explora vulnerabilidades dos mais diversos *softwares* como *Internet Explorer*;

SQL-Injection: é uma método que possibilita a inclusão de um código na linguagem SQL em páginas *web*, permitindo ao invasor o acesso ao servidor de banco de dados;

Trojan ou Cavalo de Tróia: executa funções sem que o usuário invadido tenha conhecimento, admitindo que o computador seja explorado pelos invasores;

Vírus: a forma de ataque mais conhecida, é *software* que faz cópia de si próprio, ocasionando vários problemas, tais como o mau funcionamento de *softwares*;

Worm: *software* que não necessita ser executado para ser usado. Fornece dados que são transmitidas a *hackers* de modo secreto, sendo na maioria dos casos de modo imperceptível ao usuário;

Ataque físico: caracterizam-se pelo roubo de aparelhamentos discos, fitas magnéticas, CD-ROM, disquetes ou outros elementos de armazenamento de dados que são retirados da empresa para posterior análise ou destruição;

Denial of Service: ataque de negação de serviço, culpado por sobrecarregar servidores com amplo volume de dados, originando a parada do sistema operacional, gerando o preenchimento da memória do computador e a sobrecarga de operações do processador;

Packet Sniffing: *software* que realiza a captura de pacotes IP que podem dominar informações importantes de *softwares* de bate-papo ou mesmo *softwares* de *e-mail* como o *Outlook Express*;

Scan: também conhecido como *Port Scanning*, avalia portas IP que têm serviços associados, como por exemplo, *telnet*. Entre os *softwares* mais usados para esta tarefa estão o *Nmap*.

5. VULNERABILIDADES

Vulnerabilidades são más codificações de várias origens.

Conforme Sêmola (2003 apud SOUZA, 2007, p.24):

As vulnerabilidades podem ter origens diversas como apresentado na lista sugerida por SÊMOLA:Agentes da natureza umidade, poeira, poluição e calor podem causar danos aos ativos. Deve-se levar em consideração também fatores geográficos que possam resultar em ameaças. Por exemplo, instalações próximas a rios que causam inundações;Hardwares: falhas no dimensionamento do equipamento a ser utilizado, problemas de projeto e manutenção; Softwares: falhas no desenvolvimento que permitem a inclusão e execução de softwares com código malicioso. Mídias de armazenamento: falhas de fabricação ou estocagem de CD-ROM, discos rígidos, DVD-ROM entre outros. Meios de comunicação: problemas no cabeamento, antenas de rádio inadequadas entre outros problemas na infra-estrutura de comunicação. Humanas: relativas aos danos que o ser humano pode causar às informações quando de espionagem, má utilização e acidentes derivados da falta de treinamento, insatisfação com o trabalho, erros, dentre outros fatores.

Porém a vulnerabilidade de injeção SQL é de *softwares* onde se encontra falhas no desenvolvimento que aceita a inserção e execução de *softwares* com códigos maldosos.

5.1 TIPOS DE VULNERABILIDADES NA WEB

Todo profissional de segurança da informação sabe que as vulnerabilidades em aplicações *web* são muitos comuns hoje em dia. Logo abaixo apresentaremos uma breve descrição do funcionamento de algumas vulnerabilidades mais comuns que normalmente são encontradas nas aplicações *web*.

SQL Injection: as vulnerabilidades de SQL Injection acontece quando é permitido a entrada de caracteres especiais que alteram o fluxo da consulta SQL no banco de dados;

CSRF (Cross-Site Request Forgery): ocorre quando é possível alterar na requisição do usuário, que executara uma determinada ação como sendo o próprio usuário;

XSS (Cross-Site Scripting): injeta códigos na aplicação web vulnerável para obter informações sensíveis, como senhas, nomes de usuários, etc.

Reflected XSS: quando recebe entradas do usuário em uma pagina web que exhibe seu conteúdo;

Stored XSS: ocorre quando pode-se armazenar o código malicioso dentro da aplicação web vulnerável de forma estável;

DOM-based XSS: acontece quando o usuário clica em uma URL maliciosa;

Format String: abrange os programas executáveis que estão sendo usado em aplicações web;

Directory Traversal: a vulnerabilidade pode estar presente em uma aplicação que não permite acesso à usuários que leiam ou acessem arquivos no sistema, mas não permite identificar corretamente permissões que o usuário tem no sistema;

Command Injection: esta vulnerabilidade ocorre quando a aplicação executa comandos no sistema do servidor e como estes comandos não são verificados, é possível que seja inseridos comandos no sistema com objetivo de obter informações e ganhar acesso total ao sistema do servidor;

Privilege Escalation: o Ataque visa à escalação de privilégios, ocorre quando é utilizado um erro de lógica na aplicação, com a finalidade de obter privilégios para executar tarefas que antecipadamente não estaria autorizado.

Estas vulnerabilidades acontecem muitas vezes em aplicações que usam funções para múltiplos usuários (FERREIRA, 2013).

5.2 SOFTWARES PARA PENTEST

Existe vários software para ajudar nos recursos de scanner, achar vulnerabilidades e até exploraras. Porém existem que usam para ajudar a melhorar a segurança dos sites, e por outro lado pessoas mal intencionadas para obter benefícios com esses recursos.

Blind Cat

Blind SQL Injection é um tipo de ataque que destrói as vulnerabilidades mais conhecidas na investigação dos programadores desavisados e após vai aumentando o grau de complexidade (WEBCHEATS,2013).

Sqlmap

É uma instrumento open source que automatiza o processo de detecção e exploiting de vulnerabilidades a Sql Injection. Apoio para Banco de Dados - Mysql, Oracle, PostgreSQL, Msql, Microsoft Acess, DB2 , Informix, Sybase e InterbasE (WEBCHEATS,2013).

Havij v1.12(Windows)

Esse programa tem a vista limpa, bem eficiente e rápida, pois e só pegar o endereço *web* (URL) o qual quer identificar as vulnerabilidades e avaliar os erros no banco de dados que poderia ser aproveitado por pessoas maliciosas. Detecta automaticamente qual database. Entende automaticamente qual termo equivale a um resultado positivo e negativo. Tenta distintos tipos de sintaxe na injeção. Suporta injeção manual. Tenta adivinhar tabelas e colunas em mysql<5 e no MsAccess também. Recurso de entender a página de admin. Recurso para quebrar senhas md5(faz a busca em vários sites pelo hash) (WEBCHEATS,2013).

Sql Ninja

É uma instrumento *free* que serve para fazer SQL *Injections* em aplicações *web* que utilizam banco de dados *Microsoft SQL Server*. Programa bem conhecido e forte que por sinal faz parte da suíte de aplicativos da distribuição *linux Backtrack*. *Bruteforce* da conta do SA(com dois tipos de ataque: dicionário e incremental). *Upload* do *netcat* (ou algum outro praticável) usando normal HTTP *requests* após um ataque bem contínuo. TCP/UDP *portscan* para o SQL *Server*, para tentar achar uma porta que está aberta pelo firewall e usar essa porta para uma reverse Shell(WEBCHEATS,2013).

Acunetix

É um *scanner* de *site*, em busca de vulnerabilidades. O *scanner* procura falhas críticas e leves, entre os ataques que ele verifica se o site está vulnerável estão: Dados, sql injection, php injection entre outros. É utilizado tanto por *crackers* quanto desenvolvedores de site para verificar eventuais falhas. No caso dos desenvolvedores, para corrigi-las, enquanto os *crackers*, para aproveitá-las e efetuar uma invasão(WEBCHEATS,2013).

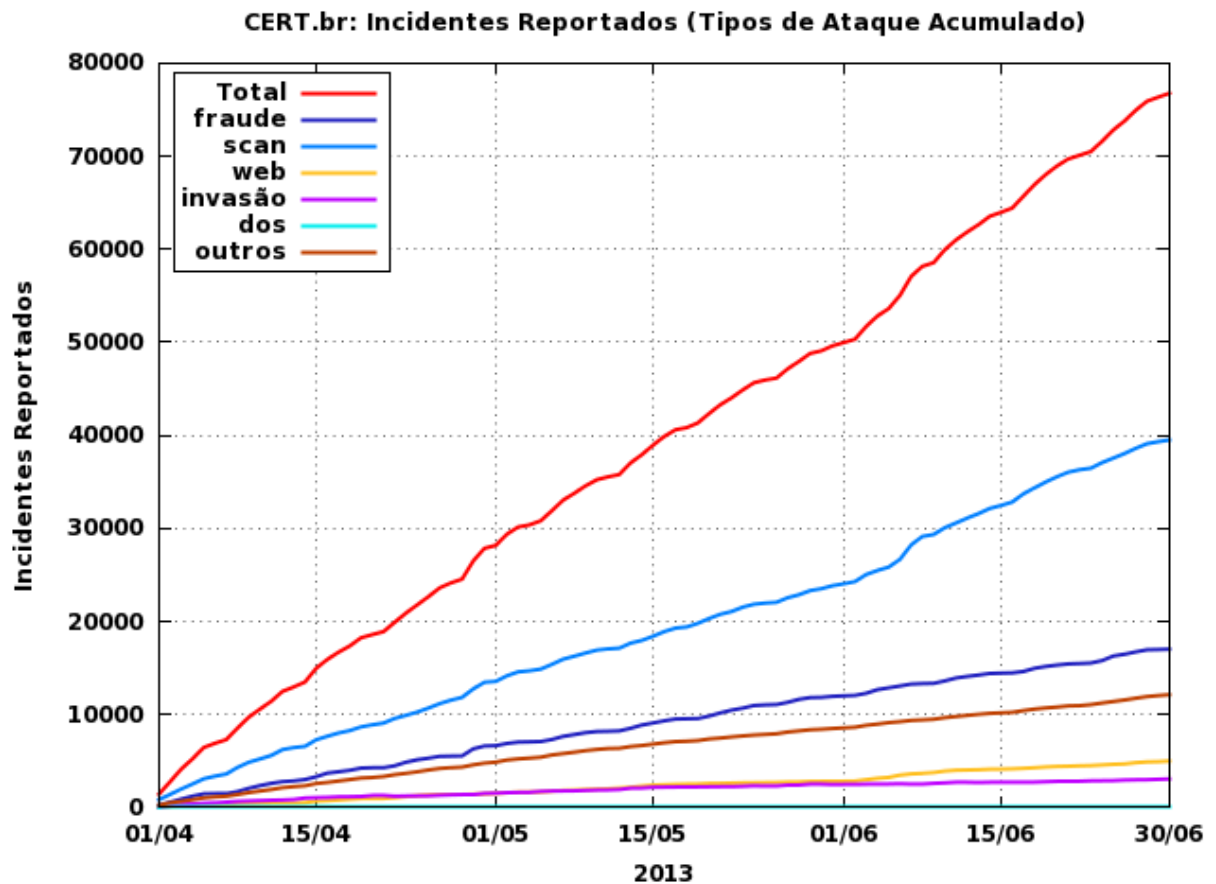


Figura 1 - Incidentes Reportados ao CERT.br -- Abril a Junho de 2013 (In: CERT.br. Disponível em <http://www.cert.br/estatisticas>. Acesso: 02 nov. 2013)

6. SEGURANÇA

Hoje em dia basta ligar algum dispositivo que tenha acesso a *Internet*, para se tornar vulnerável a ataques.

Assim que o dispositivo é iniciado, enfrenta perigos vindos da internet, incluindo ataques de *spyware*, *vírus*, cavalos de tróia, *SQL injection*, e *hackers* tentando criar uma conexão direta com o seu computador.

Mas existem recursos e *softwares* para melhorar e até inibir esses perigos, como:

Firewall: é um tipo de *software* que pode ser dividido em diversas categorias, mas que tem sempre o mesmo objetivo, que é o de não permitir a entrada de pacotes IP e, deste modo, contendo ameaças.

Filtro de Pacote: usa regras estáticas para filtrar pacotes que têm origem em servidores externos. É muito comum e considerado simples de ser configurado.

Proxy: este tipo de firewall tem por intenção filtrar os pacotes que são causados na rede interna da empresa (*LAN*) e muitas vezes ele evita a conexão com servidores externos que podem ser prejudiciais ao sistema de informação.

Firewall pessoal: *software* que intercepta as conexões de entrada e saída em um computador. Baseia-se em regras padrão ou definidas pelo usuário e define quais conexões podem ser recebidas e quais devem ser rejeitadas.

Firewall reativo: há funções que admitem adotar ataques e emitir alarmes quando encontrar sequências de pacotes IP chamadas de assinaturas e impede o acesso impróprio automaticamente (SOUZA, 2007).

7. DEMONSTRAÇÃO DE SQL INJECTION

Á seguir será apresentado com *prints screen*, uma demonstração de *SQL Injection* com um formulário de *login*:

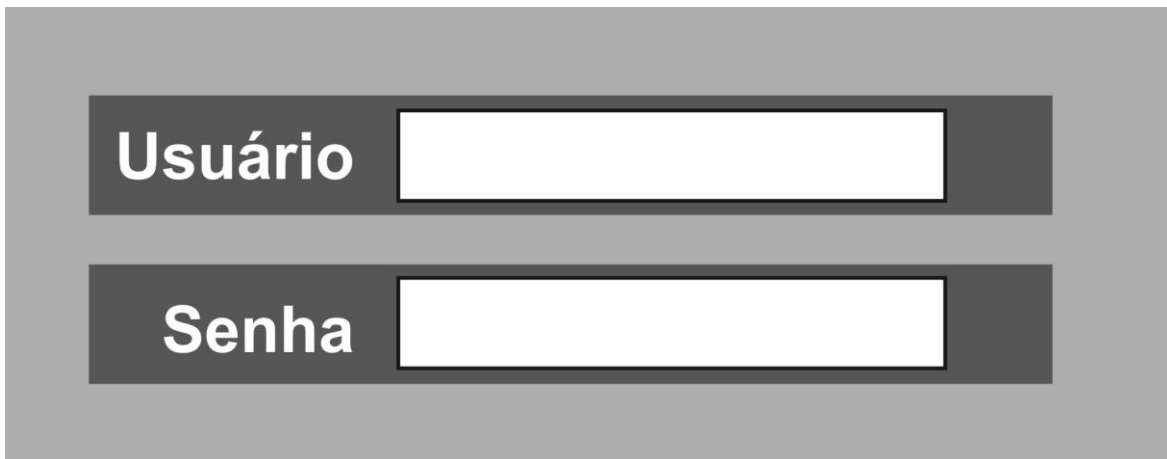
A screenshot of a login form. It consists of two rows. The first row has a dark grey rectangular label on the left containing the word 'Usuário' in white, followed by a white rectangular input field. The second row has a similar dark grey rectangular label on the left containing the word 'Senha' in white, followed by another white rectangular input field. The entire form is set against a light grey background.

Figura 2 - Formulário de Login (In: próprio autor).

Abaixo mostra um exemplo clássico de um código PHP e uma consulta SQL responsáveis por efetuar uma busca de usuários no bando de dados.

```
/**
 *Retorna um usuário
 *@return Puser
 */
public function GetUser ($st_user , $st_pass)
{
    $st_query = "SELECT
                usu_in_id,
                usu_st_nome,
                usu_st_login,
                usu_st_email
    FROM control.tbl_usuario
    WHERE usu_st_login = '$st_user'
    AND usu_st_password = md5('$st_pass')
    AND usu_bo_status = TRUE";
```

Figura 3 – Código PHP(In: próprio autor).

Agora será inserido o código 'OR 1=1 OR' '=' no campo de Usuário do formulário de login e uma senha qualquer.

The image shows a login form with two input fields. The first field is labeled 'Usuário' and contains the text 'OR 1=1 OR' '='. The second field is labeled 'Senha' and contains six asterisks '*****'.

Figura 4 – Injeção de SQL (In: próprio autor).

A consulta SQL no código PHP ficaria assim

```
/**
 *Retorna a lista
 *@return Puser
 */
public function GetUser ($st_user , $st_pass)
{
    $st_query = "SELECT
                usu_in_id,
                usu_st_nome,
                usu_st_login,
                usu_st_email
                FROM control.tbl_usuario
                WHERE usu_st_login = 'OR 1=1 OR' '='
                AND usu_st_password = md5('123456')
                AND usu_bo_status = TRUE";
```

Figura 5 – Resultado da Injeção SQL (In: próprio autor).

A consulta SQL resultante na figura acima, simplesmente retornaria todos os usuários do banco, o resultado dela seria exatamente igual à :

“SELECT usu_in_id, usu_st_nome, usu_st_login, usu_st_email FROM control.tbl_usuario”

E o atacante teria acesso total ao sistema de banco de dados, ou seja poderia apagar a tabela de usuários, com um simples comando no campo usuário:

“DELETE FROM tbl_usuario; SELECT * FROM tbl_usuario WHERE ‘=’”

Abaixo segue o resultado da consulta acima

```

/**
 *Retorna um usuário
 *@return Puser
 */
public function GetUser ($st_user , $st_pass)
{
    $st_query = “SELECT
                usu_in_id,
                usu_st_nome,
                usu_st_login,
                usu_st_email
                FROM control.tbl_usuario
                WHERE usu_st_login = ‘ ‘; DELETE FROM
                tbl_usuario; SELECT * FROM tbl_usuario WHERE ‘ ‘ = ‘ ‘
                AND usu_st_password = md5('123456')
                AND usu_bo_status = TRUE”;
```

Figura 6 – Deletando informações do banco de dados (In: próprio autor).

7.1 Evitando esse tipo de Ataque

Nesse tipo de SQL Injection aplicado nessa pagina PHP, para evitar esse ataque o jeito é usar a função nativa PHP “addslashes(\$str)” para substituir o caractere ‘ por \’ vindos de campos de texto do formulário. Isso deixara o código do formulario de login assim:

```

/**
*Retorna um usuário
*@return Puser
*/
public function GetUser ($st_user , $st_pass)
{
    $st_user = addslashes($st_user);
    $st_pass = addslashes($st_pass);

    $st_query = "SELECT
                usu_in_id,
                usu_st_nome,
                usu_st_login,
                usu_st_email
    FROM control.tbl_usuario
    WHERE usu_st_login = '$st_user'
    AND usu_st_password = md5('$st_pass')
    AND usu_bo_status = TRUE";
}

```

Figura 7- Código PHP corrigido (In: próprio autor).

Caso o atacante fosse inserir o código SQL ' OR 1=1 OR '=' resultaria em:

```

/**
*Retorna um usuário
*@return Puser
*/
public function GetUser ($st_user , $st_pass)
{
    $st_user = addslashes($st_user);
    $st_pass = addslashes($st_pass);

    $st_query = "SELECT
                usu_in_id,
                usu_st_nome,
                usu_st_login,
                usu_st_email
    FROM control.tbl_usuario
    WHERE usu_st_login = ' OR 1=1 OR '='
    AND usu_st_password = md5('123456')
    AND usu_bo_status = TRUE";
}

```

Figura 8 – Injeção de SQL sem êxito (In: próprio autor).

O problema de usar a função *addslashes(\$str)* é que não poderá esquecer de usá-la para nenhum campo de texto do formulário, pois se não o código do banco de dados vai continuar vulnerável a SQL Injection, e também tomar cuidado com os campos numéricos dos formulários, sempre verificar se os dados vindos dos campos numéricos são realmente numéricos.

7.2 RASTREANDO VULNERABILIDADES

Nesse tipo de rastreamento de vulnerabilidades, usaremos um site real, de forma didática para saber se existe algum tipo de vulnerabilidade que poderia ser explorada, para esta ação de busca foi utilizado o *Software “Acunetix Web Vulnerability Scanner 8”*, com a finalidade da realização de um relatório demonstrando se o site está vulnerável à SQL *injection*, ou outra vulnerabilidade, e se o mesmo estiver demonstraremos a forma de evitar essa vulnerabilidade tornando o site mais seguro.

Segundo *Acunetix* (2013),

Acunetix é uma aplicação essencial para qualquer pessoa ou organização que administra uma plataforma que se obtêm de dados sensíveis (senhas, dados bancários, e etc.) Pois fornece um análise completa de portas exaustiva contra o servidor do site, procurando por portas abertas e analisando o serviço de rede a funcionar nestas portas. Também fornece segurança através da layer SQL, encontra vulnerabilidades alem das comuns na Web, podendo mostrar toda rede de endereços do site, descobrir as suas franquezas, permissões, processos do servidor, inspecionar o código HTTP em banners e determinar das palavras-chave guardadas.

Escaneamento em site anônimo	
Scan details	
Scan information	
Starttime	29/08/2013 22:09:02
Scan time	6 hours, 21 minutes
Profile	Default
Server information	
Responsive	True
Server banner	Apache/2.2.14 (Ubuntu)
Server OS	Unix
Server technologies	PHP

Tabela 4 – Informações do web site (In: Acunetix,2013)



Figura 9 – Nível de Segurança do Website (In: Acunetix,2013).

Nesse tipo de escaneamento, são três níveis de segurança, para medir se o site está seguro, os níveis são *High*, *Medium* e *Low*, o site o qual está feito esse estudo, conforme mostra a figura acima o *web* site está no nível *High*, ou seja, uma ou mais alta severidade de vulnerabilidades foram descobertas pelo scanner, um usuário mal-intencionado pode explorar essas vulnerabilidades e comprometer o banco de dados ou reconfigurar o site causando danos imensuráveis, abaixo mostra as quantidades de níveis encontrados:

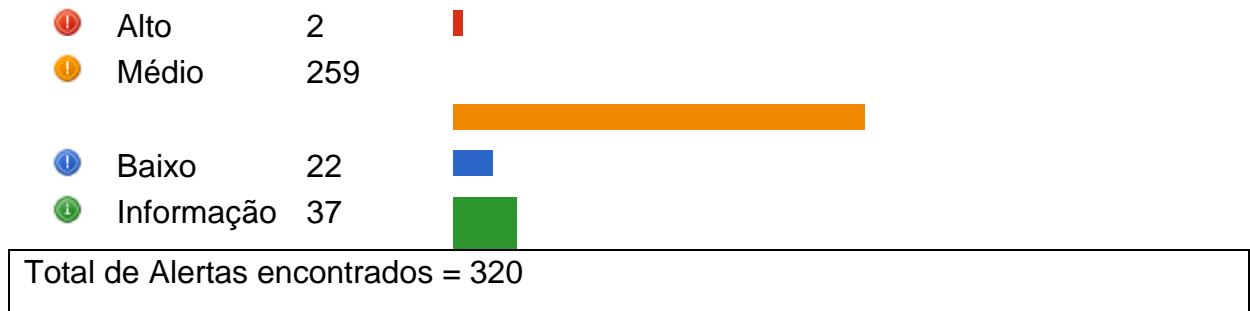


Figura 10 – Alertas encontrados

Gráfico de Alertas

Abaixo será apresentada as vulnerabilidades encontradas, e o modo de inibir os mesmos, iniciando pelo nível mais alto ao mais baixo:

Nível - Alto

Configuração do Código-fonte PHP

Um backup ou arquivo de configuração temporária foi encontrado neste diretório, foi confirmado que este arquivo contém código de fonte PHP, se acontece falhas de edição de texto ou a conexão SSH cai durante a edição, então os arquivos de backup temporários não podem ser limpos corretamente. Além disso, o desenvolvedor do site cria este tipo de arquivos de backup, de seu trabalho ou por administradores ao fazer backups do servidor web.

Itens afetados:

/configuration.php-dist

Código de Fonte Explorado:

```
<?php
/**
 * @version      $Id: configuration.php-dist 14401 2010-01-26 14:10:00Z louis $
 * @package      Joomla
 * @copyright    Copyright (C) 2005 - 2010 Open Source Matters. All rights reserved.
 * @license      GNU/GPL, see LICENSE.php
 * Joomla! is free software and parts of it may contain or be derived from the
 * GNU General Public License or other free or open source software licenses.
 * See COPYRIGHT.php for copyright notices and details.
 *

```



```

* -----
* THIS SHOULD ONLY BE USED AS A LAST RESORT WHEN THE WEB
INSTALLER FAILS
*
* If you are installing Joomla! manually i.e. not using the web browser installer
* then rename this file to configuration.php e.g.
*
* UNIX -> mv configuration.php-dist configuration.php
* Windows -> rename configuration.php-dist configuration.php
*
* Now edit this file and configure the parameters for your site and
* database.
*/
class JConfig {
    /**
     * -----
     * Site configuration section
     * -----
     */
    /* Site Settings */
    var $offline = '0';
    var $offline_message = 'This site is down for maintenance.<br /> Please check
back again soon.';
    var $sitename = 'Joomla!';           // Name of Joomla site
    var $editor = 'tinymce';
    var $list_limit = '20';
    var $legacy = '0';
    /**
     * -----
     * Database configuration section
     * -----
     */
    /* Database Settings */
    var $dbtype = 'mysql';               // Normally mysql

```

```

        var $host = 'localhost';                // This is normally set to
localhost
        var $user = "";                        //          MySQL
username
        var $password = "";                   // MySQL password
        var $db = "";                          // MySQL database
name
        var $dbprefix = 'jos_';               // Do not change unless
you need to!
        /* Server Settings */
        var $secret = 'FBVtggIk5IAzEU9H';     //Change this to something
more secure
        var $gzip = '0';
        var $error_reporting = '-1';
        var $helpurl = 'http://help.joomla.org';
        var $xmlrpc_server = '1';
        var $ftp_host = "";
        var $ftp_port = "";
        var $ftp_user = "";
        var $ftp_pass = "";
        var $ftp_root = "";
        var $ftp_enable = "";
        var $tmp_path      = '/tmp';
        var $log_path      = '/var/logs';
        var $offset = '0';
        var $live_site = "";                   // Optional, Full url to Joomla
install.
        var $force_ssl = 0;                   //Force areas of the site to be SSL ONLY. 0 =
None, 1 = Administrator, 2 = Both Site and Administrator
        /* Session settings */
        var $lifetime = '15';                 // Session time
        var $session_handler = 'database';
        /* Mail Settings */
        var $mailer = 'mail';

```

```

var $mailfrom = "";
var $fromname = "";
var $sendmail = '/usr/sbin/sendmail';
var $smtpauth = '0';
var $smtpuser = "";
var $smtppass = "";
var $smtpghost = 'localhost';
/* Cache Settings */
var $caching = '0';
var $cachetime = '15';
var $cache_handler = 'file';
/* Debug Settings */
var $debug      = '0';
var $debug_db   = '0';
var $debug_lang = '0';
/* Meta Settings */
var $MetaDesc = 'Joomla! - the dynamic portal engine and content
management system';
var $MetaKeys = 'joomla, Joomla!';
var $MetaTitle = '1';
var $MetaAuthor = '1';
/* SEO Settings */
var $sef = '0';
var $sef_rewrite = '0';
var $sef_suffix = "";
/* Feed Settings */
var $feed_limit = 10;
var $feed_email = 'author';
}
?>

```

O impacto desta vulnerabilidade, é que o código de fonte explorado irá divulgar informações confidenciais que vai contribuir a um usuário mal intencionado a preparar ataques avançados.

Para corrigir esta vulnerabilidade, basta remover este arquivo do servidor web e programar uma política de segurança dentro da organização para proibir criação de arquivos temporários nos diretórios acessíveis a partir da web.

Nível - Médio

Apache HTTPD Negação de Serviço

A vulnerabilidade de negação de serviço foi encontrado na forma como os vários intervalos de sobreposição são manipulados pelo servidor HTTPD Apache, existe uma ferramenta de ataque para explorar essa vulnerabilidade “killapache.pl” , com essa ferramenta citada pode ser feito remotamente e com um numero modesto de pedidos pode causar memória muito significativo e uso da CPU no servidor.

A versão atual do Apache do Servidor é a 2.2.14, e a vulnerabilidade descoberta foi usando o Script de Sql Injection “(Version_Check.script)”.

O impacto desta vulnerabilidade é a divulgação de informações, e para corrigir essa vulnerabilidade é a negação de serviço remoto.

Nível - Médio

Mensagens de Erro de Aplicativo

A pagina contém erro e mensagens de aviso que pode divulgar informações sensíveis.

Itens Afetados

/administrator/index.php

/index.php/

/index.php/ap.html

/index.php/apresentacao-pibic.html

/index.php/bolsadeestudo.html

/index.php/calendario-aula-pos-menu/518-pibic-edital.html

/index.php/calendario-aula-pos-menu/862-quimambiental.html

O impacto desta vulnerabilidade é a divulgação de informações confidenciais, e esta informação pode ser utilizada para lançar novos ataques, para corrigir esse erro é necessário revisar o código fonte para este script.

Nível - Médio

Listagem de Diretório

O servidor web está configurado para exibir a lista de arquivos contidos no diretório, isso não é recomendado, pois o diretório pode conter arquivos que normalmente não são expostos por meio de links da web site.

Itens Afetados

/administrator/templates/khepri/images/h_cherry

/administrator/templates/khepri/images/h_green

/administrator/templates/khepri/images/h_teal

/images/bannersCentro

/images/biblioteca/imagens

O impacto desta vulnerabilidade é que um usuário pode ver a lista de todos os arquivos desse diretório, possivelmente expondo informações confidenciais, para corrigir essa vulnerabilidade deve-se certificar-se de que os diretórios não contem arquivos sensíveis ou restringir listagens de diretórios de configuração do servidor web.

Nível – Médio

Formulário HTML sem proteção CSRF

Foi encontrado os formulários sem proteção, com isso poderia ser explorada usando “*Cross-site request forgery*”, que é um tipo de *exploit* malicioso de um site em que comandos não autorizados são transmitidos a partir de um usuário que o site confia.

Itens Afetados:

/vestibular/administrator

/vestibular/index.php/contato.html

/vestibular/index.php/contato.html~

/vestibular/index.php/perguntas.html

/administrator

/administrator/index2.php

O impacto desta vulnerabilidade, é que um atacante pode forçar os usuários de uma aplicação web para executar ações de escolha do invasor. Um exploit CSRF “*Cross-site request forgery*”, bem sucedida pode comprometer os dados do usuário final e operação no caso de usuário normal. Se o usuário final alvo e a conta de administrador, isso pode comprometer a aplicação web inteira.

Para corrigir essa vulnerabilidade, basta verificar se o formulário requer proteção CSRF e implementar contramedidas CSRF, se necessário.

Nível – Médio

Credenciais Enviadas

As credenciais estão sendo enviadas em texto simples, porém todas credenciais de usuários devem ser transmitidos através de um canal criptografado (HTTPS) para evitar ser interceptado por usuários mal intencionados.

Itens Afetados

/administrator

/administrator/index2.php

/phpmyadmin

/phpmyadmin/index.php

/phpmyadmin/index.php (403313c4ca2624f295387f1d654ae7e7)

/phpmyadmin/index.php (6923e40e76221e1ff445c088b114e633)

/phpmyadmin/index.php (f7b7b2447a40fedeb00d2dcbcafc74af)

/vestibular/administrator

O impacto desta vulnerabilidade, é que uma terceira pessoa pode ser capaz de ler as credenciais do usuário, interceptando uma conexão não criptografada HTTP.

Para corrigir essa vulnerabilidade encontrada, as informações confidenciais devera sempre ser transferida para o servidor através de uma conexão criptografa (HTTPS).

Nível – Baixo

Pagina de Login

Esta pagina de login não tem nenhum tipo de proteção contra um ataque muito comum de adivinhação de senhas, conhecido como um ataque de força bruta, um ataque de força bruta é uma tentativa de descobrir uma senha, ao tentar sistematicamente todas as combinações possíveis de letras, números e símbolos até descobrir a combinação correta de um login.

Itens Afetados:

/administrator/index.php

Para corrigir essa vulnerabilidade é necessário implementar algum tipo de bloqueio de conta após um determinado numero de tentativas de senha incorreta.

Acima não foram citados todas as vulnerabilidades encontradas no site, foi citado as mais importantes para ajudar a melhorar a segurança, caso um usuário mal intencionado tivesse a posse dessas vulnerabilidades conseguiria se aproveitar dessas falhas e comprometer o site.

8. MATERIAIS E MÉTODOS

Este trabalho foi desenvolvido a partir do método de pesquisa bibliográfica por meio do método indutivo, usando levantamento de dado feito em testes práticos sendo que um foi realizado em um site anônimo. O teste teve como objetivo detectar vulnerabilidades a sql *injection*, por fim foram proposta soluções de proteção para o mesmo.

9. RESULTADOS

Ao realizar a pesquisa, foi citados meios de vulnerabilidades para uma maior compreensão dos mesmos, e com isso demonstrando um ataque de Injeção de SQL e a forma de inibir esse ataque.

E também de forma didática um análise completa em um site anônimo, que foram achados vários tipos de vulnerabilidades inclusive de Injeção de SQL, tornando o site vulnerável a usuários maliciosos.

As vulnerabilidades encontradas foram expostas a ação para corrigir essa falha, tornando o site seguro.

10. CONSIDERAÇÕES FINAIS

A segurança da informação é necessária que seja realizada de forma correta, pois atualmente o acesso à web é indispensável, com isso os *webs sites* podem estar hostil a vulnerabilidades.

Este trabalho apresentou uma análise das vulnerabilidades existentes, com o proposito de implantação de projeto de segurança, a fim de tornar a web mais segura, pois com avanço tecnológico o risco de aumentar ataques são maiores.

O estudo desenvolvido proporcionou a experiência de ver como os programadores tem que sempre se atualizar e com isso atualizando os códigos webs e banco de dados, para sempre deixar o site o mais seguro possível.

REFERÊNCIAS

ACUNETIX, **Web Application Security**. Disponível em <<http://www.acunetix.com/vulnerability-scanner/>>. Acesso em: 03 set. 2013.

FARIAS, Marcelo Bukowski. **Injeção de SQL em aplicações Web Causas e prevenção**. 2009. 38p. Monografia – Universidade Federal do Rio Grande do Sul Instituto de Informatica, Porto Alegre, 2009.

FERREIRA, K. R. et al. R. (Eds.). **Bancos de dados geográficos**. Curitiba: MundoGeo, 2005. Cap. 5, p. 169–201.

FERREIRA, Marcos. **Conhecendo as vulnerabilidades web**. TrustSign. Disponível em <<http://www.trustsign.com.br/blog/conhecendo-as-vulnerabilidades-web/>>. Acesso em: 19 ago. 2013.

RACCIATTI, Hernán Marcelo. **Técnicas de SQL Injection: Un Repaso**, 2002. Disponível em <<http://www.redeszone.net/wp-content/uploads/Tecnicas-de-SQL-Injection.pdf>>. Acesso em: 08 jul. 2013.

SOUZA, Ranieri Marinho. **Implantação De Ferramentas E Técnicas De Segurança Da Informação Em Conformidade Com As Normas Iso 27001 E Iso 17799**. 2007. 132p. Dissertação (Mestrado) - Pontifícia Universidade Católica de Campinas, Campinas , 2007.

TEIXEIRA, Emanuel Pedro Loureiro. **Ferramenta de Análise de Código para Detecção de Vulnerabilidades**. 2007. 180p. Universidade de Lisboa – Faculdade de Ciências Departamento de Informática, 2007.

WANGHAM, Michelle S.; MELLO, Emerson Ribeiro; FRAGA, Joni da Silva; CAMARGO, Edson. **Segurança em Serviços Web**. In: VI SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, SBSEG'06, 2006, Santos, SP – Brasil.

WEBCHEATS, Community. **TOP 10 Ferramentas SQL Injection**, 2013. Disponível em <<http://www.webcheats.com.br/forum/geek-zone-geral/2125528-Overcom3-lista-alguns-sites-vulneraveis-sql-injection-e-programas-sql-injection.html>>. Acesso em: 10 ago. 2013.