**2. Fundamentação Teórica**

**2.1 SEGURANÇA DA INFORMAÇÃO**

O termo Segurança da Informação, normalmente associado a questões como Vulnerabilidade de Sistemas, se refere a todos os possíveis mecanismos de controle e proteção disponibilizados por uma instituição para defender sua Informação.

Informação é um ativo importante para a organização e os meios de defendê-la são bem variados podendo ser mecanismos lógicos, físicos ou até políticos.

**2.2 NORMA ISSO 27002**

É a norma definida pela ABNT para regulamentar a Segurança da Informação e quais seriam os procedimentos a serem adotados para garantir a segurança dos ativos.

A norma cita desde segurança da informação até controles, políticas, objetivos e até descrevendo ambientes para garantir que a informação esteja segura.

**2.3 APLICAÇÕES WEB**

São sistemas desenvolvidos para responder requisições HTTP usando a arquitetura cliente-servidor. Esses sistemas são hospedados por servidores web que recebem e transferem todas as requisições recebidas para a aplicação.

Essas aplicações podem ser divididas em dois tipos: Estáticas, onde apenas são disponibilizados conteúdos estáticos em formato Html ou dinâmicas (interativas) onde para cada usuário que acesse, a aplicação comporta-se de forma diferente.

Exemplos de aplicações não faltam: E-commerces, sites de bancos, do governo, redes sociais etc.

**2.4 TIPOS DE ATAQUES MAIS COMUNS EM APLICAÇÕES WEB**

**2.4.1 Força Bruta**

É o tipo de ataque baseado na tentativa e erro onde as credenciais de acesso a dados protegidos, software ou sistema são encontradas baseadas em palpites Caso a tentativa consiga ser efetiva, dá ao atacante total acesso a informações, permissões e credenciais da vítima dentro do sistema alvo.

Como se trata de um longo processo de tentativas é comum que sejam utilizadas, no processo, ferramentas automatizadas aliadas ao uso de algumas técnicas que reduzam o espaço de amostras. Dessa forma o processo para descobrimento de credenciais torna-se mais rápido.

A técnica mais comum utiliza dicionários que são amostras escolhidas pelo atacante baseado no prévio conhecimento dos padrões de senha ou de gostos pessoais da vítima. [citação cartilha]

**2.4.2 SQL Injection**

É um ataque que explora a falta de tratamento dos parâmetros textuais enviados a uma aplicação. A injeção de SQL pode ocorrer através da passagem de instruções SQL em parâmetros textuais utilizados por um sistema, a fim de executar ações não autorizadas no banco de dados.

O ataque só tem êxito devido ao interpretador SQL ~~que~~ presente em todos os bancos e utilizado ~~possuem~~ para converter o texto advindo das aplicações em instruções SQL.

A conversão realizada pelo interpretador acontece de forma instantânea e não envolve processo de compilação. Portanto, o trabalho do interpretador se limita a executar o que recebe causando uma possível brecha, caso os dados recebidos não sejam devidamente tratados.

Tendo como exemplo a consulta SQL abaixo:

select \* from usuarios where id = 1;

Se o parâmetro *“Id”* receber um dado malicioso que possa comprometer o funcionamento da instruçãoeesse dado não for devidamente tratado pela aplicação, o texto enviado pode ser interpretado com sucesso pelo interpretador de SQL do banco de dados e consultas maliciosas como essa conseguem ser executadas:

select \* from usuarios where id = 1 **and 1=1; --**

Nessa última consulta, o banco de dados é forçado a retornar como resultado não apenas os dados de um usuário com *id = 1*, mas os dados de todos os usuários cadastrados, uma vez que a igualdade *1=1* sempre será uma condição verdadeira. O comentário de linha ‘--’ força que todas as instruções que vierem após e que estejam na mesma linha sejam ignoradas.

Em caso de sucesso, o SQL Injection dá ao atacante acesso direto ao banco podendo conseguir executar códigos maliciosos para: descobrir nomes e colunas das tabelas, descobrir dados cadastrados na tabela, apagar tabelas ou até mesmo o banco, etc.

**2.4.3 XSS**

Representa o acrônimo de Cross-Site Scripting. É a execução de scripts maliciosos dentro de um site confiável. As formas mais comuns de apresentação do ataque são: A Persistida e Refletida.

Persistida é quando um código Javascript é inserido de forma permanente no banco de dados da aplicação devido à falta de validação da entrada de texto que contém um código Javascript. No momento que o usuário acessa a página atacada, o código é trazido do banco e interpretado pelo navegador acreditando ser um código da aplicação. Um bom exemplo disso é o trecho abaixo:

*<script> alert(“Hello XSS”);</script>*

Caso o código consiga ser inserido no banco via aplicação, ~~cada pessoa que~~ quando ~~acessar~~ acessada, a página atacada ~~verá~~ exibirá um pop-up ~~abrir~~ contendo o texto *“Hello XSS”* ~~sendo exibido~~, dessa forma, o usuário do sistema não saberá ~~sem saber~~ que foi realmente alvo de um ataque.

Já a forma Refletida, faz uso de uma chamada realizada ao servidor para incluir um código malicioso que também é executado no lado cliente. Tomando como exemplo uma aplicação que receba um parâmetro sem tratamento pela URL e exiba-o na tela como indicado abaixo:

*www.meusitevulneravel.com.br?parametroInjetavel=teste*

E sendo exibido na tela assim:

*echo ‘<h1>‘ + getParameter(‘parametroInjetavel’) + ‘</h1>';*

Caso o parâmetro *parametroInjetavel* não receba o tratamento adequado, um código malicioso pode ser injetado por um atacante, posteriormente ser interpretado e executado pelo browser no lado do cliente. Como abaixo:

*www.meusitevulneravel.com.br?parametroInjetavel=<script> alert(“Hello XSS”);</script>*

O código acima exibiria o mesmo pop-up do exemplo de XSS refletido citado anteriormente. Devido à falta de registros armazenados o XSS Refletido é bem mais difícil de ser executado, porém é bem trabalhoso para identificar o ataque. ***[duvida é trabalhoso identificar uma brecha para ataque ou o ataque depois de feito é difícil de ser identificado?]***

No Brasil, já houve registro de ataque via XSS Refletido. O site oficial do Santos Futebol Clube, sofreu uma invasão e foram inseridas na página informações de uma falsa notícia sobre a venda de um jogador famoso do Santos, fazendo com que vários leitores fossem enganados.[citação não salvo] . Esse ataque ocorreu devido à falta de tratamento dos parâmetros enviados à tela.

Com o XSS o atacante pode :[citação Josh Pauli]

1. Usar pop-ups (utilizado em provas de conceito);
2. Sequestrar Identificadores de Sessão;
3. Fazer download e instalar programas danosos como keyloggers;
4. Redirecionar a página para uma URL diferente.

**2.4.4 Path Transversal**

É um ataque executado baseado em tentativa e erro onde o atacante tenta atingir arquivos que estão dentro ou fora do diretório onde o sistema está hospedado. Expressões como “../../” são utilizadas para conseguir navegar dentro da hierarquia de pastas do sistema.

Em caso de sucesso o atacante consegue acessar informações não autorizadas e sensíveis como: o código compilado da aplicação, arquivos do Sistema Operacional da máquina que hospeda a aplicação, Arquivos de Senhas, etc.

Um exemplo de código para esse tipo de ataque seria a execução da seguinte chamada HTTP:

*GET /sistemaVulneravel.php HTTP/1.0*

*Cookie: TEMPLATE=../../../../../../../../../etc/passwd*

Caso executado com sucesso, o código acima conseguiria realizar o download do arquivo de senhas de uma máquina que utiliza Linux.

**2.5 Pentest**

Também conhecido como Teste de Intrusão, é o nome dado a auditoria de segurança feita por um profissional de TI, denominado Pentester. O Pentest busca simular a ação de indivíduos mal intencionados ~~para avaliar como está a questão de segurança da empresa/sistema.~~ para identificar se há vulnerabilidades que possam comprometer a segurança da informação de um sistema ou empresa.

O alvo de um Pentest pode ser um software ou até mesmo a política de uma empresa. No primeiro caso, são utilizadas ferramentas para tentar burlar o esquema de segurança do software, coletar resultados e reproduzir ataques. No segundo caso, podem ser avaliadas: a política de segurança da empresa e o comportamento de seus funcionários frente a algumas situações como: recebimento de e-mails suspeitos e manuseio de informações sigilosas.

**2.6 Kanban**

É uma metodologia de desenvolvimento ágil pouco prescritiva [citação kanban] e por isso, consegue ser facilmente integrada a outras metodologias para melhorar o processo de desenvolvimento de software de equipes ágeis.

Basicamente, o Kanban tem como fundamento os pontos abaixo:

1. Fluxo de Trabalho (Workflow):

A maior preocupação do Kanban é com a transparência do processo de desenvolvimento, a todos os colaboradores. O motivo disso é que caso seja identificado um gargalo, existe a possibilidade dos colaboradores encontrarem processos que o resolvam rapidamente;

1. Limite de quantidade de trabalho paralelo (WIP):

Existe uma variável chamada WIP (Work In Progress, Trabalho em andamento) que representa a vazão de trabalho que a equipe consegue suportar. A principal motivação dessa ideia é deixar transparente a quantidade de tarefas que a equipe consegue atender em determinado período do processo;

1. Gerenciamento e Medidas sobre o fluxo:

Como todas as metodologias de desenvolvimento, é preciso estabelecer métricas e avaliar números para saber como está o andamento da equipe. Para fazer melhorias, primeiro deve-se conhecer onde a equipe acerta e onde ela erra;

1. Oportunidade de Melhorias:

Como foi dito previamente, através das medidas é que se encontram os pontos de melhoria. O Kanban foca no processo contínuo de aprimoramento do processo.

**2.7 Scrum**

É uma metodologia de desenvolvimento ágil bem conhecida no âmbito de gestão e planejamento de projetos de software. Para o Scrum, os itens de funcionalidades que serão desenvolvidos em um projeto são chamados de Product Backlog e o desenvolvimento é dividido em ciclos de tempo denominado Sprints. [citação Scrum]

As Sprints são normalmente divididas em semanas e no inicio ocorre uma reunião chamada Planning, para planejar quais atividades serão inseridas no pipeline da Sprint. Passada essa etapa, o acompanhamento é feito em reuniões diárias rápidas (cerca de 5 minutos) chamadas Daily. Nela todos os membros dão feedback do que foi feito, do que será feito e impedimentos encontrados durante a Sprint. O motivo disso é manter a comunicação da equipe e resolver problemas que atrapalhem o progresso da Sprint.

Ao final da Sprint, acontecem duas reuniões chamadas Sprint Review e Sprint Retrospective. A Sprint Review é onde o time apresenta as funcionalidades desenvolvidas no curso da Sprint. Enquanto que a Sprint Retrospective, é a reunião para avaliar como foi a Sprint e apontar pontos problemáticos e possíveis melhorias a serem feitas na próxima Iteração.

**3. METODOLOGIA**

**3.1 Kali Linux**

É um sistema operacional open source baseado no Debian. O Kali é o sucessor do Back-Track e foi desenvolvido para auxiliar o trabalho do pentest, pois já vem pré-configurado com uma série de ferramentas para testes de exploração, dentre as quais se destacam:

1. Nmap;
2. Hydra;
3. Metasploit;
4. Burp Suite;
5. John the Ripper.

O Kali é mantido atualmente pela empresa Offensive Security Ltd, uma empresa americana de Segurança da Informação que gerencia além do Kali outros projetos na área.

Ele pode ser encontrado na página do projeto [citação projeto Kali] e lá diversas ISO’s podem ser adquiridas para a instalação. Durante os testes realizados neste trabalho, foi utilizada a versão 1.1.0.

Para instalar e executar o Kali, a configuração mínima é:

|  |  |
| --- | --- |
| Arquitetura do Processador | i386 ou AMD64 |
| Espaço em Disco | 8 GB disponíveis |
| Memória | 512 MB |

**3.2 FERRAMENTAS UTILIZADAS**

**3.2.1 Hydra**

É uma das ferramentas disponibilizadas no Kali Linux e pode ser utilizada para realizar ataques do tipo força bruta. Ela foi escrita em C e é uma ferramenta bem versátil para testes de força bruta, sendo capaz de realizar tentativas de intrusão usando vários protocolos como, por exemplo:

1. HTTP GET e POST;
2. FTP;
3. SSH;
4. Mysql;
5. Cisco;
6. LDAP.

Como é uma ferramenta open source e colaborativa, existe um projeto hospedado no repositório Github onde qualquer desenvolvedor interessado pode baixar o código fonte, abrir tickets e submeter correções. [citação Projeto Hydra].

Para os testes descritos neste trabalho, foi utilizada a versão 8.1 que é a versão atual do software.

**3.2.2 SqlMap**

É uma ferramenta escrita em Python que automatiza os processos de detecção e intrusão usando SQL Injection. Como a grande maioria das ferramentas disponibilizadas no Kali, é uma ferramenta open source e colaborativa sendo bem famosa entre os pentesters profissionais. Ela consegue interagir com os principais bancos de dados mais conhecidos e fazer inúmeras tentativas de intrusão.

O SqlMap possui uma página própria e um projeto no Github [citação projeto SQLMAP] de onde é possível baixar o código do projeto, abrir tickets e submeter correções e melhorias.

A versão mais atual do SqlMap é a 1.0, mesma versão utilizada durante os testes descritos neste trabalho.

**3.2.3 Nmap**

É o acrônimo de Network Mapper (Mapeador de Redes), outra ferramenta de grande uso da comunidade pentester. Trata-se de um flexível scaneador de portas utilizado na etapa de conseguir informações do sistema alvo. O software foi escrito em C, C++, Python e Lua.

Além de ser de fácil uso, possui grande documentação disponível e muitos tutoriais online [citação Nmap].

Quanto ao seu funcionamento, o Nmap utiliza pacotes IP brutos para tentar conexões com portas de um sistema e a partir disso descobrir se estão abertas ou fechadas, os serviços disponíveis nessas portas e suas versões, e o sistema operacional utilizado na máquina que hospeda a aplicação alvo.

A versão mais atual do Nmap é a 7.12. Neste trabalho foi utilizada a versão 6.49.

**3.2.4 Burp Suite**

É um grande framework de testes de intrusão que contém vários módulos interessantes para o uso de testes intrusivos em sistemas web. Ao contrário das demais ferramentas descritas anteriormente, o Burp é uma ferramenta privada e licenciada onde para utilizar todos os módulos disponíveis no programa é necessário adquirir uma licença com seus criadores. Entretanto há alguns módulos do sistema que podem ser utilizados gratuitamente.

Os módulos disponibilizados pelo Burp são:

1. Burp Target;
2. Burp Intruder;
3. Burp Spider;
4. Burp Decoder;
5. Burp Proxy;
6. Burp Scanner;
7. Burp Repeater;
8. Burp Comparer.

Dentre os módulos citados, apenas o 1, 3, 5 e 4 são de uso livre.

Neste trabalho, foi utilizada a versão gratuita do software, para tal foi possível utilizar somente os módulos:

* Spider - responsável por fazer um mapeamento completo na hierarquia de diretórios do sistema atacado;
* Proxy - capaz de interceptar todas as requisições ao sistema alvo trazendo informações como cabeçalho das requisições HTTP e parâmetros passados entre cliente e servidor.

A versão mais atual do Burp Suite é a 1.7.03. Neste trabalho foi utilizada a versão 1.6.01.

**3.2.5 Metasploit**

Trata-se de um grande framework automatizado para testes de intrusão escritos em Ruby. Embora ele seja propriedade da empresa Rapid7, existe uma versão gratuita disponibilizada no Kali Linux. É uma ferramenta que trabalha com dois conceitos: Exploits e Payloads.

Exploit é o nome dado ao código mal-intencionado e é feito com o propósito de executar uma ação não autorizada a fim de causar prejuízos. Exemplo: Caso um programa possua uma falha que seja conhecida, o exploit é o código que se aproveita dessa falha para conseguir um acesso não autorizado.

Payload é o nome normalmente dado à transmissão de dados, pois significa “Carga Paga”. Para o Metasploit, payload é o código executado depois que um exploit conseguiu sucesso, ou seja, a ação nociva dentro do sistema que será executado. Exemplo: Baseando-se na falha citada no trecho anterior, após a execução do exploit, um possível payload seria a busca das senhas criptografadas dos usuários do sistema alvo ou até a inserção de um novo usuário para fins danosos ao sistema.

O Metasploit possui uma grande base de dados de exploits e payloads. Além disso, é possível que um pentester também possa escrever os próprios exploits que se integrem aos já existentes.[citação projeto Metasploit].

**3.2.6 Iceweasel**

É um navegador web de código aberto para sistemas baseados em Debian idêntico ao Mozilla. Ele é o navegador padrão configurado no Kali Linux.

**3.3 Processo de Testes**

**Referências**:

<https://www.kali.org> , acessada 19/06/16.

<http://www.itnews.com.au/news/backtrack-successor-kali-launched-336420> , acessado 19/06/16.

<http://tools.kali.org/tools-listing> , acessada 19/06/16.

<https://github.com/vanhauser-thc/thc-hydra> , acessada 17/06/16.

<https://github.com/sqlmapproject> , acessada 17/06/16.

<http://insecure.org/> (Nmap) , acessada 17/06/16.

Testes de Invasão, Uma introdução prática ao Hacking , Georgia Weidman

<https://portswigger.net/burp>, acessada 17/06/16.

<https://www.metasploit.com/> , acessada 17/06/16.

<https://github.com/rapid7/metasploit-framework> , acessada 17/06/16.

<http://cartilha.cert.br/ataques/> , acessada 17/06/16.

<https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)>, acessada 19/06/16.

<https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)>, acessada 17/06/16.

<http://www.redesegura.com.br/2012/01/saiba-mais-sobre-o-cross-site-scripting-xss>, acessada 17/06/16.

<http://www.naosalvo.com.br/o-dia-em-que-o-ganso-foi-para-o-corinthians-eu-virei-um-hacker-e-o-santos-decidiu-me-processar/> , acessada 19/06/16.

Introdução ao Web Hacking, Josh Pauli

Norma ISSO 27002.

<http://www.culturaagil.com.br/kanban-do-inicio-ao-fim/> , acessado em 21/06/16.

<http://www.desenvolvimentoagil.com.br/scrum/>, acessado em 21/06/16.