

**CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

**MÉTRICAS DE SOFTWARE PARA A SEGURANÇA DA
INFORMAÇÃO: DEFINIÇÃO E APLICAÇÃO EM SISTEMAS WEB**

PEDRO HENRIQUE CAMARGOS SANTOS

**ANÁPOLIS
2019**

PEDRO HENRIQUE CAMARGOS SANTOS

**MÉTRICAS DE SOFTWARE PARA A SEGURANÇA DA
INFORMAÇÃO: DEFINIÇÃO E APLICAÇÃO EM SISTEMAS WEB**

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientadora: Profa. Ma. Viviane Carla Batista Pociwi.


ANÁPOLIS
2019

PEDRO HENRIQUE CAMARGOS SANTOS

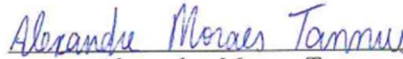
**MÉTRICAS DE SOFTWARE PARA A SEGURANÇA DA
INFORMAÇÃO: DEFINIÇÃO E APLICAÇÃO EM SISTEMAS WEB**

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

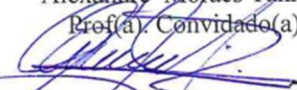
Aprovado pela banca examinadora em 6 de dezembro de 2019, composta por:



Viviane Carla Bastista Pocivi
Presidente da Banca



Alexandre Moraes Tarnus
Prof(a). Convidado(a)



William Pereira dos Santos Júnior
Prof(a). Convidado(a)

Dedico este trabalho à minha mãe,
exemplo de pessoa e que sempre lutou
pela educação dos seus filhos.

AGRADECIMENTOS

Agradeço, primeiramente a Deus, que sempre esteve presente e foi meu guia maior para a realização desse trabalho.

Agradeço à minha mãe, por sempre estar presente e me auxiliar durante esta longa jornada que é a graduação.

À minha família, por sempre acreditar e me apoiar em minhas decisões.

À instituição, por proporcionar uma formação de qualidade com professores renomados.

RESUMO

A análise de métricas é fundamental durante o processo de tomada de decisão, apoiando nas diferentes visões, seja no projeto, produto ou processo de produção de software. Considerando que “não podemos gerenciar aquilo que não podemos medir”, entende-se que a avaliação da segurança da informação nestas diferentes visões é essencial para que medidas preventivas e corretivas possam ser tomadas. Este trabalho tem como objetivo geral, avaliar o nível de segurança da informação de sistemas *web* utilizando métricas de software, e trata-se de um estudo exploratório, do tipo quanti-quali, com análise documental e realização de estudo de caso. Para tanto, se utilizou como ponto de partida um processo de gestão da segurança da informação, proposto por uma pesquisa no Bacharelado de Engenharia da Computação da UniEVANGÉLICA. As etapas dessa pesquisa incluem: (a) estudo do processo selecionado, juntamente a revisão da literatura; (b) definição de métricas usando o método Objetivo-Questão-Métricas (OQM); (c) aplicação das métricas definidas em sistemas *web*; e (d) adição de melhorias no processo inicialmente estudado. Como resultado, têm-se métricas que orientam a avaliação do nível de segurança de uma aplicação *web*, além de nortear possíveis correções aos proprietários dos sistemas. Além disso, a melhoria adicionada ao processo selecionado poderá contribuir com a maximização do nível de segurança desses sistemas, permitindo identificar as possíveis vulnerabilidades e as possíveis ações necessárias para as correções das falhas de segurança, caso existam.

Palavras-chave: Sistemas *web*. Segurança da informação. Métricas de software.

ABSTRACT

The analysis of metrics is fundamental during the decision making process, supporting the different views, whether in the project, product or software production process. Considering that “we cannot manage what we cannot measure”, it is understood that the assessment of information security in these different views is essential for preventive and corrective measures to be taken. This work aims to evaluate the level of information security of web systems using software metrics, and it is an exploratory study, quanti-quali type, with document analysis and case study. To this end, it used as a starting point an information security management process, proposed by a research at the Bachelor of Computer Engineering of UniEVANGÉLICA. The stages of this study include: (a) the study of the selected process, along with literature review; (b) definition of metrics using the Objective-Question-Metrics (OQM) method; (c) application of metrics defined in web systems; and (d) adding improvements to the process initially studied. As a result, there are enough metrics to assess the security level of a web application, and to guide possible fixes to owners of web systems. In addition, it is believed that the improvements to be proposed in the selected process may contribute to the maximization of the security level of these systems, allowing the identification of the necessary measures to correct the security breaches, if any.

Keywords: Web systems. Information security. Software metrics.

LISTA DE ILUSTRAÇÕES

Figura 1- Quantidade de incidentes (2019).....	16
Figura 2 - Fluxo de ataque.	17
Figura 3 - Níveis hierárquicos da metodologia GQM.....	25
Figura 4 - Fluxo do processo de gestão estudado.	27
Figura 5 - Adição de melhoria ao processo X.....	30
Figura 6 - Métricas identificadas.....	31
Figura 7- Vulnerabilidades da AP1.....	34
Figure 8 - X-Frame-Option Header Not Set da AP1	35
Figura 9 - Informações do servidor da AP1	37
Figura 10 - Vulnerabilidades da AP2.....	38
Figure 11 - X-Frame-Option Header Not Set da AP2	39
Figura 12 - Dados sensíveis expostos da AP2.....	39
Figura 13 - Falha de programação da AP2	40

LISTA ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	<i>Application Programming Interface</i>
CERT BR	Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil
FTT	Fábrica de Tecnologias Turing
HTTP	<i>HyperText Transfer Protocol</i>
IEC	<i>International Electrotechnical Commission</i>
ISO	<i>International Organization for Standardization</i>
LDAP	<i>Lightweight Directory Access Protocol</i>
Nmap	<i>Network Mapper</i>
NoSQL	<i>Not Only Structured Query Language</i>
ORM	<i>Object-relational mapping</i>
OWASP	<i>Open Web Application Security Project</i>
SisLAC	Sistema Laboratorial de Análises Clínicas
SQL	<i>Structured Query Language</i>
UniEVANGÉLICA	Centro Universitário de Anápolis
URL	<i>Uniform Resource Locator</i>
XSS	<i>Cross-site Scripting</i>
ZAP	<i>Zed Attack Proxy</i>

LISTA DE TABELAS

Tabela 1 – Comparação dos resultados da medição.	41
---	----

SUMÁRIO

1. INTRODUÇÃO	12
2. FUNDAMENTAÇÃO TEÓRICA.....	15
2.1. SEGURANÇA DA INFORMAÇÃO	15
2.2. PRINCIPAIS VULNERABILIDADES EM APLICAÇÕES <i>WEB</i>	18
2.2.1. Injeção	18
2.2.2. Quebra De Autenticação.....	19
2.2.3. Exposição De Dados Sensíveis	19
2.2.4. Entidades Externas De XML (XXE).....	20
2.2.5. Quebra De Controle De Acesso.....	20
2.2.6. Configurações De Segurança Incorretas	20
2.2.7. Cross-Site Scripting (XSS).....	20
2.2.8. Desserialização Insegura	21
2.2.9. Utilização De Componentes Vulneráveis	21
2.2.10. Registro E Monitoramento Insuficientes.....	22
2.3. FERRAMENTAS AUTOMATIZADAS PARA TESTE DE SEGURANÇA....	22
3. ANALISE/DESENVOLVIMENTO	26
3.1 METODOLOGIA	26
3.4 RESULTADOS ALCANÇADOS	29
3.4.1 RESULTADO DO TESTE DA AP1.....	33
3.4.2 RESULTADO DO TESTE DA AP2.....	37
3.4.3 COMPARATIVO DOS RESULTADOS.....	41
4. CONSIDERAÇÕES FINAIS/CONCLUSÃO.....	43
REFERÊNCIAS BIBLIOGRÁFICAS	45

1. INTRODUÇÃO

Processo de software é definido como um conjunto de atividades, ações e tarefas realizadas na criação de algum artefato (PRESSMAN, 2016, p.16). Dentro do contexto da segurança da informação, a aplicação destes processos é de grande valia para que o sistema desenvolvido atenda aos requisitos da segurança da informação.

Um processo de software não precisa necessariamente ser seguido à risca, ou seja, pode ser adaptado as necessidades da organização (PRESSMAN, 2016). No entanto, é necessário garantir que um processo seja utilizado, avaliado e monitorado continuamente, para identificar possíveis melhorias e implementá-las através de medições chamadas pela engenharia de software de métricas de software, garantindo a melhoria contínua do processo e consequentemente aumentando a qualidade do produto final. “A medição nos permite obter o entendimento do processo e projeto, dando-nos um mecanismo para uma avaliação objetiva” (PRESSMAN, 2011, p.583).

Sabendo da importância de tais processos para a segurança dos sistemas *web* da atualidade, foi desenvolvido um trabalho pelos graduados Jean Carlos Gomes Martins e Vanessa Mota Alves no segundo semestre de 2017, no curso de Bacharelado de Engenharia de Computação da UniEVANGÉLICA, cujo tema foi “Processo de gestão de segurança da informação: definição e aplicação em um sistema laboratorial”. Na oportunidade, foi proposto um processo de gestão para colaborar com a tríade da segurança da informação (confidencialidade, integridade e disponibilidade).

Porém, neste processo, não foram definidas as métricas para avaliar o nível da segurança da informação do produto de software. Mediante ao processo proposto, e considerando a crescente quantidade de informação sendo gerada e processada a todo segundo é fundamental que as empresas e desenvolvedores adotem boas práticas de segurança da informação em seus cotidianos, como por exemplo, utilizar um processo de gestão, capaz de orientar a identificação e correções das falhas de segurança.

Entretanto, somente utilizar um processo de gestão no desenvolvimento de uma aplicação *web* não garante que ela seja totalmente segura. “Se você não medir, o julgamento só pode ser baseado em uma avaliação subjetiva” (PRESSMAN, 2011,

p.583), surgindo assim, dúvidas em relação a qualidade do produto final e a segurança das informações.

Portanto, para assegurar o máximo de segurança possível é necessário medir o produto criado e as métricas de software são uma estratégia relevante. Neste cenário, levanta-se a seguinte questão, quais métricas de software são necessárias para avaliar se um sistema *web* atende a tríade da segurança da informação?

Mediante o problema exposto, foi delineado o seguinte objetivo: avaliar o nível de segurança da informação de sistemas *web*, utilizando métricas de software. Para isso, foi delineado os seguintes objetivos específicos: (a) estudar um processo de gestão da segurança da informação que visa a melhoria continua dos processos de desenvolvimento de software *web*; (b) definir métricas para a segurança da informação em sistemas *web*; (c) aplicar as métricas definidas em sistemas *web*; e (d) incrementar melhorias identificadas ao processo de gestão estudado.

Para tanto, este trabalho tem como justificativa o fato da segurança da informação ter se tornado uma preocupação de todas as partes envolvidas no processo de criação e disponibilização da informação.

Considerando o contexto até aqui apresentado, a segurança está diretamente ligada à proteção de qualquer tipo de informação de agentes que não possuem permissão para acessar determinada informação. “A segurança é um atributo do sistema que reflete sua capacidade de se proteger de ataques externos, sejam acidentes ou deliberado” (SOMMERVILLE, 2011, p.211).

Para isso, as empresas devem se certificar que o produto desenvolvido possua um nível de segurança considerável, pois um projeto pode vir ao fracasso por não atender as necessidades da segurança da informação, e garantir que as informações geradas pelos seus usuários não caiam em mãos erradas.

Sem um nível razoável de proteção os usuários não se sentem confiantes quanto à disponibilidade, à confiabilidade e a segurança, fazendo com que abandonem o uso destes sistemas (SOMMERVILLE, 2011, p.213). Portanto, atingir um nível de qualidade da segurança da informação de um produto de software *web* se torna uma preocupação das empresas.

Com isso, as organizações produtoras de software devem utilizar em seu processo de desenvolvimento de software, um processo de gestão que visa melhorar a qualidade do produto no quesito segurança da informação. Entretanto,

somente utilizar um processo de gestão não fica evidente qual é o nível de segurança que o produto final adquiriu. É necessário, que existam métricas para que essa verificação possa ser feita de forma precisa. Além disso, é necessário que o processo de gestão seguido seja monitorado e avaliado continuamente para que diante desta avaliação possa se implementar as melhorias identificadas.

Entretanto, na literatura não foram encontradas métricas bem definidas que evidenciam o nível da segurança da informação dos sistemas *web* da atualidade. Mediante as informações apresentadas, este trabalho visa avaliar o nível de segurança da informação de produtos de software *web*, a partir da definição de métricas e análise dos resultados das medições, para oferecer feedbacks e sugerir implementações de mudanças.

Pretende-se que os resultados colaborem, com a segurança da informação e a continuidade da operação destes sistemas, vez que, os proprietários destas aplicações estarão cientes que as suas soluções necessitam de correções.

Este trabalho está organizado em quatro seções. Sendo a primeira seção, a introdução, que aborda a visão geral da pesquisa. A segunda seção, fundamentação teórica, que apresenta os conceitos e informações que subsidiam o tema. A terceira seção, análise e desenvolvimento, que apresenta a estrutura do estudo de caso e os resultados alcançados. E, por fim, a quarta seção, que contempla as considerações finais.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. SEGURANÇA DA INFORMAÇÃO

Um sistema de informação consiste em um conjunto composto por hardware, sistema operacional e software aplicativo, que trabalham juntos para coletar, processar e armazenar dados para indivíduos e organizações (SOLOMON; KIM, 2014). Um sistema *web* também é considerado um sistema de informação, porém a sua utilização é através do navegador, tornando-o mais acessível e, ao mesmo tempo, mais vulnerável e suscetível à ataques.

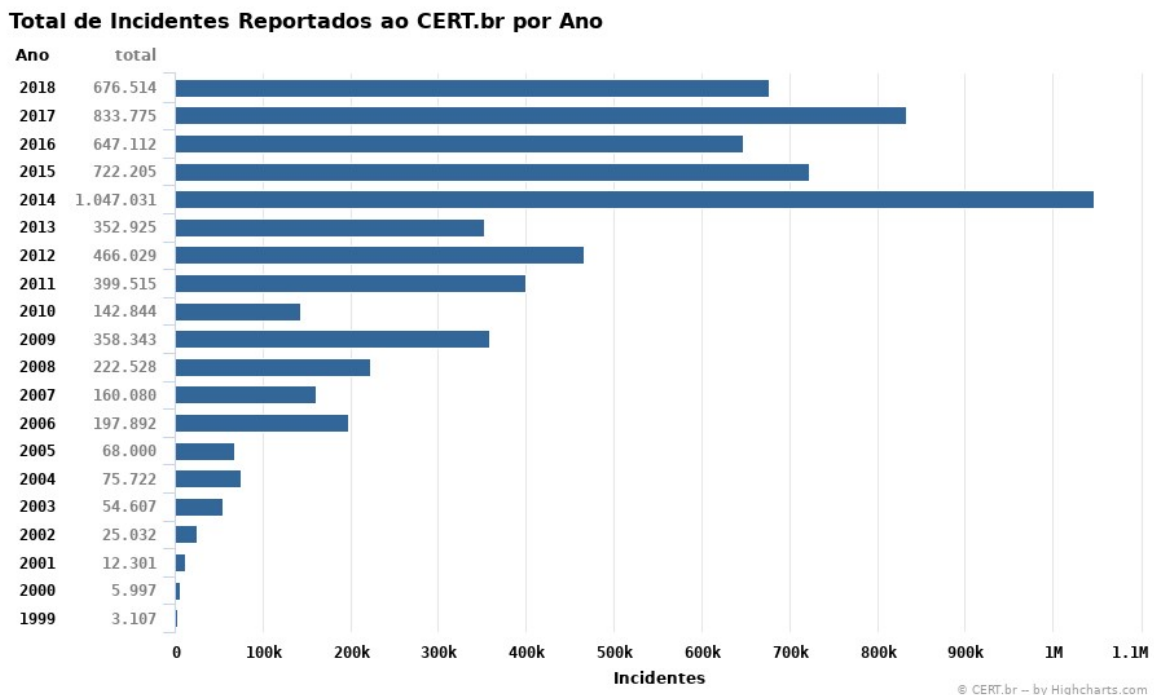
Vulnerabilidade por sua vez é entendida segundo a norma ABNT ISO/IEC 17799 (ABNT, 2005), como sendo uma fragilidade de um ativo ou grupo de ativos que pode ser explorada por uma ou mais ameaças. Já uma ameaça é um possível perigo que pode explorar uma vulnerabilidade, ou seja, qualquer ação que possa comprometer a segurança da informação e causar prejuízos à organizações e pessoas (STALLINGS, 2008). Já o *hacker* é a figura responsável por ocasionar uma ameaça ao explorar as vulnerabilidades do sistema através de um ataque.

Ataques por sua vez podem acontecer de diferentes maneiras, e é entendível, como sendo técnicas que os invasores usam para explorar as vulnerabilidades dos sistemas (SANS, 2008b,a; Stuttard and Pinto, 2007).

Com a crescente quantidade de sistemas *web* sendo desenvolvidos e disponibilizados ao público em geral a todo o momento é necessária atenção redobrada na segurança das informações consumidas e fornecidas pelos usuários, para que as vulnerabilidades encontradas em tais sistemas sejam identificadas ainda na fase de desenvolvimento.

Observa-se na Figura 1 a crescente quantidade de incidentes ocorridos no decorrer dos anos mediante a popularização de sistemas computacionais.

Figura 1- Quantidade de incidentes (2019)



Fonte: <https://www.cert.br/stats/incidentes/>

A CERT não explica os motivos das variações dos incidentes reportados, e não foram encontrados dados passíveis para a realização de análise quanto a esta variação.

OWASP - *Open Web Application Security Project* é uma comunidade aberta dedicada a permitir que as organizações desenvolvam, adquiram e mantenham aplicações e APIs confiáveis, OWASP (2017).

A OWASP publicou uma lista atualizada com os dez principais riscos mais recorrentes e que causam mais prejuízo aos negócios, sendo eles: riscos de injeção, quebra de autenticação, exposição de dados sensíveis, entidade externas de XML, quebra de controle de acesso, configuração de segurança incorretas, cross-site scripting (XSS), desserialização insegura, utilização de componentes vulneráveis e registro e monitorização insuficiente.

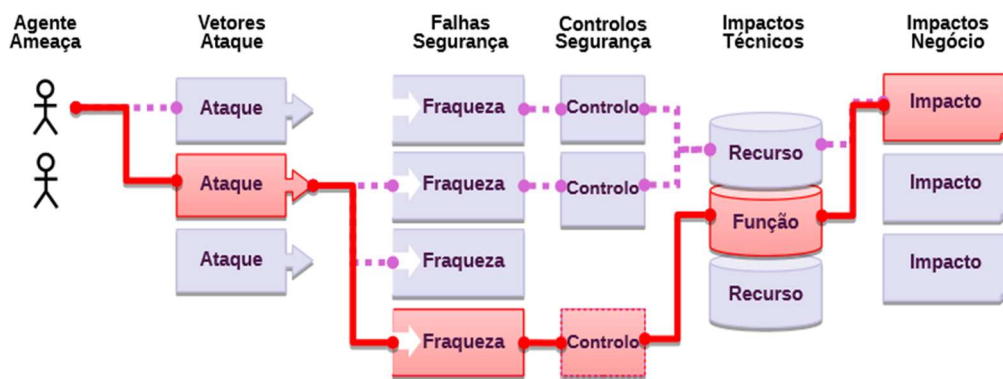
A proteção da informação contra esses tipos de ataque é de fundamental importância para a continuação dos negócios. A informação é um ativo que, como qualquer outro ativo importante, é essencial para os negócios de uma organização e conseqüentemente necessita ser adequadamente e devidamente protegida. Sendo

compreendido também, como um conjunto de dados responsável por constituir uma mensagem (ABNT ISO/IEC 17799, 2005).

“Segurança da informação é a proteção da informação de vários tipos de ameaças para garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio” (ABNT ISO/IEC 17799, 2005, p.9).

Pode-se observar na Figura 2 como acontece um ataque, primeiramente o agente malicioso deve possuir um mecanismo para efetuar o ataque, podendo ser feito por ferramentas automatizadas ou por um processo manual, o próximo passo é identificar as falhas de segurança existentes no alvo, para que a exploração dessas falhas aconteça e se tenha controle do sistema ou obtenção de informações sensíveis, os próximos passos dizem respeito aos prejuízos que determinada falha de segurança pode ocasionar, desde prejuízos técnicos como, a exclusão de um banco de dados, ou prejuízos aos negócios, como o sequestro de informações importantes.

Figura 2 - Fluxo de ataque.



Fonte: OWASP (2017).

Visto a importância da segurança da informação para os sistemas *web* é necessário que exista uma forma de identificar as vulnerabilidades e certificar o nível de segurança que tal sistema possui.

Para isso, o nível de proteção pode ser avaliado, analisando a confidencialidade, a integridade e a disponibilidade da informação, bem como quaisquer outros requisitos que sejam considerados (ABNT ISO/IEC 17799, 2005, p.23).

Vale ressaltar, que a engenharia social é um dos tipos de ataques mais poderosos atualmente, que consiste na manipulação de uma pessoa (usuário do sistema) com a intenção de extorquir dados e obter acesso a informações não autorizadas.

“Em quase todos os casos, a engenharia social envolve em enganar usuários autorizados para que executem ações para usuários não autorizados” (KIM; SOLOMON, 2014, p.81). Entretanto, este tipo de ataque vai além do escopo desta pesquisa, vez que, diz respeito a políticas de segurança adotadas pela organização para treinar o pessoal contra este tipo de ataque.

2.2. PRINCIPAIS VULNERABILIDADES EM APLICAÇÕES WEB

Existem milhares de vulnerabilidades assombrando as aplicações *web* e contribuindo para a crescente quantidade de incidentes vistos anteriormente. Os riscos expostos anteriormente são os principais levantados pela OWASP (2017), comunidade online que cria e disponibiliza material para auxiliar na segurança da informação de aplicações *web*.

Estes riscos foram baseados no nível de impacto que tais vulnerabilidades trazem aos negócios e servirão para nortear o objetivo da pesquisa, sendo assim, será feito um detalhamento para cada risco. Vale ressaltar que o impacto causado por estas vulnerabilidades depende do tipo de negócio, visto que, a existência de uma vulnerabilidade não significa um ameaça imediata.

2.2.1. Injeção

Segundo a OWASP (2017), injeção, pode ser compreendido como o envio de dados não confiáveis para um interpretador, para executar um comando ou uma consulta. Os tipos de injeção mais comuns são SQL, NoSQL, comandos do sistema operativo, ORM, LDAP, linguagem de expressão (EL) ou injeção de OGNL e é mais recorrente em códigos legados.

O impacto ao negócio depende das necessidades de proteção dos dados, visto que, a injeção pode levar ao controle total do sistema ou resultar em perdas de dados significativos.

Como prevenção sugerida é recomendável utilizar APIs que evitam por completo o uso de interpretadores, utilizar ferramentas de ORM, validar os dados de

entradas de usuário e usar estratégias com o LIMIT para evitar a exposição de grandes volumes de dados.

2.2.2. Quebra De Autenticação

A quebra de autenticação é proveniente de algoritmos relacionados à autenticação mal construídos, e permite ao atacante assumir a identidade de um usuário do sistema de forma temporária ou permanente.

O controle de sessão é o pilar da autenticação e está presente em todas as aplicações que guardam estados, para isso, os atacantes podem quebrar autenticações através de processos manuais, ou fazendo uso de ferramentas automatizadas com listas de senhas fracas e comuns, “uma aplicação que não trata adequadamente os erros que ocorrem está sujeita a diversos riscos de segurança, incluindo indisponibilidade e quebra de mecanismos de proteção” (UTO; MELO, 2015, p.263).

A prevenção sugerida se resume em limitar o número máximo de tentativas de autenticação malsucedidas e sempre que identificado grandes números de tentativas, deve-se alertar os administradores do sistema, não disponibilizar aplicações com credenciais pré-definidas e implementar verificações de senhas fracas.

2.2.3. Exposição De Dados Sensíveis

A exposição de dados sensíveis, tais como dados pessoais e financeiros, consiste na falta de proteção desses tipos de dados contra pessoas não autorizadas, sendo assim os atacantes podem obter tais informações e até mesmo modifica-las, sendo a falha mais comum a não encriptação de dados (MONTEVERDE, 2014).

Diferente da quebra de autenticação na qual é atacado diretamente a criptografia, nesta modalidade os atacantes tendem a obter acesso através de dados expostos indevidamente, mesmo quando é utilizado a criptografia os dados podem ser vulneráveis em casos de criptografia fraca (VIJAYARANI; TAMILARASI, 2011)

Para isso as dicas sugeridas de prevenção, como não armazenar dados sensíveis sem necessidade, garantir que todos os dados sensíveis sejam encriptados e utilização de protocolos de segurança como TLS e HTTPS.

2.2.4. Entidades Externas De XML (XXE)

Vulnerabilidade de entidade externas de XML diz respeito ao uso de processadores XML antigos ou mal configurados que avaliam referências a entidades externas dentro dos documentos XML, sendo assim, permite que diversos outros tipos de ataques sejam realizados, como por exemplo, ataques de negação de serviço, tal como, billion laughs OWASP (2017).

As prevenções sugeridas são, optar por um formato de dados mais simples como, por exemplo, o formato JSON e desativar o processamento de entidades externas.

2.2.5. Quebra De Controle De Acesso

A quebra de controle de acesso consiste na tentativa de um usuário não autorizado tentar ter acesso às informações que necessitam de permissão para serem consumidas. “As restrições sobre o que os utilizadores autenticados estão autorizados a fazer nem sempre são corretamente verificadas” OWASP (2017). Com isso, está vulnerabilidade permite ao atacante, ter acesso às informações sensíveis, modifica-las e até mesmo alterar permissões de acesso de outros usuários.

2.2.6. Configurações De Segurança Incorretas

Configurações de segurança incorretas é o aspecto mais observado nos dados analisados pela OWASP, e consiste em configurações padrões inseguros, armazenamento em nuvem sem qualquer tipo de controle de acesso, mensagem contendo informações sensíveis e uso de HTTP mal configurado (MONTEVERDE, 2014).

Como forma de prevenção desse tipo de ataque é recomendado, remover qualquer tipo de *framework*, biblioteca ou funcionalidade não utilizada, sempre manter as configurações atualizadas de forma adequada e nunca revelar informações de configuração através de mensagens de erro.

2.2.7. Cross-Site Scripting (XSS)

Ataques de XSS acontecem sempre que a aplicação permite incluir dados não confiáveis em uma página *web* sem que haja qualquer tipo de validação,

possibilitando assim, o atacante executar scripts no *browser* da vítima, podendo capturar dados armazenados em sessão, roubar credenciais ou redirecionar a vítima para outros endereços, dando margem para diversos outros tipos de ataques (UTO; MELO, 2015).

Existem três tipos de ataque por XSS, sendo o primeiro deles, o Reflected XSS, que permite a execução de código javascript e HTML, possibilitando o redirecionamento para um novo endereço controlado pelo atacante, seu impacto é moderado. O segundo tipo, conhecido como *Stored XSS*, diz respeito à aplicação armazenar dados de entrada do utilizador sem qualquer tipo de tratamento, permitindo roubo de credenciais e sessões, seu impacto é considerado alto. Já o terceiro tipo, conhecido como Dom XSS, tem um impacto moderado, consistindo na manipulação DOM e tipicamente acontece em aplicações que utilizam frameworks javascript, Single Page Applications (SPA) e APIS que manipulam a página dinamicamente (UTO; MELO, 2015).

Como formas de prevenção é sugerido, utilizar *frameworks* que oferecem proteção contra esse tipo de ataque e tratamento das informações de entrada no pedido HTTP.

2.2.8. Desserialização Insegura

A aplicação é vulnerável a esse tipo de ataque se ocorrer a desserialização de dados não confiáveis ou objetos adulterados disponibilizados pelo atacante. Essa modalidade de ataque pode levar a execução de código remoto e até mesmo ataques de injeção. Apesar de existir ferramentas que identificam esta vulnerabilidade à assistência humana é necessária (OWASP, 2017).

As formas de prevenção sugeridas pela OWASP (2017), são, aceitar somente objetos serializados de fontes seguras e tipos de dados primitivos, caso não seja possível, deve-se verificar a integridade do objeto serializado como, por exemplo, o uso de assinatura digital.

2.2.9. Utilização De Componentes Vulneráveis

Está vulnerabilidade diz respeito ao uso de bibliotecas e frameworks que possuem vulnerabilidades conhecidas, ocasionando perdas de dados significativos e até mesmo possibilitando o controle completo do servidor. Está vulnerabilidade é

mais recorrente em aplicações que utilizam padrões de projeto baseadas em um uso extensivo de componentes OWASP (2017).

Para isso as prevenções sugeridas, são, remoção de componentes, bibliotecas e *frameworks* não utilizados ou desatualizados, utilizar ferramentas automatizadas para identificar vulnerabilidades em componentes, obter componentes apenas de fontes oficiais e monitorar constantemente novas versões destes componentes OWASP (2017).

2.2.10. Registro E Monitoramento Insuficientes

O abuso do registro e monitoramento insuficientes que dão margem para quase todos os outros tipos de ataques. Os atacantes aproveitam da falta de monitorização para efetuarem seus ataques sem serem detectados, possibilitando a extração, alteração e exclusão de dados. A OWASP (2017) diz que esta vulnerabilidade possui um tempo de detecção de mais de duzentos dias, e pode ser detectável com uma análise dos ficheiros do servidor após uma série de testes de intrusão.

As sugestões de prevenção da OWASP (2017), são, assegurar que todas as autenticações, falhas no controle de acesso e falhas na validação de dados de entrada sejam registados com um nível de detalhamento suficiente para identificar o utilizador e utilizar processos de monitorização e alerta capazes de identificar atividades suspeitas e utilizar uma metodologia como a NIST 800-61 para respostas de incidentes e plano de recuperação.

2.3. FERRAMENTAS AUTOMATIZADAS PARA TESTE DE SEGURANÇA

Mediante a crescente quantidade de incidentes, conforme apresenta a Figura 1 e a lista com as dez principais vulnerabilidades na segurança de sistemas *web*, fica evidente a necessidade de atenção com a segurança da informação. Para identificar as vulnerabilidades citadas, torna-se mais eficiente fazer o uso de ferramentas automatizadas que possibilitarão a maximização do tempo e servirão para fazer uma varredura no sistema *web*, identificando as possíveis vulnerabilidades existentes.

Atualmente, há diversas ferramentas disponíveis de uso gratuito e de fácil operabilidade. Para a identificação de vulnerabilidades na aplicação *web* foi utilizada a ferramenta OWASP ZAP, de propriedade da OWASP, comunidade que está inserida e consolidada no cenário da segurança da informação e que cria e disponibiliza ferramentas e artefatos para colaborar com a segurança de dados. “O ZAP Attack Proxy (ZAP) da OWASP é uma das ferramentas de segurança gratuitas mais populares do mundo e é mantido ativamente por centenas de voluntários internacionais” (ZAP, 2019), procurando pelas vulnerabilidades mais conhecidas e recorrentes da atualidade, fazendo diversas tentativas de ataque e identificando e sinalizando as vulnerabilidades encontradas.

Um dos recursos mais interessantes que esta ferramenta possui, são as sugestões disponibilizadas para cada vulnerabilidade encontrada, tornando assim o trabalho de correção dessas falhas de segurança mais fácil. Estas sugestões disponibilizadas pela ferramenta, estão em conformidades as sugestões de prevenção e correção expostas anteriormente no detalhamento de cada falha de segurança.

Já para identificar informações vitais do servidor em que as aplicações estão hospedadas, foi escolhida a ferramenta NMAP, visto que, é uma das ferramentas mais difundidas, simples e fáceis de usar. Na qual temos como resultado da varredura, quais as portas estão abertas, quais serviços estão sendo executados e, muitas das vezes a versão desses serviços.

Com os dados da varredura do NMAP, o agente malicioso obtém de forma bastante prática e rápida, informações importantes do servidor. Podendo utilizar essas informações para explorar uma vulnerabilidade já conhecida, como no caso de uma versão de um servidor WEB desatualizado, na qual as falhas de segurança são divulgadas no patch de atualização, tornando fácil a exploração dessas falhas.

2.4. MÉTRICAS DE SOFTWARE

A segurança da informação deve garantir os três pilares fundamentais da informação segura, conhecida como a tríade C-I-D, assim como explica KIM e SOLOMON (2014):

- A disponibilidade consiste na informação acessível sempre que solicitado por usuários autorizados.

- A confidencialidade consiste na visualização da informação somente por usuários autorizados.
- A integridade cuida da validade, precisão dos dados e podem ser alterados somente usuários autorizados.

Mediante as informações apresentadas, para certificar que o sistema *web* atenda a tríade C-I-D (Confidencialidade, Integridade, Disponibilidade) é necessário que durante o processo de desenvolvimento seja adicionado segurança aos requisitos do sistema. Para isso é fundamental utilizar um processo de gestão da segurança da informação, garantindo que as principais vulnerabilidades sejam reconhecidas e solucionadas ainda na fase de desenvolvimento.

É importante que cada característica relevante de qualidade do produto de software seja especificada e avaliada utilizando, quando possível, métricas validadas ou amplamente aceitas (ISO/IEC 9126-1, p.2).

Um engenheiro de software coleta medidas e desenvolve métricas para obter indicadores. Um indicador é uma métrica ou combinação de métricas que proporcionam informações sobre o processo de software ou no próprio produto. Um indicador proporciona informações que permitem ao gerente de projeto ou aos engenheiros de software ajustar o processo, o projeto ou o produto para incluir melhorias (PRESSMAN, 2011, p.539).

Métricas por sua vez podem ser definidas segundo Pressman (2011), como sendo uma medida quantitativa, na qual dará origem a indicadores que proporcionam ao engenheiro de software informações sobre a qualidade do produto de software. A definição de métricas acontece de diversas maneiras, uma das mais aceitas no mercado é a utilização da metodologia GQM (*Goal Question Metric*).

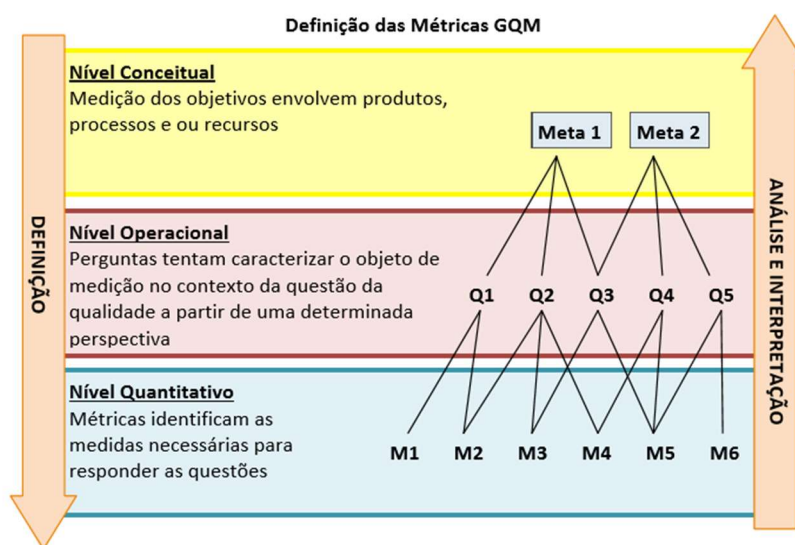
A metodologia GQM é uma abordagem de cima para baixo (*top-down*) usada para estabelecer um sistema de medição baseado em metas definidas em 3 níveis. As medições devem ser definidas de acordo com alguns objetivos específicos para melhorar a efetividade, portanto, o uso da metodologia GQM possibilita, definir, implantar, medir e melhorar os processos.

O GQM enfatiza a necessidade de (1) estabelecer um objetivo de medição explícita que é específico para a atividade do processo ou característica do produto que deve ser avaliada, (2) definir um conjunto de questões que devem ser respondidas para atingir o objetivo e (3) identificar métricas bem formuladas que ajudam a responder a essas questões, Pressman (2016, p.656).

A metodologia GQM define um modelo de três níveis hierárquicos, sendo o primeiro deles o nível conceitual, que tem como objetivo elaborar metas para o

processo ou produto de software, enfatizando o mesmo que o primeiro princípio de medição do PRESSMAN (2011), já no segundo nível, chamado de nível operacional, acontece a elaboração de perguntas que deverão ser respondidas para alcançar as metas definidas e no terceiro nível tem-se as métricas levantadas com o objetivo de responder as perguntas elaboradas no nível anterior.

Figura 3 - Níveis hierárquicos da metodologia GQM.



Fonte: UFPE, 2019.

O princípio de medição se embasa em cinco atividades, sendo elas a formulação que é a atividade de criação de medidas, a coleção que representa a coleta de dados para criar as métricas formuladas, a análise que é a atividade na qual se computa as métricas e aplica as ferramentas matemáticas, seguido da interpretação que avalia as métricas para resultar nas informações sobre a qualidade e por último a etapa de feedback que são as recomendações derivadas da interpretação das métricas (PRESSMAN, 2011).

Portanto, a definição de métricas de segurança são ferramentas de suma importância para que profissionais de segurança da informação consigam avaliar os níveis de segurança de seus sistemas, produtos e processos, e podem ser usadas para identificar vulnerabilidades e avaliar os riscos, orientando ações corretivas de maior prioridade e aumentando o nível de maturidade da segurança (BATISTA, 2007, p38).

3. ANALISE/DESENVOLVIMENTO

A presente seção está dividida em dois tópicos, sendo o primeiro tópico, o desenho metodológico, que irá descrever os procedimentos que foram realizados, para que os objetivos deste trabalho fossem alcançados, e o segundo tópico, contempla os resultados alcançados.

3.1 METODOLOGIA

Esta pesquisa se caracteriza como um estudo exploratório do tipo quanti-quali, com análise documental e realização de estudo de caso.

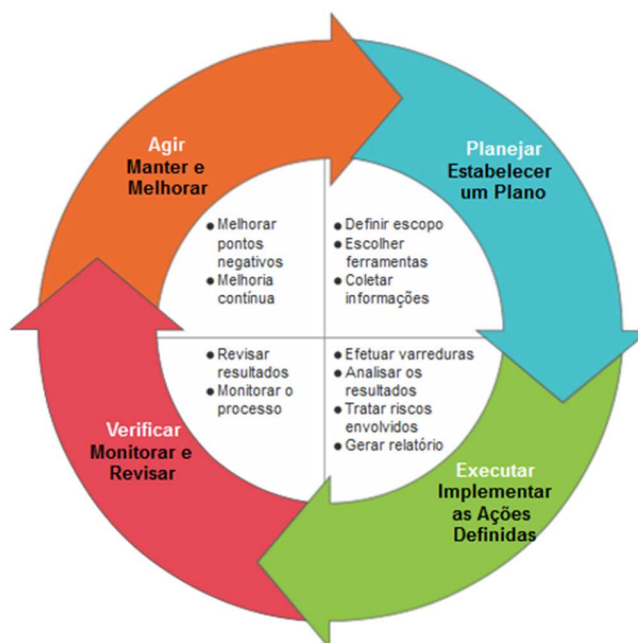
A partir daqui, para fins didáticos e de melhor entendimento, o processo de gestão estudado vai ser chamado de processo X.

Para que os objetivos deste trabalho fossem alcançados, foi necessário estudar o processo X (ALVES; MARTINS, 2017), com o fim de identificar possíveis melhorias no contexto da medição da segurança, e definir métricas de software que possam avaliar o nível de segurança da informação no produto de software *web*.

Para isso, a revisão da literatura, normas e guias de boas práticas altamente aceitos também foram analisados. O processo X foi estudado seguindo a metodologia GQM (*Goal Question Metric*), que possibilitou a criação de metas a partir de uma leitura crítica da literatura e do processo de gestão. Posteriormente foi criado e respondido os questionamentos, para alcançar as metas definidas, que resultaram na elaboração e na identificação de métricas. Essas métricas foram aplicadas em dois sistemas *web* em uso.

Na Figura 4 é possível observar o ciclo de vida do processo X, na qual contempla o planejamento, a coleta, a análise, as contramedidas, a revisão dos resultados e a adição de melhoria.

Figura 4 - Fluxo do processo de gestão estudado.



Fonte: (MARTINS; ALVES, 2017).

A definição das métricas se deu a partir de seis etapas estabelecidas na metodologia GQM, sendo elas: 1 - definição de metas; 2 - geração de perguntas; 3 - especificação das medidas necessárias para se obter as respostas; 4 - desenvolvimento de um mecanismo para coleta de dados; 5 - validação dos dados coletados e, 6 - análise dos dados em conformidade com as metas.

Para validar as métricas propostas foi necessário seguir um modelo de avaliação na qual contemplou os atributos que todas as métricas de software devem possuir. Segundo (PRESSMAN, 2011, p.542), os atributos necessários que as métricas devem possuir são:

- “Simples e computáveis.
- Empiricamente e intuitivamente persuasiva.
- Consistente no seu uso das unidades e dimensões.
- Independente da linguagem de programação.
- Um mecanismo efetivo para feedback de alta qualidade”.

Toda métrica identificada na etapa anterior contemplou os atributos necessários que uma métrica de software deve possuir. Resultando dessa forma, em métricas confiáveis e um sistema de medição preciso.

Após a definição de métricas, estas foram aplicadas através de ferramentas automatizadas, em dois sistemas *web*, sendo possível obter os dados necessários para demonstrar os resultados da análise, da medição, da necessidade e da importância de se medir a qualidade da segurança da informação destas aplicações.

A aplicação das métricas identificadas, contou com o uso de ferramentas automatizadas que realizou testes nos sistemas *web* e, a partir dos resultados de tais testes, foi possível avaliar o nível da segurança da informação dos sistemas *web* em questão.

Para isso, as principais ferramentas automatizadas de uso gratuito e de fácil operabilidade foram selecionadas e utilizadas, com o fim de se aplicar as métricas identificadas nos sistemas *web*. O uso dessas ferramentas automatizadas proporcionou a maximização do tempo para a realização do estudo de caso, facilitando a identificação de vulnerabilidades.

Os sistemas *web* em questão estavam hospedados e disponíveis aos usuários, pois dessa forma, tem-se um cenário mais próximo do ambiente de produção de tais aplicações.

A primeira ferramenta selecionada, OWASP ZAP, foi utilizada para identificar as possíveis vulnerabilidades nas aplicações *web*. Sendo assim, foram minimizadas configurações específicas na ferramenta, visto que, o intuito é que a aplicação dessas métricas possam ser replicadas em outros contextos.

Portanto, foi necessário apenas configurar o *proxy* do navegador e especificar acessos (usuário e senha), para que a ferramenta pudesse explorar as demais partes do sistema, que são restringidos à usuários autenticados. Também foi alterado a força do ataque, para “Alto”, fazendo com que a ferramenta aumente a quantidade de requisições feitas à aplicação, com o objetivo de identificar a maior quantidade de vulnerabilidades possível, ou seja, a diferença entre os tipos de força, baixo, médio e alto se resumem à quantidade de requisições executadas.

O teste, com força de ataque “Médio”, teve duração de aproximadamente duas horas para ambos os sistemas testados. Quando alterado para a força “Alto”, a duração é estendida para aproximadamente vinte e quatro horas, fazendo com que as possibilidades de encontrar uma falha de segurança sejam maiores, vez que, o teste é feito de forma mais agressiva.

As aplicações testadas não estavam hospedadas no protocolo HTTPS. Sendo assim, não foi necessária configuração adicional. A partir daí, foi preciso informar a *URL* da aplicação alvo e aguardar o término da varredura.

A segunda ferramenta selecionada, NMAP, foi utilizada para coletar informações vitais do servidor, como versões de sistemas operacionais, portas abertas e serviços em execução. Para esta coleta foi informado o endereço de IP da aplicação alvo, sem a necessidade de configurações adicionais.

A partir da inserção do IP, a varredura foi iniciada no modo *Intense scan* com o objetivo de obter a maior quantidade possível de informações sobre o servidor.

3.4 RESULTADOS ALCANÇADOS

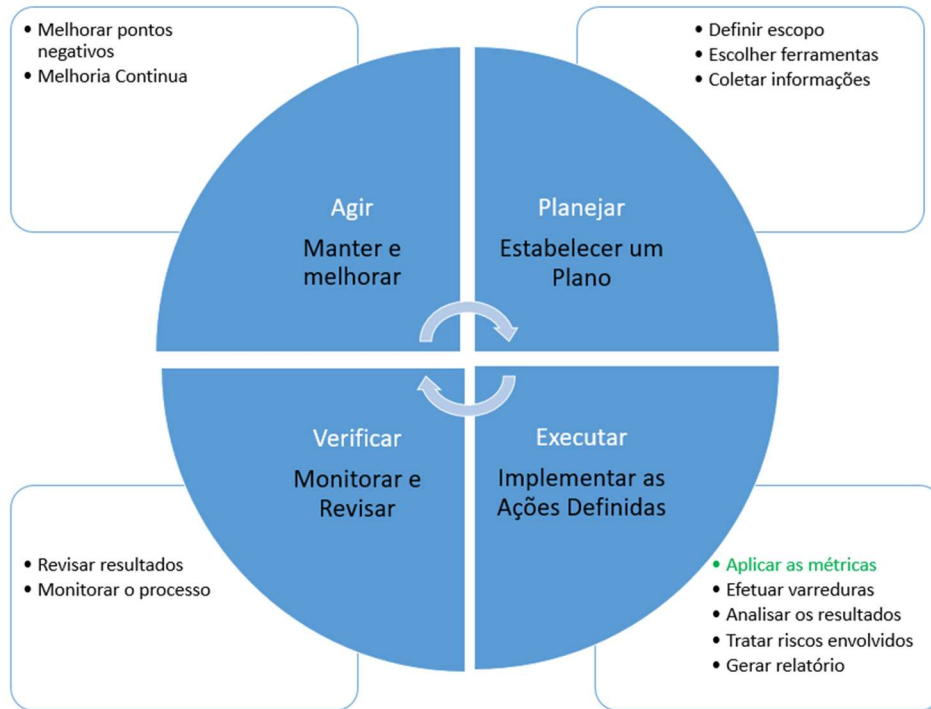
Através do estudo do processo X e da revisão da literatura, foi possível identificar os principais riscos que muitas aplicações *web* estão sujeitas a possuir, assim como as contramedidas necessárias para correção desses riscos.

Para atingir os objetivos específicos um¹ e quatro², o processo de gestão foi analisado e, como proposta de melhoria, foi acrescentado uma atividade ao ciclo de vida de tal processo. Pode-se observar na Figura 5, na etapa de executar, que a atividade “Aplicar métricas” foi adicionada.

¹ Objetivo específico um – Estudar um processo da segurança da informação.

² Objetivo específico quatro – Adicionar melhorias no processo X.

Figura 5 - Adição de melhoria ao processo X.



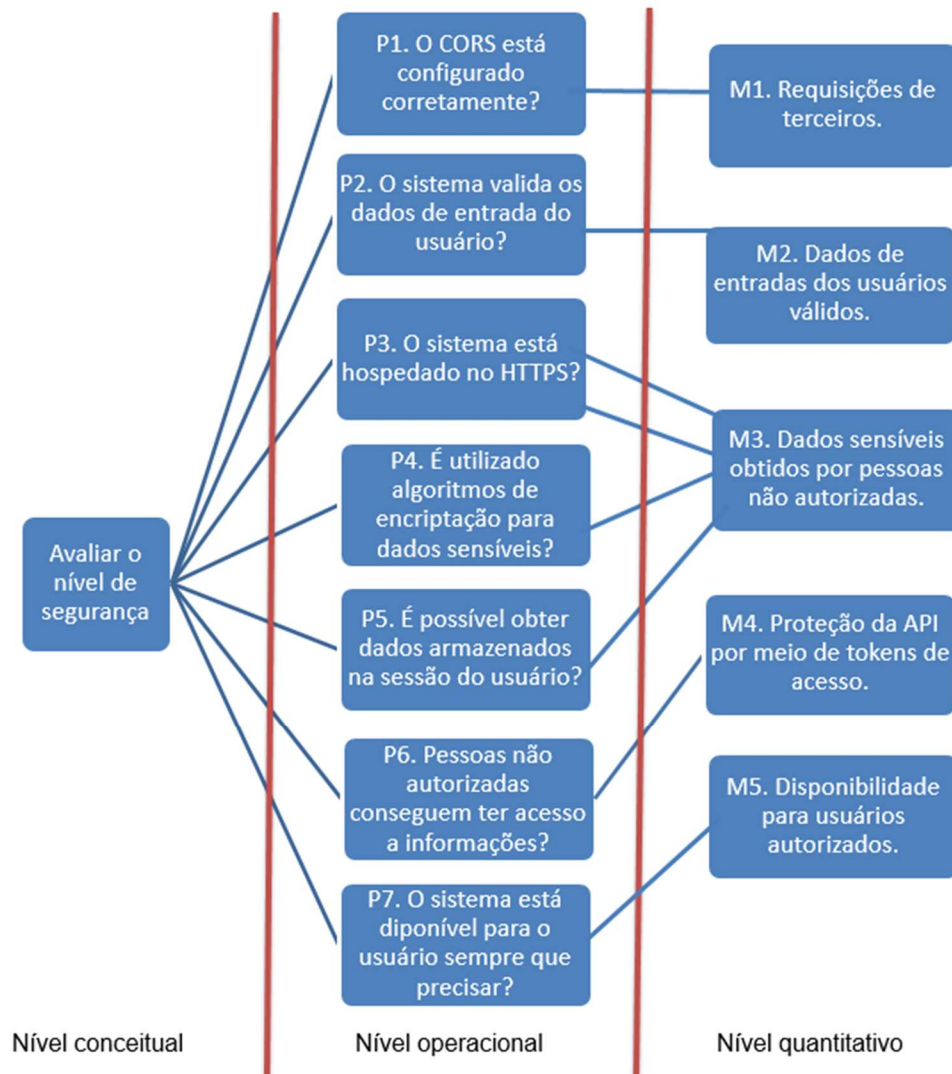
Fonte: Elaborado pelo autor.

Com base nisso, foi possível identificar informações suficientes para a definição de métricas que visam avaliar o nível de segurança de uma aplicação *web* e consequentemente atingir o objetivo específico dois³, assim como quais ferramentas seriam capazes de identificar e apontar a existências das possíveis falhas de segurança.

A Figura 6 apresenta as métricas identificadas. Vale enfatizar, que as métricas foram identificadas como foco nas principais vulnerabilidades existentes hoje e, mapeadas tendo em vista a tríade da segurança da informação.

³ Objetivo específico dois – Definir métricas para a segurança da informação.

Figura 6 - Métricas identificadas.



Fonte: Elaborado pelo autor.

O modelo apresentado na Figura 6 mostra o uso da metodologia GQM para identificação de métricas, sendo que no primeiro nível foi definida a meta para ser atingida, no segundo nível os questionamentos de como pode-se atingir a meta definida e, no terceiro nível, as métricas.

A métrica M1 trata da confidencialidade das informações, visto que, somente usuários autorizados podem obter determinada informação. Para isso, a aplicação não deve permitir que terceiros venham consumir os seus serviços. Portanto o

CORS⁴ deve ser configurado, com o fim de bloquear requisições de origens desconhecidas.

A métrica M2 trata a confidencialidade e a disponibilidade da informação, pois se não existe a validação dos dados inseridos pelo usuário, a aplicação pode possuir vulnerabilidades, tais como SQL INJECTION e XSS. Dessa forma, o agente malicioso conseguiria obter informações vitais e causar diversos danos aos dados coletados pela aplicação.

A métrica M3 trata da integridade como a confidencialidade dos dados, vez que, o sistema deve garantir que somente pessoas autorizadas tenham acesso à informação fidedigna. Para isso, dados sensíveis não devem ser expostos e mesmo que sejam expostos e interceptados, o atacante não deve conseguir interpretar e alterar a informação.

A métrica M4 trata da confidencialidade e da integridade da informação por parte do server. Com isso, a API só deve responder as requisições quando feitas por usuários que tenham permissão para acessá-las.

Por fim, a métrica M5 se refere à disponibilidade das informações, onde o sistema deve disponibilizar sempre que requisitado, por usuários que possuem permissão, as informações solicitadas.

Mediante as métricas definidas, a próxima etapa deste trabalho contemplou a realização do estudo de caso. Para isso, duas aplicações *web* foram disponibilizadas para a efetiva execução das métricas.

Para atingir o objetivo específico três, duas aplicações foram disponibilizadas. A primeira aplicação é de propriedade da empresa InLine Engenharia de Software, uma empresa situada na cidade de Anápolis-GO, produtora de soluções *web*. Tal aplicação foi hospedada em um servidor de teste. O Angular 5.0 foi a tecnologia de desenvolvimento FRONT-END e Java Spring Boot a tecnologia de desenvolvimento BACK END.

A segunda aplicação disponibilizada para o estudo de caso é de propriedade da FTT (Fábrica de Tecnologias Turing), unidade integrante dos cursos de Engenharia de Computação e Engenharia de Software da UniEVANGÉLICA, e se

⁴ CORS – CROSS-ORIGIN RESOURCE SHARING (Compartilhamento de recursos com origens diferentes) é um mecanismo que informa ao navegador que uma aplicação web pode executar uma requisição *cross-origin* HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem (MOZILLA, 2019).

trata de uma aplicação *web* desenvolvida em PHP para o Laboratório de Análises Clínicas do curso de Farmácia da UniEVANGÉLICA.

Para melhor expor as informações e para fins didáticos, a partir daqui, chamaremos a solução da InLine de “AP1”, e a aplicação disponibilizada pela FTT de “AP2”.

Visto as informações apresentadas até aqui e, para atingir o objetivo específico três⁵, o estudo de caso foi realizado. Para a realização do estudo de caso as métricas foram aplicadas, utilizando as ferramentas automatizadas escolhidas, nas duas aplicações *web* disponibilizadas. Foi utilizado um notebook Lenovo ideapad330, com 8GB de memória RAM, 240GB de SSD, processador Intel Core i7 8550U e placa de vídeo MX150.

A seguir, são descritas as análises dos resultados obtidos após a realização das atividades propostas na metodologia do trabalho.

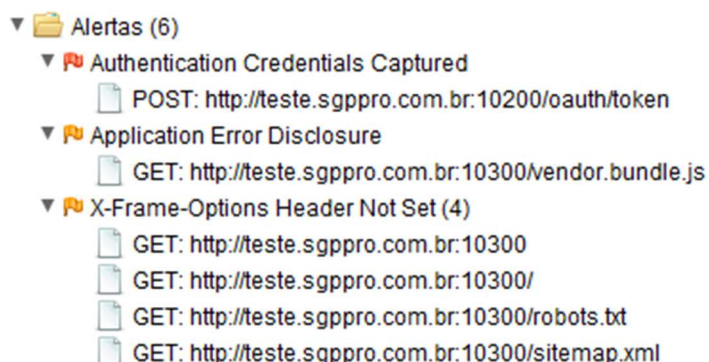
3.4.1 RESULTADO DO TESTE DA AP1

Através da ferramenta OWASP ZAP foi possível identificar algumas vulnerabilidades, que serão categorizadas com risco, alto, médio e baixo. As categorizações dos riscos supracitados são classificadas pela própria OWASP e se baseiam no impacto que a falha de segurança existente na aplicação pode causar à organização, ou seja, vulnerabilidades categorizadas com o risco alto, são consideradas muito prejudiciais, quanto as categorizadas com o risco baixo são consideradas pouco prejudiciais.

Na Figura 7 são expostos os alertas identificando as vulnerabilidades. Logo a seguir veremos suas respectivas explicações e sugestões de correção e prevenção.

⁵ Objetivo específico três – Aplicar as métricas propostas em sistemas *web*.

Figura 7- Vulnerabilidades da AP1



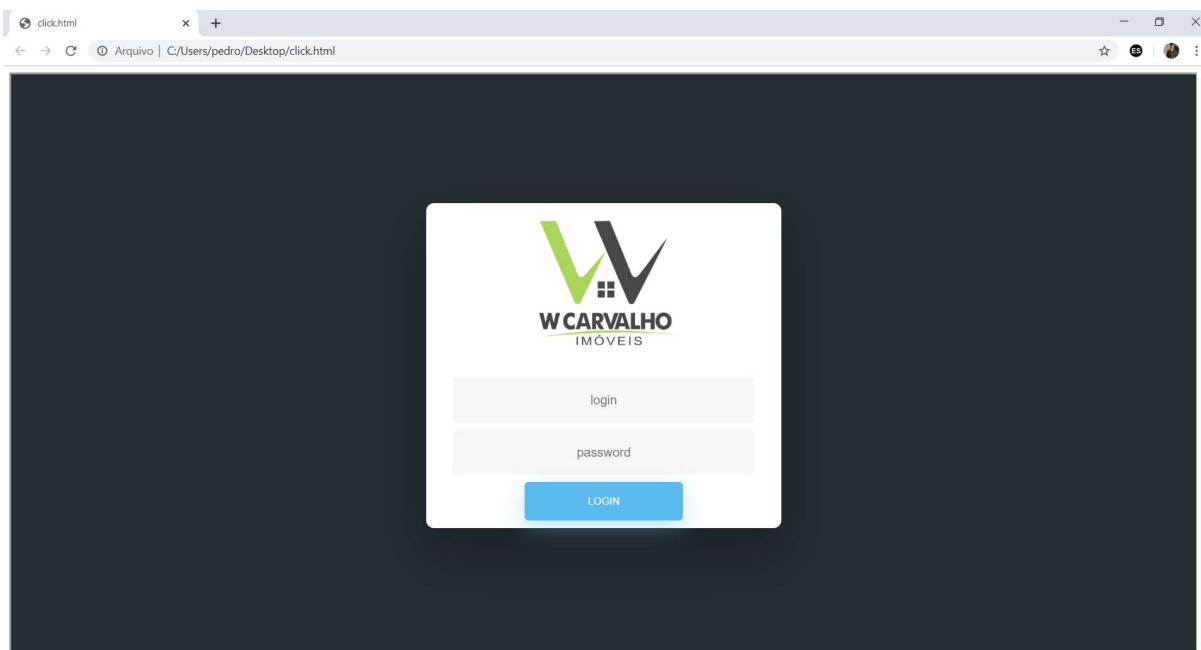
Fonte: Captura de tela da ferramenta OWASP ZAP.

Foram encontrados três tipos de vulnerabilidade. A seguir, será detalhado cada uma das falhas de segurança encontradas, em conformidade com as métricas proposta.

M1. Requisições de terceiros

A AP1 não possui o CORS configurado corretamente, para isso qualquer usuário ou agente malicioso poderá fazer requisições para a aplicação. Nesses casos, recomenda-se que o CORS seja configurado, com o objetivo de bloquear requisições de origens desconhecidas.

Outra vulnerabilidade identificada foi a, *X-Frame-Options Header Not Set*, categorizada, com o risco médio e diz respeito a possibilidade, do agente malicioso conseguir incorporar o conteúdo da aplicação em outras páginas, através de frames, como mostra a Figura 8, com o objetivo de explorar diversas outras vulnerabilidades, como por exemplo, fazer com que o usuário do sistema tente se autenticar, para que assim, o atacante consiga obter de forma fácil, o acesso ao sistema.

Figure 8 - *X-Frame-Option Header Not Set* da AP1

Fonte: Captura de tela Google Chrome.

Com prevenção dessa vulnerabilidade é necessário configurar o *X-Frame-Options*, que instruem os navegadores a bloquearem o uso de frames por domínios desconhecidos.

M2. Dados de entrada do usuário são válidos

A AP1 utiliza ORM (Object Relational Mapping), que se resume em uma tecnologia que representa as tabelas de um banco de dados relacionado em classes, sendo esta uma das técnicas recomendadas pela própria OWASP para evitar vulnerabilidades como *SQL INJECTION*. Portanto, não foi encontrada nenhuma falha de segurança.

A AP1 não permite que agentes maliciosos consigam obter dados da sessão do usuário, uma vez que o sistema não possui vulnerabilidades de XSS e valida os dados de entrada do usuário.

M3. Dados sensíveis obtidos por usuários não autorizados

Foi identificada a vulnerabilidade de *AUTHENTICATION CREDENTIALS CAPTURED*, com o risco categorizado como alto. Tal falha de segurança diz respeito ao fato da aplicação não criptografar os dados que serão utilizados para a requisição, desse modo, deixando dados sensíveis expostos, possibilitando ser obtidos por interceptação por agentes maliciosos, usando, por exemplo um *sniffer* na rede.

Existem ferramentas específicas para explorar esse tipo de vulnerabilidade, com por exemplo o WIRESHARK, que funciona como um *sniffer* na rede que interceptar todos os pacotes que estão trafegam, sendo assim, é possível obter os dados inseridos pelo usuário no formulário de autenticação ou em qualquer outra requisição.

Como solução e prevenção dessa vulnerabilidade, é recomendado utilizar algoritmos de criptografia, para os dados enviados ao servidor. Outra sugestão é hospedar a aplicação no protocolo HTTPS, fazendo com que o atacante, mesmo que consiga interceptar a informação, não irá conseguir interpretar e modificar os dados.

Outra vulnerabilidade identificada foi a de *Application Error Disclosure*, categorizada com o risco médio, e diz respeito à aplicação impedir o vazamento de informações, quando ocorre um erro. As mensagens de erro fornecem ao invasor uma excelente visão sobre o funcionamento interno de um aplicativo (OWASP, 2017). Sendo assim, torna-se necessário que a aplicação, trate todos os erros que podem acontecer e nunca apresente informações confidenciais aos usuários, quando esses erros acontecerem.

M4. Proteção da API por meio de *tokens* de acesso

A AP1 só atende as requisições de usuários autenticados, visto que, a aplicação utiliza *tokens* de acesso para validar a autorização da requisição, portanto, nenhuma informação foi obtida da API para usuário não autenticados, sendo assim a AP1, atende a métrica M4.

M5. Tempo diário que o sistema fica disponível para usuários autorizados

Tal métrica diz respeito à disponibilidade do sistema, sendo um dos pilares da segurança da informação, para isso, durante todo o período de teste o sistema se manteve disponível e estável, com isso, todas as requisições foram atendidas em tempo hábil, portanto, a AP1, atende à métrica M5.

Já no servidor foram identificadas onze portas abertas (9000, 8000, 7070, 5050, 3030, 3000, 1521, 443, 90, 80, 22) como mostra a Figura 9. Diversos serviços sendo executados, como também foi possível identificar a versão desses serviços através da ferramenta NMAP. No servidor em questão estavam hospedadas diversas aplicações por essa razão a quantidade de portas abertas é grande.

Figura 9 - Informações do servidor da AP1

Port	Protocol	State	Service	Version
10000	tcp	closed	snet-sensor-mgmt	
9090	tcp	closed	zeus-admin	
9000	tcp	open	cslistener	
8080	tcp	closed	http-proxy	
8000	tcp	open	rtsp	
7070	tcp	open	http	Apache httpd 2.4.38 ((Debian))
5050	tcp	open	http	nginx
3030	tcp	open	http	nginx 1.14.2
3000	tcp	open	http	nginx 1.8.1
2000	tcp	closed	cisco-sccp	
1521	tcp	open	oracle-tns	Oracle TNS listener 11.2.0.2.0 (unauthorized)
443	tcp	open	http	nginx
90	tcp	open	http	nginx 1.15.8
80	tcp	open	http	nginx 1.15.8
22	tcp	open	ssh	OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
21	tcp	closed	ftp	
20	tcp	closed	ftp-data	

Fonte: Captura de tela da ferramenta NMAP.

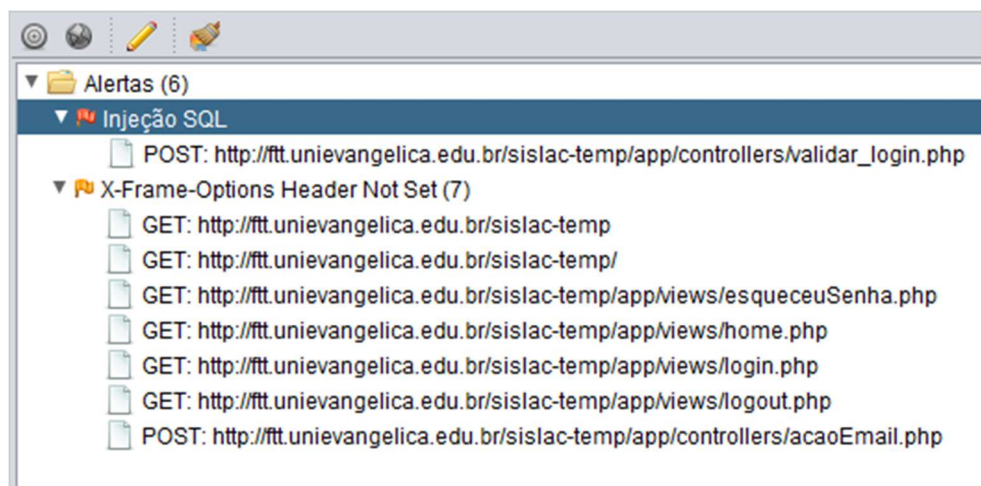
As informações identificadas do servidor, podem ser usadas para explorar uma vulnerabilidade já conhecida, em decorrer dos *patches* de atualizações, que expôs as correções das falhas de segurança. O escopo deste trabalho não aborda as principais vulnerabilidades em servidores *web*. Entretanto, a OWASP (2017), sugere, sempre que possível, manter as versões de sistemas operacional, serviços, bibliotecas e frameworks, na versão mais atual.

3.4.2 RESULTADO DO TESTE DA AP2

A AP2, em 2017, passou por um teste de penetração e teve suas vulnerabilidades expostas para que o proprietário de tal sistema tivesse a oportunidade de fazer as devidas correções (MARTINS; ALVES, 2017). O objetivo de repetir os testes no mesmo sistema é validar se as correções foram efetuadas e se certificar de que não existem mais nenhuma nova falha de segurança.

Para isso, a aplicação foi disponibilizada em um servidor de teste, pois executar tais testes em um ambiente de produção poderia comprometer a disponibilidade dos serviços. Vale ressaltar, que a aplicação em produção é exatamente a mesma versão disponibilizada no ambiente de teste.

Figura 10 - Vulnerabilidades da AP2



Fonte: Captura de tela da ferramenta OWASP ZAP.

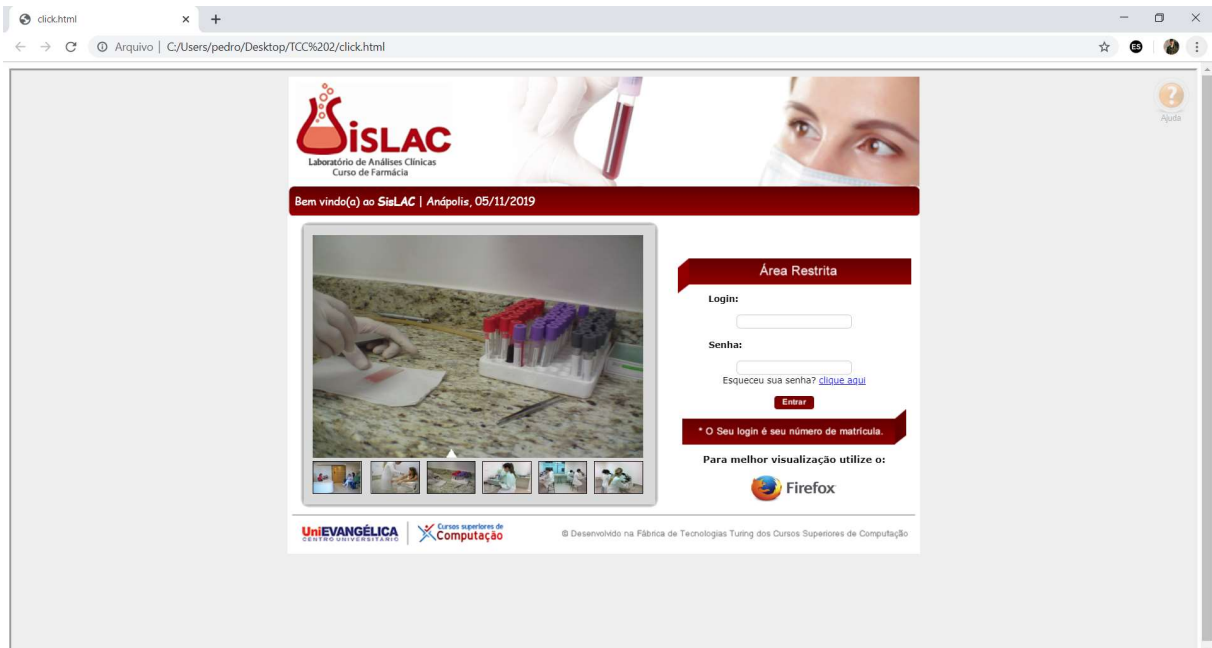
Na figura 10 são expostas as falhas de segurança identificadas no sistema em questão pela ferramenta OWASP ZAP. Foram encontrados 2 tipos de vulnerabilidades. A seguir, serão detalhadas cada uma das falhas de segurança encontradas, em conformidade com as métricas propostas.

M1. Requisições de terceiros

A AP2 não permite que origens desconhecidas façam requisições à aplicação, para isso o CORS está configurado corretamente, dessa forma pode-se evitar a existência de diversas vulnerabilidades, tais como, obtenção de dados sensíveis.

A única vulnerabilidade identificada para a métrica M1 foi a *X-Frame-Options Header Not Set*, categorizada com risco médio e diz respeito a possibilidade, do agente malicioso conseguir incorporar o conteúdo da aplicação em outras páginas, através de frames, como mostra a Figura 11, com o objetivo de explorar diversas outras vulnerabilidades, como por exemplo, fazer com que o usuário do sistema tente se autenticar, para que assim o atacante consiga obter de forma fácil o acesso ao sistema.

Figure 11 - X-Frame-Option Header Not Set da AP2



Fonte: Captura de tela Google Chrome.

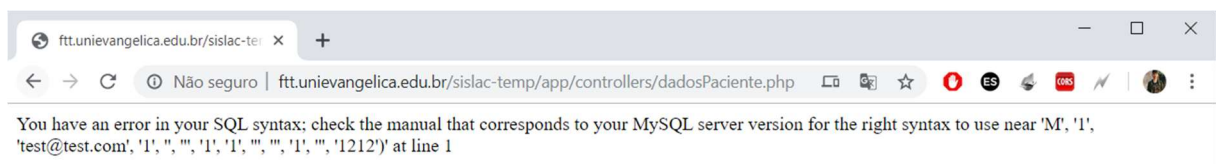
Com prevenção dessa vulnerabilidade é necessário configurar o *X-Frame-Options*, que instruem os navegadores a bloquearem o uso de frames por domínios desconhecidos.

M2. Dados de entrada do usuário são válidos

Uma das vulnerabilidades encontradas na AP2, diz respeito a falta do tratamento de dados inseridos pelo usuário, dessa forma possibilitando a existência de vulnerabilidades com SQL INJECTION.

A figura 12 mostra um exemplo da vulnerabilidade citada, na qual o sistema pega os dados inseridos pelo usuário e concatena para formar uma *query* que posteriormente será executada no banco.

Figura 12 - Dados sensíveis expostos da AP2.



Fonte: Captura de tela Google Chrome.

Neste exemplo foram inseridos vários caracteres de aspas simples nos campos do formulário de paciente, na qual conseqüentemente provocaria um erro na execução da *query*, resultando em um erro de sintaxe. Como podemos observar na Figura 12, o erro ocasionado também não é tratado pelo sistema, que conseqüentemente acaba retornando o problema de sintaxe para o usuário e disponibiliza a informação de qual banco de dados a aplicação está utilizando.

Tal falha poderia ser evitada utilizando como técnica de desenvolvido um ORM, que mapeia o banco de dados e disponibiliza uma interface para execução de consultas, dessa forma evitando que os dados inseridos pelo usuário fossem concatenados na *query*.

M3. Dados sensíveis obtidos por usuários não autorizados

Uma das falhas encontradas na AP2, diz respeito a falta de confidencialidade com a informação, sendo um dos principais pilares da segurança da informação e está relacionada com quais usuários poderiam ter acesso a determinados dados.

A AP2 não utiliza algoritmos que encriptam os dados das requisições e também não está hospedado no https, protocolo esse que encripta os dados, fazendo com que mesmo que os agentes maliciosos consigam interceptar a informação, eles não serão capaz de modificar ou interpretar.

A AP2 também possui uma falha na programação permitindo que qualquer usuário consiga obter informações importantes do servidor, pois a aplicação não trata as urls digitadas pelo usuário e acaba retornando um erro 404 (NOT FOUND do protocolo http) juntamente com informações vitais do servidor. Pode-se observar na Figura 13 a falha citada.

Figura 13 - Falha de programação da AP2



Fonte: Captura de tela Google Chrome.

M4. Proteção da API por meio de *tokens* de acesso

A AP2 só atende as requisições de usuários autenticados, visto que, a aplicação redireciona o usuário sem permissão para a tela de login, quando o mesmo faz solicitações de recursos à aplicação sem estar autenticado.

Sendo assim, não foram encontradas falhas de segurança no que tange a obtenção de dados sem autenticação. Portanto, a AP2, atende a métrica M4.

M5. Tempo diário que o sistema fica disponível para usuários autorizados

Tal métrica diz respeito à disponibilidade do sistema, sendo um dos pilares da segurança da informação, para isso, durante todo o período de teste o sistema se manteve disponível e estável, toda e qualquer requisição feita ao mesmo durante o período de teste foi atendida em tempo hábil. Sendo assim, a AP2 atende a métrica M5.

A ferramenta NMAP não pôde ser executada na AP2, pois as configurações de rede não permitem que terceiros façam solicitações de *ping* nos servidores da aplicação. Também foram realizadas tentativas acessando da rede da instituição, no entanto, foi constatado que as configurações também bloqueiam as solicitações internas. Visto isso, não foi possível identificar informações vitais dos servidores através da ferramenta NMAP.

3.4.3 COMPARATIVO DOS RESULTADOS

A seguir, veremos uma tabela comparativa entre as duas aplicações utilizadas no estudo de caso.

Tabela 1 – Comparação dos resultados da medição.

Métricas	AP1	AP2
M1. Requisições de terceiros são bloqueadas?	Não Atende	Não Atende
M2. Dados de entrada do usuário são válidos?	Atende	Não Atende
M3. Dados sensíveis podem ser obtidos por agentes não autorizados?	Não Atende	Não Atende
M4. Proteção da API por meio de <i>tokens</i> de acesso?	Atende	Atende

M5. Disponibilidade para usuários autorizados	Atende	Atende
---	--------	--------

A AP1, atendeu as métricas M2, M4 e M5, e não atende as métricas M1 e M3. Para isso, se faz necessário correções corretivas, tais como, configurar o CORS, hospedar a aplicação no protocolo HTTPS, usar algoritmos de encriptação de dados para requisições e configurar o cabeçalho *X-Frame-Options*.

A AP2, atendeu as métricas M4 e M5. E não atendeu as métricas M1, M2 e M3. Para que as falhas sejam corrigidas, é necessário que a aplicação passe por um processo de refatoração do código, visto que, não se faz o uso de ORM. Com isso, as entradas dos usuários não são validadas, possibilitando a existência de diversas vulnerabilidades. Outras questões que devem ser corrigidas, são, configuração do cabeçalho *X-Frame-Options* e utilização de algoritmos de encriptação de dados, visto que, a aplicação não está hospedada no protocolo HTTPS.

Após a finalização dos testes, fica evidente que ambas as aplicações possuem falhas de segurança que devem ser corrigidas, para que possíveis danos sejam evitados. Vale ressaltar, que com a utilização de um processo de gestão da segurança da informação durante o desenvolvimento do produto de software, muitas dessas falhas de segurança poderiam ter sido corrigidas antes do sistema ser disponibilizado no ambiente de produção, evitando transtornos e possíveis prejuízos.

4. CONSIDERAÇÕES FINAIS/CONCLUSÃO

Este trabalho propôs um estudo no contexto da segurança da informação, abordando a análise de um processo de gestão da segurança da informação, buscando agregar melhorias a este processo, no que diz respeito a elaboração de métricas de software.

A partir do uso da metodologia *GQM* foi possível obter métricas capazes de avaliar o nível de segurança de uma aplicação *web* e nortear possíveis correções aos proprietários desses sistemas.

Para a realização do estudo de caso, foram utilizadas ferramentas automatizadas, tais como OWASP ZAP e NMAP. Por meio delas, foi possível aplicar as métricas propostas em dois sistemas *web*. Os resultados do estudo de caso mostram que ambas as aplicações possuem falhas de segurança e, portanto, necessitam de correções.

Neste contexto, se pode relatar contribuições nos âmbitos tecnológicos, socioeconômicos e educacionais.

Quanto ao contexto tecnológico, têm-se um processo de gestão da segurança da informação melhorado, sistematizado, que inclui mecanismos formais de medição da segurança, a partir da adição de métricas de software identificadas com o uso da metodologia *GQM*. Que, no contexto apresentado, pode ser replicado por profissionais de segurança da informação.

Sobre a contribuição socioeconômica, esse trabalho trouxe para o cenário da segurança da informação, na qual acontecem inúmeros incidentes, a possibilidade que os proprietários de aplicações *web* fiquem cientes do nível de segurança que seus sistemas possuem e das possíveis vulnerabilidades existentes, para que tomem medidas preventivas ou corretivas.

Proporcionando maior rentabilidade as empresas e um sistema mais seguro para o usuário final, uma vez que os possíveis prejuízos ocasionados pela falta de segurança sejam na sua maioria identificados e solucionados.

Em termos de resultados educacionais este trabalho contribui com o meu conhecimento e formação acadêmica e também com os de novos profissionais, pois, expõe de forma objetiva as principais vulnerabilidades que as aplicações *web* podem possuir, proporcionando um diferencial para o profissional e as empresas que se preocupam com a segurança dos dados.

Analisando os resultados do estudo de caso, percebe-se uma possibilidade para a realização de trabalhos futuros, que se trata da identificação de métricas específicas para vulnerabilidades em servidores, considerando que este aspecto não inclui o escopo definido para esta pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ABNT ISO/IEC 17799. **Código de prática para a gestão da segurança da informação**. Rio de Janeiro, 2005.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126-1. **Engenharia de software - Qualidade de produto**. Rio de Janeiro, 2003.

BATISTA, C. F. A. **Métricas de Segurança de Software**. Disponível em: <https://www.maxwell.vrac.puc-rio.br/10990/10990_4.PDF>. Rio de Janeiro, 2007. Acesso em: 10 de Maio. 2019.

CERT. Centro de Estudos Respostas e Tratamento de Incidentes de Segurança No Brasil. **Total de Incidentes Reportados ao CERT.br por Ano**. Disponível em: <<https://www.cert.br/stats/incidentes/>>. Acesso em: 04 Abril. 2019.

GIL, A. C. **Métodos E Técnicas De Pesquisa Social**. 6.ed. São Paulo: Atlas, 2008.

MARTINS, Jean; ALVES, Vanessa. **Processo de gestão da segurança da informação: Definição e aplicação em um sistema laboratorial**. Monografia do curso de engenharia de computação do centro universitário de Anápolis, 2017.

MOZILLA. **Cross-Origin Resource Sharing (CORS)**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle_Acesso_CORS>. Acesso em: 30 de Outubro de 2019.

OWASP. Open Web Application Security Project. **OWASP Top 10 – 2017: The ten most critical web application security risks**. Disponível em: <https://www.owasp.org/images/0/06/OWASP_Top_10-2017-pt_pt.pdf>. Acesso em: 28 Abril, 2019.

PRESSMAN, Roger. **Engenharia de Software: uma abordagem profissional**. 7ª Edição. Porto Alegre: AMGH, 2011.

PRESSMAN, Roger. **Engenharia de Software: uma abordagem profissional**. 8ª Edição. Porto Alegre: AMGH, 2016.

MONTEVERDE, Wagner A. **Estudo e Análise de Vulnerabilidades Web**. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/5823/1/CM_COINT_2013_2_02.pdf>. Acesso em: 13 Dezembro, 2019.

SANS. Security 519 – **web application workshop**. SANS INSTITUTE, 2008a.

SANS. Security 542 – **advanced web application penetration testing**. SANS INSTITUTE, 2008b.

SOLOMON, Michael; KIM, David. **Fundamentos de segurança de sistemas de informação**. Rio de Janeiro: LTC, 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 9° Edição. São Paulo: Person Prentice Hall, 2011.

STALLINGS, William. **Criptografia e segurança de redes**. 4° Edição. São Paulo: Prentice Hall Brasil, 2008.

STUTTARD, D. PINTO, M. **The Web Application Hacker's Handbook**. Wiley Publishing, Inc, 2007.

UFPE. **GOAL – QUESTION – METRIC**. Disponível em: <http://www.cin.ufpe.br/~scbs/metricas/seminarios/GQM_texto.pdf>. Acesso em: 08 Março. 2019.

UTO, Nelson; MELO, Sandro Pereira de. **Vulnerabilidades em Aplicações Web e Mecanismos de Proteção**. 2015. Disponível em: <https://www.researchgate.net/profile/Sandro_Melo/publication/266445537_Capitulo_6_Vulnerabilidades_em_Aplicacoes_Web_e_Mecanismos_de_Protecao/links/55b0e3c708ae092e964fb3d5.pdf>. Acesso em: 13 Dezembro, 2019.

VIJAYARANI, S.; TAMILARASI, A. An efficient masking technique for sensitive data protection. In: Recent Trends in Information Technology (ICRTIT), 2011 International Conference on. [S.l.: s.n.], 2011. p. 1245–1249.

ZAP, OWASP. **Projeto OWASP Zed Attack Proxy**. Disponível em: <https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project#tab=Main>. Acesso em: 28 Abril. 2019.