CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Luan Menezes Silva

Segurança da Informação no Desenvolvimento de Aplicações Web

Americana, SP

2015

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Luan Menezes Silva

Segurança da Informação no Desenvolvimento de Aplicações Web

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Esp. Edson Roberto Gaseta

Área de concentração: Segurança da Informação

Americana, S. P.

2015

Silva, Luan Menezes
S581s Segurança da informação no desenvolvimento de aplicações web. / Luan Menezes Silva. – Americana: 2015.
41f.

Monografia (Graduação em Tecnologia em Segurança da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Esp. Edson Roberto Gaseta

1. Segurança em sistemas de informação I. Gaseta, Edson Roberto II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.518.5

Luan Menezes Silva

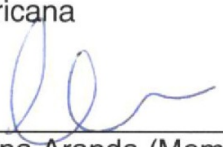
Segurança da Informação no Desenvolvimento de Aplicações Web

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

Americana, 26 de Junho de 2015.

Banca Examinadora:

Edson Roberto Gaseta (Presidente)
Especialista
Fatec Americana



Maria Cristina Aranda (Membro)
Doutora
Fatec Americana

Juliane Borsato Beckdorff Pinto (Membro)
Graduada
Fatec Americana

AGRADECIMENTOS

A instituição FATEC – Americana pelo apoio, pelo trabalho em manter os cursos oferecidos em alto nível.

Aos professores da FATEC – Americana pela orientação, compreensão e conhecimentos dispensados durante o curso.

Ao professor Edson Roberto Gaseta pela orientação e conhecimento a mim dispensado na elaboração deste trabalho.

DEDICATÓRIA

Dedico este trabalho à minha família e à minha namorada, por terem sido a base de tudo o que eu sou, e tudo que eu tenho conquistado em minha vida.

RESUMO

A importância da segurança da informação tem crescido a cada dia, devido à grande quantidade de informações que trafegam pelas redes. As organizações enfrentam dificuldades para garantir que a informação gerada seja entregue ao seu destinatário de forma integra, sem que haja alterações durante o caminho por *hackers* ou pessoas maliciosas. O principal desafio atual dos profissionais de segurança da informação e desenvolvimento de software é produzir uma aplicação segura, seja na criação ou alteração da mesma. Desta forma, este trabalho propõe apresentar regras e normas de segurança da informação que possam ser implantadas em processos de desenvolvimento de softwares, tendo como resultado um software sem vulnerabilidades ou que estejam próximas à zero. Ferramentas que tem por objetivo encontrar falhas presentes em aplicações serão utilizadas para avaliar e discutir como foi realizado o processo de desenvolvimento das aplicações. Com base nos resultados que serão alcançados em torno da segurança da informação em aplicações web, este trabalho tem por objetivo orientar a importância de utilização de padrões de segurança da informação no processo de desenvolvimento de software.

Palavras Chave: Segurança da Informação; Modelos de Desenvolvimento; Desenvolvimento de Aplicações Web.

ABSTRACT

The importance of information security has grown every day, because so much information that travel across networks. Organizations are struggling to ensure that the information generated is delivered to the receiver with integrity, without changes along the way by hackers or malicious people. The main current challenge of security information professionals and software development is to produce a secure application, either in the creation or modification thereto. Thus, this work proposes to present rules and information security standards that can be implemented in software development processes, resulting in software without vulnerability or with quantities close to zero. Tools that aims to find faults present in applications will be used to assess and discuss how was performed the application development process. Based on the results that will be achieved around information security in web applications, this work is intended to guide the importance of using information security standards in the software development process.

Keywords: *Information Security, Development Models, Web Application Development.*

SUMÁRIO

LISTA DE FIGURAS E DE TABELAS

1 INTRODUÇÃO

A questão da segurança da informação tornou-se um tema importante na sociedade contemporânea. De grandes empresas que guardam nos computadores os segredos de seus negócios, até indivíduos que trocam correspondências eletrônicas de caráter pessoal, todos têm o legítimo direito de esperar que os dados confiados às máquinas sejam mantidos intactos e confidenciais, acessíveis apenas às pessoas autorizadas (ARANHA, 2014).

A segurança da informação possui em seu objetivo a confidencialidade, integridade e disponibilidade. Para que isso seja possível, existem normas referentes à segurança da informação que descrevem como devem ser realizados os processos e passos para implementação de segurança em ambientes e softwares.

A falta de segurança em projetos de software é uma das principais preocupações das organizações. É por meio das vulnerabilidades presentes em projetos de software que ocorre a quebra de sigilo e roubo de informações (BRAZ,2009). Uma fonte de problemas de segurança é a não consideração de requisitos de segurança no completo desenvolvimento do sistema (WAGNER, 2011).

A **delimitação do tema** foi definida devido à crescente necessidade de produtos de software que suportam processos de negócios e tem motivado consideravelmente pesquisadores no melhoramento de processos de desenvolvimento de software. Neste sentido, a engenharia da segurança da informação aumentou sua importância, tornando-se parte de processos de negócios a fim de proteger os ativos e as informações das organizações (WAGNER, 2011).

Com o objetivo de obter um software final seguro, existe a necessidade de estudar algumas normas e modelos de segurança da informação que definem as regras necessárias a serem utilizadas. O foco deste trabalho é a segurança da informação no desenvolvimento de aplicações web, devido à tendência de mercado e estudos, serem realizados neste tipo de aplicação.

A principal norma a ser estudada neste trabalho será a ISO/IEC 15408 - *Information technology — Security techniques — Evaluation criteria for IT security*, pois ela é um padrão internacional de desenvolvimento de produtos seguros, que descreve uma lista de regras de segurança que o produto deve conter. Segundo Albuquerque e Ribeiro (2002), a norma ISO/IEC 15408 é o melhor ponto de partida para o desenvolvimento de software seguro, pois ela descreve conceitos necessários para a segurança em sistemas.

O **problema** foi em relação às organizações que desenvolvem aplicações web, quando não conseguem entregar um software seguro, pois podem ocorrem falhas no processo de desenvolvimento, falta de procedimentos ou a não conscientização dos desenvolvedores para seguir os padrões estabelecidos e falta de documentação necessária para a utilização da aplicação de maneira correta pelos usuários. A pressão para entregar o produto final, seja ela interna ou externa também pode influenciar, tendo em vista que a partir da pressão poderá ser necessário ausentar-se dos padrões estabelecidos para um desenvolvimento seguro.

A **pergunta** foi como implementar segurança da informação através de normas no desenvolvimento de software.

As **hipóteses** foram definidas como:

a) Pode ocorrer a não aceitação da empresa desenvolvedora de software, do nível de segurança a ser adotado, por custo e/ou falta treinamento.

b) Pode ocorrer a inviabilidade da implantação de modelos e normas, devido a grande quantidade de recursos técnicos, financeiros e humanos necessários.

c) Pode ser possível implantar modelos e normas de segurança da informação, com a aceitação da empresa e dos funcionários.

d) Devido à pressão para a entrega do produto final, pode não ser possível completar o desenvolvimento de forma segura.

O **objetivo geral** foi estudar quais os requisitos necessários para um software possuir confidencialidade, ser integro e estar disponível sempre que necessário. O

desenvolvimento de software seguro é descrito através de diversas normas ISO/IEC e também alguns modelos, nos quais são relacionados atributos e regras a serem seguidos para obter um software seguro e robusto.

Como **objetivos específicos** foram definidos os seguintes:

a) Estudar, analisar e apresentar as principais regras e norma de segurança da informação; Orientando os profissionais de desenvolvimento de software quanto aos recursos existentes.

b) Contextualizar o desenvolvimento de software e apresentar modelos de desenvolvimento; Objetivando o aumento de segurança da informação em aplicações finais.

c) Avaliar aplicações web com a utilização de ferramentas varredoras de vulnerabilidades; Descobrindo informações valiosas para corrigir falhas nas aplicações.

O **método científico** de pesquisa utilizado foi apresentar a norma e os modelos de segurança da informação relacionadas ao desenvolvimento de *software* seguro, com o objetivo de descrever um ambiente de desenvolvimento de software com segurança.

O trabalho foi **estruturado** em sete capítulos, sendo que o primeiro conceitua a segurança da informação nos dias atuais, o segundo apresenta algumas boas praticas de segurança da informação no desenvolvimento de software, o terceiro discute o desenvolvimento de software e metodologias para produzir um software seguro, o quarto apresenta a norma ISO que orienta quanto ao desenvolvimento de software seguro, o quinto trata a respeito de auditoria de sistemas de informações web e apresenta ferramentas para auxiliar a determinar que uma aplicação seja segura e o sexto demonstra na pratica a utilização de ferramentas para auxiliar na descoberta de vulnerabilidade de uma aplicação web. Com base nas informações conseguidas a partir dos estudos realizados nos capítulos anteriores, o capítulo sete se reserva às considerações finais.

2 BOAS PRÁTICAS DE SEGURANÇA PARA O DESENVOLVIMENTO DE SOFTWARE

A constante evolução dos sistemas de informação e do processo de desenvolvimento de software traz grandes benefícios quanto à velocidade da fabricação de softwares, porém também traz grandes preocupações para garantir que as informações sejam trafegadas de forma integrada. No processo de desenvolvimento existem algumas práticas e regras que se aplicadas, podem auxiliar para que o software final seja mais seguro.

2.1 AMBIENTES PARA O DESENVOLVIMENTO DE SOFTWARE

Segregar ambientes em desenvolvimento de software auxilia principalmente a garantir integridade e disponibilidade das informações para os negócios ou entidades. No processo de desenvolvimento podem existir algumas possibilidades de segregação, no entanto a mais comum é separar um ambiente com base de desenvolvimento, homologação e produção.

Ambiente de desenvolvimento é o ambiente que os desenvolvedores utilizam para construir o software, sem que nenhum usuário tenha necessariamente o acesso a este ambiente.

O ambiente de homologação é o ambiente de teste, após a fabricação do software no ambiente de desenvolvimento, as atualizações são aplicadas no ambiente de homologação para que os usuários validem se as informações do software continuarão corretas após as atualizações.

O ambiente de produção é o ambiente que usuários finais irão utilizar o software, onde as informações da empresa ou entidade são reais, portanto o nível de quantidade de erros neste ambiente deve tender a zero.

2.2 UTILIZAR FERRAMENTAS PARA DESENVOLVIMENTO DE SOFTWARE ATUALIZADAS

Outra importante pratica de segurança, é sempre manter atualizados os softwares relacionados a uma rede ou sistemas de informação, isso porque no desenvolvimento de qualquer software podem ocorrer falhas e com isso criar uma

vulnerabilidade para o SW, dessa maneira, *hackers* ou pessoas maliciosas podem se valer desta abertura e interferir no funcionamento do SW, o que compromete a segurança da informação. Quando vulnerabilidades são descobertas, certos fabricantes costumam lançar atualizações específicas, chamadas de patches, hot fixes ou service packs. Portanto, para manter os programas instalados livres de vulnerabilidades, além de manter as versões mais recentes, é importante que sejam aplicadas todas as atualizações disponíveis (CERT.br,2012).

2.3 DOCUMENTAÇÃO

A importante etapa de documentação em um processo de desenvolvimento de software tem como objetivos principais descrever a estrutura técnica do software e os procedimentos para utilização correta do mesmo. A documentação, no entanto, não é utilizada por algumas empresas e/ou desenvolvedores autônomos no processo de desenvolvimento devido ao custo elevado.

A documentação de um software pode ser dividida em dois grandes grupos, documentação técnica e documentação de uso. A documentação de uso visa descrever manuais e procedimento de trabalho para o usuário final, e não possui detalhamentos técnicos. A documentação técnica possui diversas informações do software, detalhes da arquitetura do sistema, diagramas explicando o funcionamento do SW, explicações de funções contidas dentro dos programas. A documentação gerada em um processo de desenvolvimento de software pode e deve ser utilizada para auxiliar em futuras alterações no SW, isso porque a equipe envolvida na alteração poderá planejar a ação a ser realizada, que diminui a possibilidade de erros.

2.4 BACKUP DOS ARTEFATOS DE DESENVOLVIMENTO

O backup também é um dos principais problemas com a segurança, pois é importantíssimo e poucas pessoas ou empresas mantem o foco necessário neste assunto, quando todas as normas, praticas e padrões de segurança falharem e a informação for comprometida, o único recurso que poderá reestabelecer a integridade da informação será o backup. Os backups devem fazer parte da rotina de operação dos seus sistemas e seguir uma polıtica, devendo ser feito da forma

mais automatizada possível, para reduzir o seu impacto sobre o trabalho dos profissionais responsáveis (NIC.br,2003).

3 DESENVOLVIMENTO DE SOFTWARE

Um processo de desenvolvimento de software é um conjunto de atividades, que tem por objetivo produzir um sistema ou aplicação, por ser altamente complexo, o processo de desenvolvimento deve ser planejado para que a aplicação final tenha qualidade.

3.1 DESENVOLVIMENTO DE SOFTWARE NÃO SEGURO

As falhas no processo de desenvolvimento de um SW normalmente resultam em uma aplicação suscetível a ataques hackers, com isso a aplicação deixa de ser tornar algo confiável e de atender ao propósito de quem a utiliza.

Segundo Holanda e Fernandes (2011) um dos principais fatores causadores dessas vulnerabilidades é a codificação ingênua do software por um programador, quando o mesmo considera basicamente os cenários positivos de execução de um código, sem se preocupar com o caso de usuários maliciosos. Em casos de falhas na codificação, também deve ser considerada a criação de vulnerabilidades no software de modo intencional, uma vez que essas falhas podem não ser encontradas nas fases seguintes do desenvolvimento do software.

Os incidentes decorrentes da exploração dessas vulnerabilidades são geralmente relacionados com a indisponibilidade, a divulgação indevida de informação e a perda de integridade da informação (HOLANDA; FERNANDES, 2011). Para evitar as causas mais comuns de falhas em projetos de desenvolvimento de software é necessário seguir um processo de desenvolvimento que seja baseado nas melhores práticas de mercado.

3.2 DESENVOLVIMENTO DE SOFTWARE SEGURO

A definição de um software seguro se entrelaça com a de segurança, uma vez que esse tipo de aplicação deve conter as regras de segurança, o tripé confiabilidade, integridade e disponibilidade entre outros. Além de atender a essas regras, o software precisa atender principalmente ao motivo pelo qual seu projeto foi concebido, para o controle das informações ao qual o mesmo visa gerenciar.

Segundo Holanda e Fernandes (2011), um software seguro é um software livre de vulnerabilidades, que funciona da maneira pretendida e que essa maneira pretendida não compromete a segurança de outras propriedades requeridas do software, seu ambiente e as informações manipuladas por ele.

No processo de criação de um software seguro, é necessário projeto, desenvolvimento, procedimentos de implementação detalhados, e o entendimento dos mecanismos e técnicas de segurança disponíveis. As falhas de segurança devem ser eliminadas logo no inicio do clico de desenvolvimento de software, com isso as empresas eliminam custos significativos com atualizações e patches de correções das falhas e reduzem o risco para os recursos digitais.

As empresas que desenvolvem softwares e aplicações devem conter procedimentos para que os desenvolvedores sigam corretamente as regras definidas, objetivando um software com o menor número de vulnerabilidades o possível.

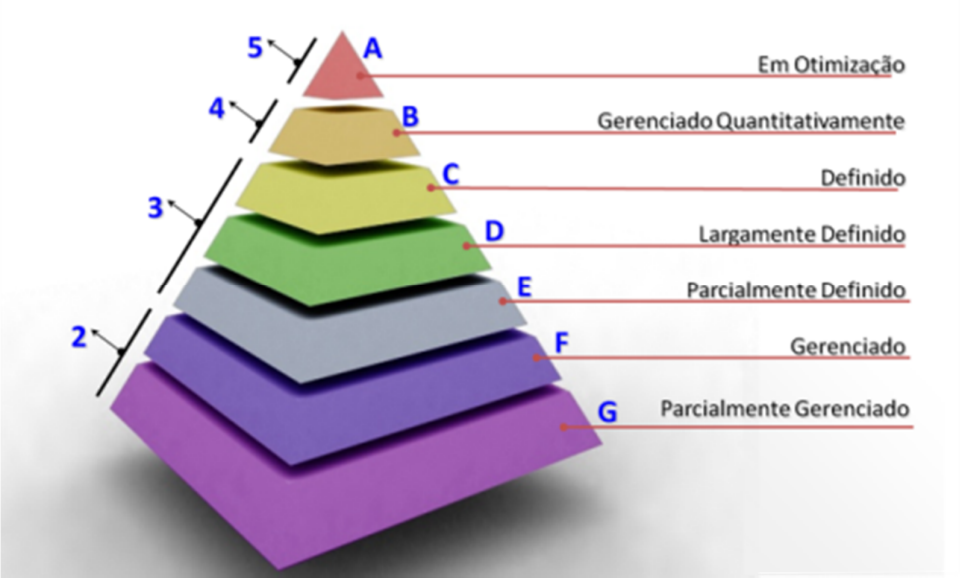
3.3 METODOLOGIA DE DESENVOLVIMENTO MPS.BR

MPS.br ou programa para Melhoria de Processo do Software Brasileiro foi iniciado no ano de 2003 com o intuito de orientar principalmente as micro, pequenas e médias empresas que desenvolvem software quanto ao processo de desenvolvimento utilizado para a criação do mesmo. A criação do modelo foi coordenado pela SOFTEX (Associação de Excelência de Software Brasileiro), com o apoio do MCT (Ministério da Ciência e Tecnologia), da FINEP (Financiadora de Estudos e Projetos) e do BID (Banco Interamericano de Desenvolvimento).

A base técnica para a construção e aprimoramento do modelo de melhoria e avaliação de processo de software é composta pelas normas NBR ISO/IEC 12207 – Processo de Ciclo de Vida de Software, pelas emendas 1 e 2 da norma internacional ISO/IEC 12207 e pela ISO/IEC 15504 – Avaliação de Processo (também conhecida por SPICE: Software *Process Improvement and Capability dEtermination*), portanto, o modelo está em conformidade com essas normas. O modelo também cobre o conteúdo do CMMI-SE/SW, através da inclusão de processos e resultados esperados além dos estabelecidos na Norma ISO/IEC 12207 (MPS.BR,2006).

Segundo MPS.BR(2006), o modelo MPS.br está divido em Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS), sendo que no MR-MPS está contido os requisitos necessários que as empresas devem atender para estar em conformidade com o MPS.br. Nele também são definidos os níveis de maturidade que são uma combinação entre processos e sua capacidade.

Figura 1 – Níveis de Processos do MPS.br



Fonte: Softdesign-rs

Na figura 1 estão descritos os sete níveis de maturidade partindo de G até A, dos quais o MR-MPS pode implementar. Todos os níveis possuem no guia geral do MPS.br as definições de proposito, resultados esperados, informações adicionais para definição e implementação do processo e possíveis bibliografias de apoio. A definição dos sete níveis de maturidade são:

G – Parcialmente Gerenciado: O nível de maturidade G é composto pelos processos Gerência do Projeto e Gerência de Requisitos (MPS.BR,2006).

F – Gerenciado: O nível de maturidade F é composto pelos processos do nível de maturidade anterior (G) acrescidos dos processos Aquisição, Gerência de Configuração, Garantia da Qualidade e Medição (MPS.BR,2006).

E – Parcialmente Definido: O nível de maturidade E é composto pelos processos dos níveis de maturidade anteriores (G e F), acrescidos dos processos

Adaptação do Processo para Gerência do Projeto, Avaliação e Melhoria do Processo Organizacional, Definição do Processo Organizacional, e Treinamento (MPS.BR,2006).

D – Largamente Definido: O nível de maturidade D é composto pelos processos dos níveis de maturidade anteriores (G ao E), acrescidos dos processos Desenvolvimento de Requisitos, Integração do Produto, Solução Técnica, Validação, e Verificação (MPS.BR,2006).

C – Definido: O nível de maturidade C é composto pelos processos dos níveis - de maturidades anteriores (G ao D), acrescidos dos processos Análise de Decisão e Resolução e Gerência de Riscos (MPS.BR,2006).

B – Gerenciado Quantitativamente: Este nível de maturidade é composto pelos processos dos níveis de maturidade anteriores (G ao C), acrescidos dos processos Desempenho do Processo Organizacional e Gerência Quantitativa do Projeto (MPS.BR,2006).

A – Em Otimização: Este nível de maturidade é composto pelos processos dos níveis de maturidade anteriores (G ao B), acrescidos dos processos Implantação de Inovações na Organização e Análise e Resolução de Causas (MPS.BR,2006).

3.4 METODOLOGIA DE DESENVOLVIMENTO ÁGIL SCRUM

Scrum é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software. Segundo Vieira(2014) Scrum é um *framework* que possui um conjunto de valores, princípios e práticas que fornecem a base para que a organização adicione práticas particulares de engenharia e gestão e que sejam relevantes para a realidade da empresa.

Neste método de desenvolvimento existem papeis e responsabilidades fundamentais para a entrega do produto final, são eles:

Product Owner: Cliente ou dono do projeto, responsável pela definição do que deve ser realizado ou *Product Backlog*.

Scrum Master: Responsável pelo andamento do projeto, pelo gerenciamento do Time Scrum, garantindo o cumprimento dos *Sprints Backlog* e utilização correta dos artefatos.

Time Scrum: São os responsáveis pela codificação e teste que transformam os requisitos do *Product Backlog* em partes entregáveis do produto.

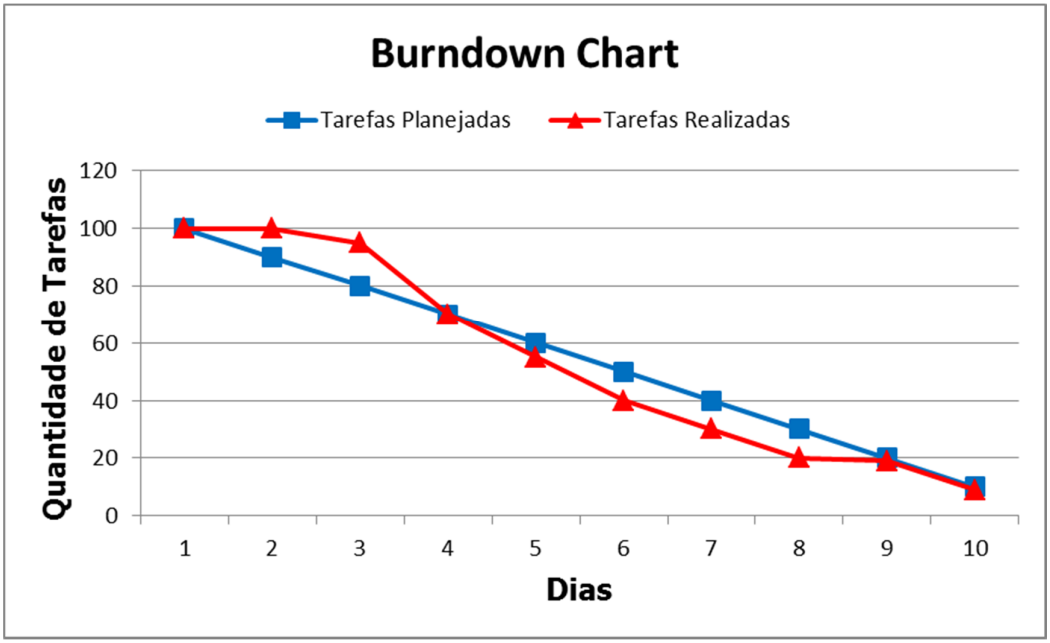
Artefatos são as ferramentas básicas ou mecanismos que permitem o planejamento, execução, controle e adaptação do processo. Os artefatos mais importantes do *framework* são:

Product Backlog: Lista de requisitos e funcionalidades ao qual o produto final deve atender.

Sprint Backlog: São as tarefas que o Time Scrum deve realizar para entregar a funcionalidade ao final do *Sprint.*

Burndowm Chart: Gráfico que mostra a quantidade de trabalho cumulativo restante de um *Sprint*, forma visual para acompanhar o status atual do projeto, conforme a figura 2.

Figura 2 – Burndown Chart



Fonte: Próprio Autor

Após as definições dos papeis e responsabilidades e também dos artefatos, o próximo passo é a realização dos trabalhos através dos *sprints*. O processo inicia-se logo após a definição do *Product Backlog*, o planejamento do *sprint* inicial é realizado através de uma reunião, onde são definidos todos os trabalhos do *Sprint* com duração média de 2 a 4 semanas (VIEIRA,2014).

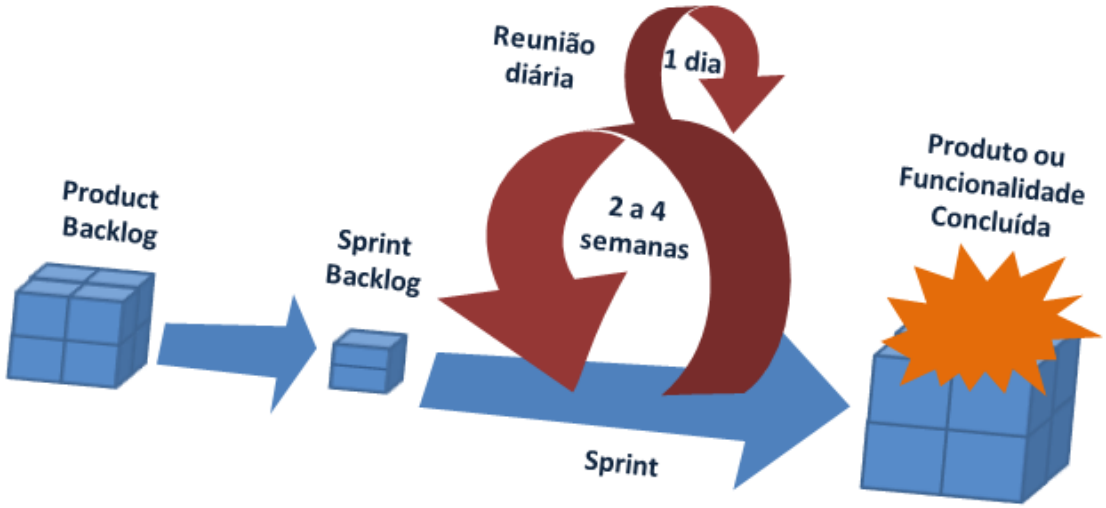
Durante o processo de Sprint devem ocorrer reuniões diárias, também chamadas de *Daily Scrum*, preferencialmente no mesmo horário e com tempo definido de aproximadamente 15 minutos, para discutir basicamente as tarefas realizadas no dia anterior, as tarefas do dia e se há algum problema que impeça concluir o *Sprint*.

Ao final de cada *Sprint* é realizada a Revisão do *sprint*, também conhecido como *Sprint Review*, segundo Vieira (2014) o objetivo desta atividade é verificar e adaptar o produto que está sendo construído.

Após o *Sprint Review* ocorre a ultima reunião chamado Retrospectiva do *Sprint* ou *Sprint Retrospective*, onde o objetivo é verificar as necessidades de mudanças ou adaptações no processo de trabalho (VIEIRA,2014).

Todo o processo pode ser visualizado na figura 3.

Figura 3 – Processos Método Ágil Scrum



Fonte: Vieira (2014)

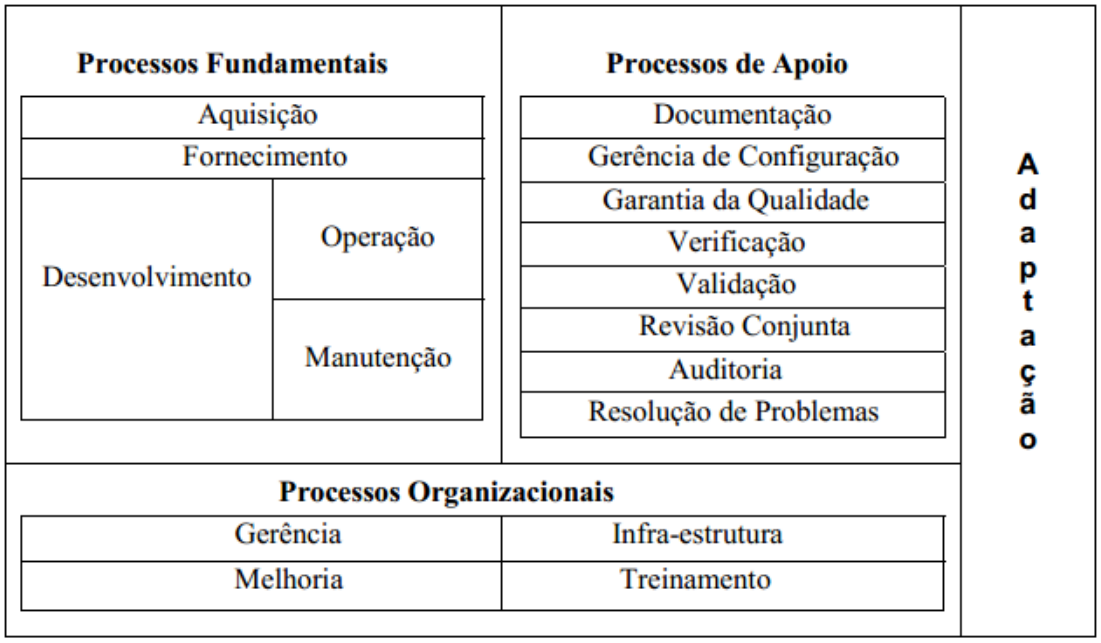
3.5 CICLO DE VIDA DE SOFTWARE

O ciclo de vida de software é uma estrutura que contém processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, que tem como objetivo abranger toda a vida do sistema, desde a definição de seus requisitos até o término de sua utilização. Para regulamentar as regras e procedimentos necessários no ciclo de vida de software foi criada a norma ISO/IEC 12207.

A ISO/IEC 12207 estabelece uma estrutura comum para os processos de ciclo de vida de software, com terminologia bem definida para execução, gerenciamento e melhoria de software. A estrutura contém processos, atividades e tarefas a serem aplicados durante a aquisição, fornecimento, desenvolvimento, manutenção e operação de produtos de software e que devem ser adaptados ao contexto e às características de cada projeto de software. Essa adaptação consiste na exclusão de processos, atividades e tarefas não aplicáveis ao processo. Os processos devem estar em conformidade com as políticas e normas já existentes na organização e devem-se evitar conflitos com os procedimentos já estabelecidos (ABNT,2000).

A ISO/IEC 12207 divide o ciclo de vida de software em quatro processos, sendo eles processos fundamentais contendo cinco atividades, processos de apoio contendo oito atividades, processos organizacionais contendo quatro atividades e o processo de adaptação contendo uma atividade.

Figura 4 – Processos ISO12207



Fonte: ABTN (2000)

Os processos fundamentais de ciclo de vida constituem um conjunto de cinco processos que atendem as partes fundamentais (pessoa ou organização) durante o ciclo de vida de software . Uma parte fundamental é aquela que inicia ou executa o desenvolvimento, operação ou manutenção dos produtos de software (ABNT,2000).

Os processos de apoio de ciclo de vida constituem um conjunto de oito processos. Um processo de apoio auxilia um outro processo como uma parte integrante, com um propósito distinto, e contribui para o sucesso e qualidade do projeto de software . Um processo de apoio é empregado e executado, quando necessário, por outro processo (ABNT,2000).

Os processos organizacionais de ciclo de vida constituem um conjunto de quatro processos. Eles são empregados por uma organização para estabelecer e implementar uma estrutura subjacente, constituída de processos de ciclo de vida e pessoal associados, e melhorar continuamente a estrutura e os processos. Eles são

tipicamente empregados fora do domínio de projetos e contratos específicos; entretanto, ensinamentos destes projetos e contratos contribuem para a melhoria da organização (ABNT,2000).

O processo de adaptação é um processo para realizar a adaptação básica dessa norma para um projeto de software. Os requisitos que se devem seguir para tanto são os seguintes: identificação do ambiente do projeto, solicitação de informações, seleção dos processos, atividades e tarefas, e documentação de decisões e motivos da adaptação.

4 PADRÃO DE SEGURANÇA DA INFORMAÇÃO PARA O DESENVOLVIMENTO DE SOFTWARE

Um padrão de segurança da segurança da informação para o desenvolvimento de software tem por objetivo orientar quanto a regras e processos que devem ser utilizados em processos de desenvolvimento para garantir que a aplicação final seja segura.

4.1 ISO/IEC 15408

A ISO 15408 é um padrão internacional de desenvolvimento de produtos seguros, o qual descreve uma lista de critérios de segurança que um produto deve ter (SPINOLA,2004). A norma visa fornecer um conjunto de requisitos comuns para as funções de segurança em produtos e sistemas e para garantia de medidas aplicadas a eles durante uma avaliação de segurança (ISO, 2005).

Seu objetivo é ser usado como base para avaliação de propriedades de segurança de produtos e sistemas de TI, permitindo a comparação entre os resultados de avaliações independentes de segurança, por meio de um conjunto de requisitos padronizados a ser atingidos (WAGNER,2011).

Segundo Spinola (2004) a ISO 15408 está dividida em três partes: Define o conceito e os princípios gerais da avaliação da segurança da informação em TI e apresenta a lista de requisitos de segurança, que estão divididos em classe, família e requisito; Cataloga uma série de requisitos funcionais e organizados em famílias e classe para a avaliação do *Target Of Evaluation* (TOE – sistema que está sendo avaliado); Define o critério de avaliação para o *Protection Profile* (PP) e *Security Target* (ST) e apresenta sete pacotes de segurança pré-definidos que são chamados de *Evaluation Assurance Level* (EAL).

A terceira parte da ISO/IEC 15408 trata a respeito de avaliação de sistemas ou TOEs, a avaliação é feita através dos sete pacotes de segurança de *Evaluation Assurance Level* ou Nivel de Garantia de Avaliação, são eles:

EAL1: Funcionalmente Testado: As funcionalidades de segurança são testadas no produto pronto e sem acesso maior ao desenvolvedor ou ao código fonte;

EAL2: Estruturalmente Testado: As funcionalidades de segurança são testadas no produto pronto e busca no desenvolvimento do produto vulnerabilidades óbvias;

EAL3: Metodicamente Testado e Verificado: O desenvolvedor deve mostrar mais sobre o processo de desenvolvimento, gerenciamento de configuração, e alguns dos resultados de testes dos desenvolvedores devem ser verificados independentemente;

EAL4: Metodicamente Projetado, Testado e Revisado: Os detalhes da estrutura do programa estarão todos disponíveis, testes são verificados independentemente, desenvolvedores deve mostrar o uso de boas praticas de segurança no desenvolvimento do sistema. O teste é feito em busca de vulnerabilidades óbvias;

EAL5: Semi-formalmente Projetado e Testado: O processo é baseado em rigorosas práticas comerciais e toda a fase de desenvolvimento foi testada, o nível de segurança alcançado é alto;

EAL6: Semi-formalmente Verificado Projeto e Testado: Para este processo são utilizadas técnicas de engenharia de segurança para um ambiente rigoroso do desenvolvimento. A busca por vulnerabilidades deve assegurar a resistência elevada ao ataque de penetração;

EAL7: Formalmente Verificado Projeto e Testado: O produto foi avaliado, tanto na fase de concepção e na fase de desenvolvimento para oferecer altos níveis de segurança.

Cada EAL possui um conjunto de requisitos de segurança ou SARs – *Security Assurance Requirements* (Requisitos de Garantia de Segurança).

5 AUDITORIA DE SISTEMAS WEB

A auditoria em ambiente de tecnologia da informação visa assegurar que as informações guardadas em forma eletrônica sejam confiáveis para o que ela foi concebida. Segundo Imoniana (2008) a filosofia de auditoria em tecnologia de informação está calçada em confiança e em controles internos, que visam confirmar se os controles internos foram implementados e se existem; caso afirmativo, se são efetivos.

5.1 TESTE DE INVASÃO DE APLICAÇÕES WEB

A auditoria de um sistema de informação contém o processo de testes de software, que visa investigar o SW com intuito de encontrar falhas em seu processo de utilização e também fornecer informações de sua qualidade ao contexto em que o mesmo deve operar. Segundo Uto (2013,p37) teste de invasão, intrusão, penetração ou *pentest*, é um método utilizado para verificar a segurança de um ambiente ou sistema, através da simulação de ataques explorando vulnerabilidades encontradas.

O *pentest* é realizado através de um processo cíclico que dependente do conhecimento técnico do auditor, principalmente nas aplicações web, pois as mesmas apresentam diversas características únicas. Existem tipos de testes de penetração, que são classificados de acordo com a quantidade de informação que é passadas ao auditor de segurança.

Tipo de teste caixa branca, nesta técnica todas as informações do alvo são fornecidas ao auditor, incluindo topologia de rede, plataformas utilizadas, diagramas de casos de uso, linguagens empregadas, códigos-fontes e endereçamento interno. O objetivo desta abordagem é simular ataques que podem ser realizados por pessoas com conhecimento privilegiado do alvo, como funcionários e ex-colaboradores (UTO,2013,p38).

Tipo de teste caixa preta, visa simular as mesmas condições de um atacante real, que possui acesso somente às informações publicas do alvo, como endereços IP externos, nomes de domínio e ramo de negocio da entidade (UTO,2013,p37).

No tipo de teste caixa cinza, este tipo de teste é uma mistura dos dois anteriores, antecipando ao auditor aquelas informações do alvo que um atacante

obteria, cedo ou tarde, em um processo de invasão real. Desse modo, o objetivo desta modalidade é acelerar a execução do teste, poupando o tempo gasto pelo auditor nos processos de reconhecimento e mapeamento (UTO,2013,p38).

Teste duplo-cego é um tipo caixa-preta, no qual a equipe de segurança do ambiente alvo não é avisada sobre a execução do trabalho (UTO,2013,p38).

Teste duplo-cinza ocorre quando a equipe de segurança do ambiente alvo sabe o período e o escopo dos testes, mas não os vetores ou canais que serão empregados pelo auditor (UTO,2013,p38).

Teste reverso é um teste em que o auditor recebe todas as informações disponíveis sobre o alvo e no qual a equipe de segurança do ambiente alvo não é notificada sobre a realização dos testes (UTO,2013,p38).

5.2 VULNERABILIDADES

Uma vulnerabilidade é uma falha que pode ser explorada por um invasor, para roubo ou alterações de informações. Exemplos de vulnerabilidades são falhas no projeto, na implementação ou na configuração de programas, serviços ou equipamentos de rede (CERT.br,2012).

A vulnerabilidade é uma falha que permite que um invasor comprometa a integridade da informação de um sistema. Vulnerabilidade é a uma junção de três elementos: uma falha do sistema, acesso do invasor à falha e a capacidade do invasor de explorar a falha.

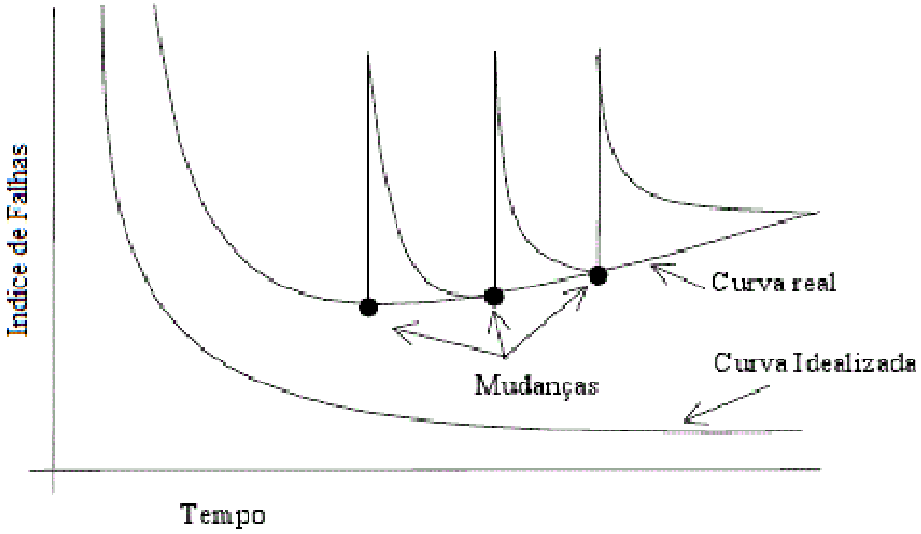
Um ataque de exploração de vulnerabilidades ocorre quando um atacante, utilizando-se de uma vulnerabilidade, tenta executar ações maliciosas, como invadir um sistema, acessar informações confidenciais, disparar ataques contra outros computadores ou tornar um serviço inacessível (CERT.br,2012).

Existem diversos tipos de vulnerabilidades que um invasor pode tentar explorar, entre elas estão *SQL Injection* quando um invasor envia um simples texto que explora a sintaxe do interpretador, frequentemente encontrado em consultas SQL; *Cross-site Scripting* (XSS) ocorre quando um atacante inclui dados maliciosos em um servidor web, e este servidor inadvertidamente inclui esses dados em uma

pagina web para um usuário comum; Utilização de Componentes Vulneráveis Conhecidos, pois esses *frameworks*, bibliotecas e outros módulos de software normalmente possuem um alto grau de privilégios, portanto caso estes possuam falhas, o invasor pode se valer das vulnerabilidades para atacar uma aplicação web OWASP(2013).

Cada defeito de software indica um erro no projeto ou no processo pelo qual o projeto foi traduzido em código de máquina executável (PRESSMAN,2011). O software não se deteriora, portanto um software que não contenha erros irá permanecer sem erros, desde que as condições para seu funcionamento se mantenham do mesmo modo. A tendência em falhas de software se altera quando há alterações no software ou no ambiente ao seu redor, conforme figura 5.

Figura 5 – Curva Ideal x Curva Real



Fonte: Pressman (2011)

5.3 FERRAMENTAS PARA ANALISE DE VULNERABILIDADE EM APLICAÇÕES WEB

As ferramentas para analise de vulnerabilidade também conhecidas varredores de vulnerabilidades, visa encontrar falhas em um software ou aplicação. Segundo Uto (2013,p48) o mecanismo básico de funcionamento consiste no envio de requisições e analise das respostas obtidas, em busca de evidencias de que uma data vulnerabilidade está presente.

Aplicações web são muito dinâmicas e podem possuir uma semântica que dificilmente uma ferramenta de analise de vulnerabilidade conseguira compreender. Em decorrência deste fato e do estado-da-arte desse tipo de tecnologia, apenas algumas classes de vulnerabilidades de aplicações podem ser detectadas de maneira confiável e, para qualquer classe, não se deve esperar que as ocorrências, em uma dada aplicação, sejam descobertas (UTO,2013,p48).

Há inúmeras ferramentas disponíveis no mercado para analise de vulnerabilidade, como exemplos existem as seguintes:

Acunetix é uma ferramenta comercial e é desenvolvida pela empresa de mesmo nome, possui alto desempenho no teste de segurança de servidores e aplicações web. Disponível apenas para a plataforma Windows analisa diversas vulnerabilidades como Injeção de SQL, execução de código, inclusão de arquivos entre outras. Possui recursos diferenciais como a possibilidade de integração de programas desenvolvidos pelo cliente e autoconfiguração de firewall para a aplicação analisada, para que esta aplicação não seja alvo de ataques até que as vulnerabilidades encontradas sejam resolvidas pelo desenvolvedor da aplicação.

Grendel Scan é uma ferramenta livre que é fornecida sob licença GPL. Esta ferramenta pode ser utilizada nos sistemas operacionais Windows, Linux e MacOS. Codificada em java este software possui uma interface fácil de ser utilizar e que permite configurar o escaneamento em servidores web, e com isso descobrir vulnerabilidade como injeção de SQL, fixação de sessão, indexação de diretórios entre outras. O resultado do teste pode ser salvo em HTML.

Nikto é um software livre codificado com a linguagem Perl e que é fornecido sob a licença GPL. Disponível para as plataformas Linux, Windows e MacOS, nikto é um scanner de servidor web que realiza testes abrangentes contra servidores da web para vários itens como checar se existem componentes desatualizados no servidor, adivinhar subdomínios, entre diversos outros. O relatório final da analise pode ser salvo em formato Texto, XML, HTML, NBE ou CSV. Uma informação importante é que nikto é uma ferramenta ruidosa, isto é, procura executar a tarefa no menor tempo possível, sem se preocupar em ser detectada por sistemas de

detecção de intrusão (UTO,2013,p50). Porém, caso seja necessário, há bibliotecas com recursos disponíveis para que ele não seja detectado por ferramentas IDS.

6 ESTUDO DE CASO

O estudo de caso refere-se à execução de uma ferramenta para descoberta de vulnerabilidades em duas aplicações web, com diferentes propósitos, a primeira uma aplicação para gerenciar chamados e recursos de TI de uma entidade ou organização, distribuída sob a licença GNU/GPL foi desenvolvida para funcionar em um servidor web preferencialmente local, sem a exposição à internet.

Figura 6 – Tela de Login GLPI

Fonte: Próprio Autor

A segunda aplicação web tem com o proposito funcionar como website de uma empresa particular, sendo um meio de comunicação do cliente com a empresa. Foi desenvolvida por uma empresa terceirizada e cedida sob licença comercial.

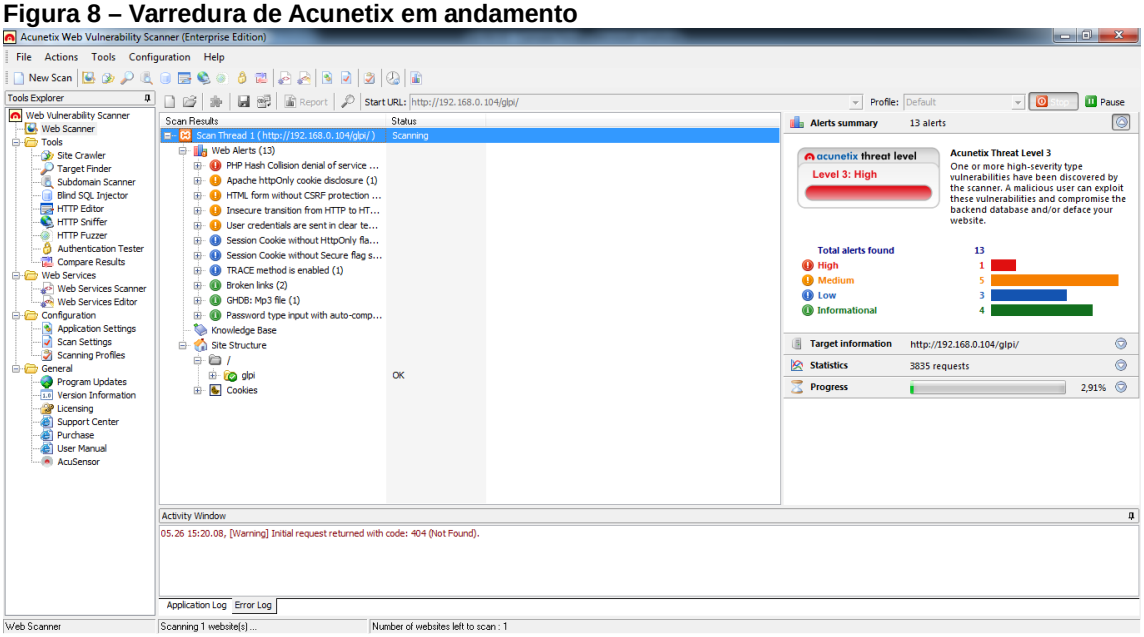
Figura 7 – Pagina Inicial Website

Fonte: Próprio Autor

A ferramenta utilizada para a varredura nas aplicações web, em busca de vulnerabilidade que permitam que invasores prejudiquem a confiabilidade das informações contidas nessas aplicações foi a Acunetix Web Vulnerability Scanner.

A execução da ferramenta nas aplicações é dada de maneira bem simples, bastando inserir a URL da aplicação a ser analisada na ferramenta Acunetix, após isso pressionar *Start* para iniciar o processo de varredura. Com essa ferramenta de analise é possível definir diversos tipos de perfis para vasculhar uma determinada aplicação web em busca de tipos de vulnerabilidades específicos, neste caso foi utilizado o padrão da ferramenta onde são vasculhadas todas as vulnerabilidades possíveis.

Figura 8 – Varredura de Acunetix em andamento

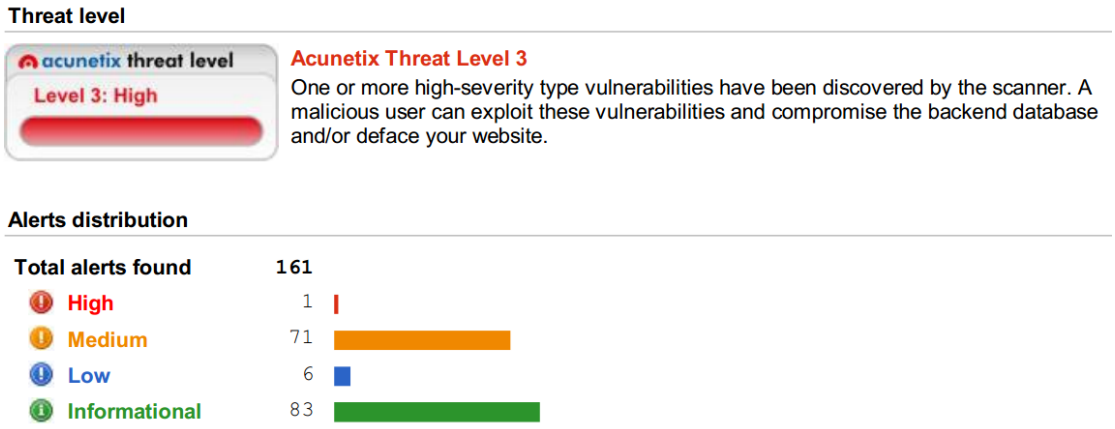


Fonte: Próprio Autor

O tempo gasto para a execução da Acunetix nas duas aplicações teve um diferença considerável, isso porque a quantidade de informações que trafegam nas duas também possui essa diferença. Na aplicação GLPI o tempo gasto foi aproximadamente 41 minutos enquanto na aplicação *website* o tempo gasto foi aproximadamente 4 minutos. No término de execução em cada aplicação a ferramenta apresenta um relatório do processo, com um nível de detalhamento alto para que o responsável possa tomar as atitudes necessárias para sanar as vulnerabilidades.

Nas imagens a seguir são apresentadas informações mais detalhadas do resultado da execução da ferramenta de descoberta de vulnerabilidades nas aplicações web.

Figura 9 – Relatório de Resultado GLPI



Fonte: Próprio Autor

Figura 10 – Relatório de Resultado Website

Fonte: Próprio Autor

Na análise de resultados após a execução da ferramenta de varredura nas aplicações web, fica evidenciada a grande diferença no processo de desenvolvimento de ambas, devido a grande quantidade de vulnerabilidades descobertas na aplicação GLPI, foram encontradas 161 vulnerabilidades, incluindo falhas consideradas de níveis médias e altas. Como em sua concepção não há conceito de ficar exporta a internet, podemos supor que não fui utilizada nenhuma norma ou regra no processo de desenvolvimento para garantir a segurança da informação desta aplicação.

Na aplicação *website* foram encontradas apenas 2 vulnerabilidades, com nível de falha baixo. Como esta aplicação foi desenvolvida em um projeto que contemplava a exposição da mesma na internet, podemos supor que foram

utilizadas normas e/ou regras no processo de desenvolvimento para garantir a segurança da informação pela empresa fabricante da aplicação.

7 CONSIDERAÇÕES FINAIS

O processo de desenvolvimento de software seguro tem se tornado cada vez mais importante para as organizações, isso porque nos dias atuais dificilmente uma empresa consegue ter organização sem um sistema de informação, e garantir que essas informações sejam trafegadas de maneira integra e autentica é um desafio constante dos profissionais de segurança da informação e desenvolvimento de software.

A partir da apresentação deste trabalho e da análise dos dados de resultado, observa-se que é importante utilizar normas e regras relacionadas à segurança da informação no processo de desenvolvimento de software com o objetivo de se obter um software seguro.

Podem ser desenvolvidos outros trabalhos a partir deste, como discutir sobre outras diversas práticas de segurança e normas relacionadas, e também é possível definir e apresentar modelos para a implantação das praticas e normas de segurança da informação em organização.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, R.; RIBEIRO, B. **Segurança no desenvolvimento de software**. Rio de Janeiro: Campus, 2002. 320 p.

ARANHA, A. C. **A sociedade e a segurança da informação**. Disponível em: <http://technet.microsoft.com/pt-br/library/cc668426.aspx> Acesso em: 13 de mai. 2014.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS - ABNT. **NBR 12207:** Tecnologia de Informação. Processos de ciclo de vida de software. Rio de Janeiro: ABNT, 2000. 35p.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS - ABNT. **NBR 27001**: Tecnologia da Informação. Sistema de Gestão da Segurança da Informação. Rio de Janeiro: ABNT, 2006. 40p.

BRAZ, F. **Instrumentação da análise e projeto de software seguro baseada em ameaças e padrões**. 2009. Tese (Doutorado em Engenharia Elétrica) - Faculdade de tecnologia, Brasília, Brasil, 2009.

CERT.br. **Cartilha de segurança para internet**. Núcleo de Informação e Coordenação do Ponto BR, São Paulo, 2012.

HOLANDA, T. M; FERNANDES, C. J. **Segurança no desenvolvimento de aplicações**. Brasília, Brasil. 2011. 43p

IMONIANA, J. O. **Auditoria de sistemas de informação**. Brasil. Editora Atlas, 2.ed, 2008. 208p.

INTERNACIONAL ORGANIZATION FOR STANDARDIZATION – ISO. **ISO 15408-1**. *Information technology*. *Security techniques. Evaluation criteria for IT security*. Genebra: ISO, 2005. 62p.

MPS.BR. ASSOCIAÇÃO PARA PROMOGRAÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – SOFTEX. **MPS.BR – Guia Geral**, versão 1.1, 2006, 56p.

NIC.br. **Práticas de segurança para administradores de redes internet**. NIC BR Security Office. 2003

PRESSMAN, R. S. **Engenharia de Software**: Uma Abordagem Profissional. Porto Alegre: McGraw-Hill, 7.ed, 2011. 780p.

Softdesign-rs. **Próxima parada mps.br**: Nível E. Disponível em <http://www.softdesign-rs.com.br/hotsites/mpse/imagens/piramide.png> Acesso em: 28 mai. 2015.

SPINOLA, R. O. **Conhecendo a ISO 15408**: Trabalhando com segurança da informação. Engenharia de Software Magazine. Devmedia, 45.ed, 2004 31-36p.

VIANA, M. **Senha fraca? Nem a criptografia salva**. TrustSign. Disponível em <https://www.trustsign.com.br/portal/blog/senha-fraca-nem-a-criptografia-salva/> Acesso em: 25 abr. 2015. 2014.

VIEIRA, D. **Scrum**: A Metodologia Ágil Explicada de uma forma Definitiva. MindMaster. Disponível em <http://www.mindmaster.com.br/scrum> Acesso em: 24 abr. 2015. 2014

UTO, N. **Teste de invasão de aplicações web**. Rio de Janeiro: Escola Superior de Redes, 2013. 510p.

WAGNER, R. **Processos de desenvolvimento de software confiáveis baseados em padrões de segurança**. 2011, 107p. Dissertação (Mestrado em Informática) – Centro de Tecnologia, Universidade Federal de Santa Maria, Santa Maria, 2011.