Um Levantamento sobre os Aspectos Técnicos dos Principais Riscos de Segurança e Ataques em Aplicações Web

João Vitor Batista do Nascimento[1], Maurício Moreira Neto[2,3], Emanuel F. Coutinho[4], Leonardo O. Moreira[1]

[1]Instituto Universidade Virtual (IUVI)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[3]Centro Universitário Christus (UNICHRISTUS) – Fortaleza, CE – Brasil

[4]Programa de Pós-Graduação em Computação (PCOMP)
Universidade Federal do Ceará (UFC) – Quixadá, CE – Brasil

joao.vitoor.batista@gmail.com, maumneto@alu.ufc.br,
emanuel.coutinho@ufc.br, leoomoreira@virtual.ufc.br

Abstract. *Many applications are being implemented in web environments to achieve some features like wide access and availability. However, such characteristics increase the exposure of applications, as it is a less controlled environment when compared to traditional or centralized applications, increasing the chances of eventual attacks and security risks. This paper aims to make a survey containing the technical aspects of the main security risks and attacks on web applications, discuss them and analyze technical solutions to reduce or eliminate the risks and attacks. Finally, it was possible to discuss solutions that can reduce or eliminate the risks and attacks listed in this work.*

Resumo. *Muitas aplicações estão sendo implementadas em ambientes web para alcançar algumas características como o amplo acesso e disponibilidade. No entanto, tais características aumentam a exposição das aplicações, pois é um ambiente menos controlado quando comparado a aplicações tradicionais ou centralizadas, aumentando as chances de ocorrerem eventuais ataques e riscos à segurança. Este artigo tem como objetivo fazer um levantamento contendo os aspectos técnicos sobre os principais riscos de segurança e ataques em aplicações web, discuti-los e analisar soluções técnicas para diminuir ou eliminar os riscos e ataques. Por fim, foi possível discutir soluções que possam reduzir ou eliminar os riscos e ataques elencados neste trabalho.*

1. Introdução

O número de aplicativos desenvolvidos no contexto de ambientes web, para uso pessoal e comercial, continuará a aumentar [Jahromi et al. 2020]. A popularidade das aplicações web cresceu, principalmente nestes tempos de pandemia, onde serviços de comércio, alimentação, educação e outros segmentos tiveram que se servir de ambientes web no intuito de manterem seus serviços em operação mesmo diante das restrições e do isolamento social. Com isso, muitos segmentos conseguiram fazer com que suas empresas,

instituições e pequenos empreendimentos não fossem totalmente interrompidos, diminuindo desempregos e o fechamento dos empreendimentos.

De uma forma resumida, uma aplicação web é hospedada em um servidor para que esteja disponível, aos usuários, por meio dos navegadores web [Conallen 2003]. Um processo servidor, normalmente um servidor de aplicação web, faz com que as aplicações web fiquem disponíveis, onde cada programa implementado nas aplicações web seja invocado por meio de uma *Uniform Resource Locator* (URL). Além disso, muitas aplicações web fazem uso de Bancos de Dados como estratégia de armazenamento e recuperação de dados ou informações. A popularidade crescente dos aplicativos da web os torna um alvo atraente para usuários mal-intencionados. Grandes volumes de dados privados comumente processados e armazenados por aplicações web são recursos valiosos para os invasores, resultando em ataques, orientados para a web, mais sofisticados [Prokhorenko et al. 2016].

Diante disso, podem-se destacar pelo menos duas características preocupantes do ponto de vista dos riscos e segurança em aplicações web: i) as características de amplo acesso e disponibilidade aos usuários na web; e ii) dependendo do modo que foi implementada a solução de gestão de dados, pode-se comprometer a privacidade dos dados que são gerenciados, produzidos e processados. Assim, surge a necessidade de entender como os riscos e ataques ocorrem, em aplicações web, para que se possa implementar soluções que mitiguem tais situações. Portanto, a principal contribuição deste trabalho é fazer um levantamento exploratório dos principais riscos e ataques em aplicações web, discuti-los e analisar soluções técnicas no intuito de diminuir ou eliminar os riscos e ataques.

Gil (2002) comenta que algumas regras práticas para formulação de problemas científicos são: i) deve ser elaborada como uma pergunta; ii) deve ser o mais detalhado possível; e iii) utilizar termos claros e sem ambiguidades. Assim, pode-se desenvolver o problema científico, que norteia este trabalho, com a seguinte questão: "Como ocorrem os principais ataques em aplicações web?". Posterior a indagação, cria-se a seguinte hipótese: "A maioria dos ataques elencados podem ser mitigados por meio de implementações na aplicação web".

O objetivo principal deste ensaio é apresentar um levantamento contendo os aspectos técnicos sobre os principais riscos e ataques a aplicações web no intuito de mitigar ou eliminar estes riscos e ataques. Para cumprir o objetivo principal supracitado, os seguintes objetivos secundários foram especificados: i) apresentar os conceitos teóricos pertinentes ao desenvolvimento de aplicações web; ii) elencar os principais ataques a aplicações web relatados na literatura; e iii) realizar uma discussão e comentar sobre como os ataques ocorrem e quais técnicas podem atenuá-los.

Este trabalho está dividido, além da introdução, nas seguintes seções: a Seção 2 apresenta todo arcabouço teórico necessário para o entendimento, compreendendo os conceitos web, tecnologias web, protocolo de comunicação *HyperText Transfer Protocol* (HTTP), arquitetura cliente-servidor, recursos estáticos, geração de recursos dinâmicos, características das aplicações web e as vulnerabilidades descritas no documento *Open Web Application Security Project* (OWASP) Top Ten. Já a Seção 3 descreve os aspectos metodológicos adotados no trabalho. A Seção 4, por sua vez, comenta sobre os resultados obtidos e apresenta a discussão, enfatizando como os ataques e os riscos em aplicações

web podem ser mitigados ou eliminados. Por fim, a Seção 5 apresenta as conclusões e comenta sobre trabalhos futuros.

2. Fundamentação Teórica

2.1. Tecnologias Web e Arquitetura Cliente-Servidor

Na web, praticamente, todas as informações são obtidas por meio de documentos [Tanenbaum e Van Steen 2007]. Um documento na Web pode possuir conteúdos textuais e também conteúdos midiáticos como áudio, vídeo, animações, imagens, arquivos binários etc. Segundo Coulouris et al. (2013) a web tem evoluído sem mudar sua arquitetura básica que é baseada em três componentes tecnológicos padrão principais: i) *HyperText Markup Language* (HTML) que é uma linguagem de marcação utilizada para especificar o conteúdo e a estrutura das páginas de forma que elas possam ser interpretadas pelos navegadores web; ii) URLs que é uma forma de identificar os documentos e outros recursos disponibilizados na web; e iii) arquitetura cliente-servidor [Kurose e Ross 2010] com regras padrão para interação, por meio do protocolo HTTP, permitindo que navegadores e outros clientes possam obter documentos e diversos recursos disponíveis em servidores web.

Em uma típica instanciação de uma arquitetura cliente-servidor na web, o cliente é um navegador web e o servidor é um processo de execução contínua aguardando solicitação dos clientes. É válido ressaltar que a arquitetura cliente-servidor é um modelo lógico, onde quem faz solicitação de algo é caracterizado como processo cliente e quem responde a uma solicitação é caracterizado como processo servidor [Kurose e Ross 2010]. As aplicações web são diferentes dos web sites mais tradicionais, pois seu objetivo é permitir que os usuários realizem tarefas mais complexas como fazer reservas de viagens, encontrar casas, pagamento de contas, transferir dinheiro, comprar produtos etc. Já os web sites tradicionais são projetados para consumo de informações estáticas [Vora 2009].

A base da web é a transferência de páginas do servidor para o cliente por meio do protocolo HTTP [Tanenbaum e Van Steen 2007]. As páginas estáticas são arquivos mais simples mantidos no servidor com conteúdo fixo, ou seja, textos e mídias que não se alteram com requisições [Souza 2018]. Da mesma forma que as páginas estáticas, os servidores também possuem outros objetos ou recursos estáticos, tais como imagens, áudios, documentos de representação de dados etc. Estes objetos e recursos estáticos são caracterizados por não se modificarem, em tempo de execução, diante de uma requisição específica ou passagem de parâmetros.

Já as páginas, objetos ou recursos dinâmicos são caracterizados por terem seus conteúdos gerados, em tempo de execução, por meio das linguagens de programação [Souza 2018]. Além disso, eles possuem capacidades de receberem parâmetros e também se personalizarem conforme uma determinada requisição de um cliente. Assim, aplicações baseadas na web têm, em boa parte da sua implementação, a adoção de páginas, objetos e recursos dinâmicos. Em resumo, aplicações web possuem programas, escritos em alguma linguagem de programação, que são invocados por meio de URLs, podendo passar parâmetros e escolher o método HTTP usado para interagir com o programa.

Muitas aplicações são orientadas a dados, tais como comércio eletrônico, educação, pesquisas científicas, grandes corporações, instituições, alimentação, saúde etc.

Assim, na infraestrutura de uma arquitetura cliente-servidor, neste contexto, deve possuir um componente servidor responsável por fazer a gestão de dados. Dependendo do número de usuários ou carga de trabalho das aplicações web, um grande volume de dados pode ser coletado, tratado, processado e analisado. Assim, o servidor que faz a gestão de dados é um componente crítico no desempenho das aplicações, pois o maior tempo gasto no processamento das requisições de aplicações está relacionada à solução de gestão de dados [Moreira 2014].

Diante do que foi exposto, pode-se perceber alguns riscos que culminem em ataques que são capazes de acontecer por meio de comunicação, pois aplicações web possuem aspectos de amplo acesso e disponibilidade. Os próprios programas disponíveis para geração de conteúdo dinâmico podem ter erros de programação por não prever alguns tipos de ataques até por desconhecimento, o que valoriza a discussão neste presente trabalho. Outro aspecto que pode ser comentado é que as aplicações web fazem gestão de dados, então ataques neste contexto podem impactar na integridade e privacidade destas informações.

2.2. Vulnerabilidades em Aplicações Web

A fundação OWASP [OWASP 2021a] mantém um documento, chamado OWASP Top 10 [OWASP 2021b], que promove uma conscientização para o aumento da segurança em aplicações web. Ao adotar o OWASP Top 10 é talvez o primeiro passo mais eficaz para produção de um código mais seguro, reduzindo os riscos que culminem em ataques às aplicações web [OWASP 2021b]. A seguir detalha-se os mais críticos riscos e ataques a aplicações web segundo o documento OWASP Top 10.

2.2.1. Injeção

Ataques de injeção como injeções de *Structured Query Language* (SQL), Sistema Operacional (SO) ou *Lightweight Directory Access Protocol* (LDAP) ocorrem quando o atacante se aproveita de falhas dos programas que fazem interações com Bancos de Dados através de comandos SQL, conseguindo obter, inserir, alterar e remover dados indevidamente em Banco de Dados servindo-se dos campos de entrada de dados como formulários ou URLs de aplicação web. Segundo OWASP (2021) [OWASP 2021b], o ataque por injeção pode impactar na perda ou confiabilidade dos dados, além de negação de acesso e podendo até perder o controle total do sistema como um todo.

2.2.2. Quebra de Autenticação

As funções de aplicações relacionadas à autenticação e gerenciamento de sessão são frequentemente implementadas incorretamente, permitindo que os invasores comprometam senhas, chaves ou *tokens* de sessão ou explorem outras falhas de implementação para assumir as identidades de outros usuários temporária ou permanentemente [OWASP 2021b]. Já os ataques à gestão de sessão acontecem, normalmente, de modo que os *tokens* particulares não expiram. A quebra de autenticação pode ser feita, manualmente ou automaticamente, pelos atacantes, servindo-se de diversas técnicas, como a força bruta para revelar a identificação. Esses ataques podem ocorrer contra aplicações, e colocar em risco

dados sensíveis, que facilitam aos atacantes executarem ações como fraudes, roubos de identidades, lavagem de dinheiro, dentre outros.

2.2.3. Exposição de Dados Sensíveis

Esse tipo de ataque tende a acontecer manualmente e exige uma maior atenção humana por parte do atacante. Os atacantes conseguem previamente as senhas das vítimas ou se utilizam de ataques do tipo *man-in-the-middle* (MitM), quando a troca de informações entre duas partes é interceptada por um terceiro, os dados são roubados e enviados ao servidor ou ao cliente. As senhas podem ser obtidas, previamente, por meio da força bruta, quando um algoritmo tenta exaustivamente várias combinações de senhas até encontrar a verdadeira, utilizando processadores gráficos (GPUs). Os impactos causados por esta falha comprometem os dados que deveriam estar protegidos, geralmente informações pessoais sensíveis como registros médicos, credenciais, dados pessoais e de cartões de crédito, por exemplo.

2.2.4. Entidades Externas de XML (XXE)

Neste tipo de ataque, os atacantes abusam de processadores *eXtensible Markup Language* (XML) vulneráveis, processadores antigos ou mal configurados. Caso consigam carregar XML ou incluir conteúdo malicioso em um documento XML, pode ser acometido por um abuso do código vulnerável, dependências ou integrações. O atacante, conhecendo essas falhas, pode executar códigos remotamente e compartilhar arquivos internos da aplicação. Caso o analisador XML não bloqueie o acesso de entidades externas, esse ataque pode impactar na proteção de aplicações e dos dados destas, pois o atacante pode obter credenciais de outros usuários, pondo em risco a integridade das informações, e podendo gerar ataques de negação de serviço [Pacheco 2019].

2.2.5. Quebra de Controle de Acesso

Este tipo de falha ocorre devido à falta de automatização de processos que auxiliem na detecção destas. A falta de testes funcionais, que devem ser realizados pelos programadores, também pode causar essa falha de segurança. Os atacantes se aproveitam da não ou má implementação de funcionalidades que protejam a aplicação de tentativas de acessos não autorizados [Corrêa e Gonçalves 2019], bem como a falta de claridade na hierarquia de cargos, que diz respeito a delegar a cada tipo de usuário seus privilégios dentro do sistema, impedindo que um usuário comum tenha acesso a áreas que somente um administrador deva ter. Essa vulnerabilidade pode fazer com que o *hacker* possa atuar com o mais alto privilégio no sistema, podendo alterar ou excluir dados sensíveis, desviar informações e até mesmo comprometer a segurança do servidor ou sistema operacional em que a aplicação está hospedada [Corrêa e Gonçalves 2019].

2.2.6. Configurações de Segurança Incorretas

Sobre esse ataque em específico, os atacantes buscam acessar às contas padrão, páginas sem utilidade, falhas sem correção, ficheiros e diretórios desprotegidos, etc., conseguem ganhar acesso não autorizado explorando a padronização nas credenciais do sistema ou configurações incorretas nos servidores ou *frameworks* da aplicação [Pereira e Esmeraldo 2019]. As falhas, por exemplo, costumam conceder aos atacantes acesso privilegiado a alguns dados ou funcionalidades da aplicação. Frequentemente, tais falhas fazem com que a aplicação seja comprometida por completo, ocasionando danos irreparáveis como a perda total de dados.

2.2.7. Cross-Site Scripting (XSS)

Esse tipo de ataque acontece aos navegadores das vítimas. Existem 3 tipos de XSS: o Reflected XSS, que a aplicação web ou *Application Programming Interface* (API) têm os dados dos usuários como parte da resposta HTML sem que esses tenham passado por uma validação ou método de tratamento de caracteres especiais, conhecidos como *escaping*. Normalmente, o utilizador clica em algum *link* malicioso que o leva a uma página web dominada pelo atacante; o Stored XSS, que a aplicação ou API armazenam os dados dos usuários sem tratá-los, sendo utilizados posteriormente por outros usuários ou administradores do sistema; e o *Document Object Model* (DOM) XSS que é um ataque que executa todos os códigos JavaScript maliciosos diretamente no navegador da vítima, sem ter contato com o servidor. Os impactos causados pelo Reflected XSS e pelo DOM XSS são considerados leves ou moderados, já o Stored XSS é considerado muito perigoso, pois o código malicioso fica armazenado na página web infectando todos os demais usuários que acessam o *link* prejudicado, roubando informações pessoais dos usuários.

2.2.8. Desserialização Insegura

Nesse caso, o abuso da desserialização é difícil, mas não impossível de acontecer. Ocorre quando a aplicação desserializa, ou seja, desconverte os dados de formato binário para o formato da linguagem de programação em que a aplicação foi construída, dados implantados ou modificados por atacantes, alterando a lógica do funcionamento do software ou realizando a execução remota de códigos [Reis 2018]. O impacto das falhas de desserialização, podem levar a ataques de negação de serviços, ataques de repetição, ataques de injeção e ataques de escalonamento de privilégios, por exemplo.

2.2.9. Utilização de Componentes Vulneráveis

Os componentes vulneráveis são conhecidos, documentados e divulgados a fim de que sejam evitados o seu uso na aplicação ou para que sejam atualizados no intuito de resolverem tal vulnerabilidade [Mocelin et al. 2018]. A insistência na utilização destes componentes faz crescer, consideravelmente, o risco de um ataque à aplicação, tendo em vista do conhecimento dessas falhas de segurança fica mais fácil a sua exploração por meio de instruções específicas ou ferramentas automatizadas [Mocelin et al. 2018]. Mui-

tos desses componentes são executados por usuários com privilégios de administrador do sistema. Se o criminoso conseguir esse importante acesso, através da exploração de um componente vulnerável, com altas capacidades dentro da aplicação, pode obter facilmente acesso a informações confidenciais no sistema.

2.2.10. Registro e Monitorização Insuficiente

O abuso do registo e monitorização insuficiente são a base de quase todos os incidentes mais importantes. Por falta de monitorização e capacidade de resposta, os atacantes têm tempo disponível para conseguir atingir os seus objetivos sem serem detectados. A partir da identificação automática de vulnerabilidades, é possível chegar a um ataque bem sucedido. Em 2016, a identificação de uma falha levava em média cerca de 191 dias, o necessário para que ocorra algum dano. Os impactos causados podem ser desde a utilização de um sistema para atacar outras aplicações ou obter acesso a dados confidenciais, podendo alterá-los ou excluí-los [Maciel 2019] [OWASP 2021b].

3. Metodologia

Um objetivo de pesquisa, normalmente, comporta uma hipótese de trabalho. Uma boa definição de um objetivo de pesquisa, geralmente, visa uma forma de apresentar que a hipótese elaborada é verdadeira [Wazlawick 2009]. Para atingir o objetivo nesta pesquisa, servindo-se da metodologia da pesquisa científica, utilizou-se as etapas discutidas por Wazlawick (2009): i) elaborar um tema de pesquisa que determina uma área de conhecimento na qual se deseja trabalhar; ii) realizar uma revisão bibliográfica; e iii) definir o objetivo da pesquisa. Esta primeira etapa da metodologia foi de suma importância na identificação da relevância do tema que permeia este ensaio. Além disso, também ajudou no delineamento do escopo da pesquisa por meio das hipóteses e dos objetivos, geral e específicos.

A pesquisa, em relação aos aspectos de caracterização e natureza, se fundamenta na pesquisa exploratória e não-experimental. Pesquisas exploratórias têm como objetivo principal o aprofundamento de ideias ou a descoberta de intuições [Gil 2002]. Segundo Gil (2002) grande parte das pesquisas exploratórias envolvem: i) levantamento bibliográfico; ii) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; e iii) análise de exemplos que estimulem a compreensão. Nesta presente pesquisa, a natureza da pesquisa exploratória foi utilizada para apresentar os conceitos básicos necessários para que se possa entender o problema e o objetivo da pesquisa, além de descrever, com maiores detalhes, o que são aplicações web, elementos das infraestruturas de aplicações web, tipos de conteúdo, estrutura do funcionamento das aplicações web, ataques e riscos à segurança em aplicações web elencados e os motivos das suas escolhas. A maior parte da utilização da pesquisa exploratória, neste trabalho, foi por meio do levantamento bibliográfico e entrevistas com desenvolvedores web que já tiveram contato com ataques e riscos à segurança em aplicações web. Portanto, a segunda etapa da metodologia se serviu da pesquisa exploratória para reunir todo arcabouço teórico e os conhecimentos necessários para o entendimento do presente estudo.

Por ser também uma pesquisa não-experimental são aplicadas observações com o objetivo de tirar conclusões a partir de um conjunto teórico reunido

[Wazlawick 2009]. Após isso, será detalhado o método de pesquisa, onde é especificada uma sequência de etapas necessárias para apresentar que o objetivo proposto foi alcançado [Wazlawick 2009]. Neste trabalho, a pesquisa de natureza não-experimental consiste no estudo sem a intervenção sistemática do pesquisador, observando e tirando conclusões a partir de um conjunto teórico levantado [Wazlawick 2009]. Portanto, a pesquisa não-experimental compreende as etapas da metodologia que refletem uma discussão sobre os aspectos técnicos de cada risco e ataque elencado durante a etapa exploratória, além de propostas soluções técnicas que possam reduzir ou eliminar os riscos e ataques que foram identificados no contexto de aplicações web. Nesta terceira e última etapa da pesquisa, utilizou-se a experiência prévia dos autores na área de desenvolvimento web.

4. Resultados e Discussão

Diante da pesquisa de natureza exploratória realizada na Seção 2 é possível apresentar os resultados obtidos por meio do levantamento bibliográfico e realizar uma discussão de soluções técnicas que possam mitigar ou eliminar os riscos e ataques que foram identificados no contexto de aplicações web, utilizando a experiência prévia dos autores que possuem conhecimentos de desenvolvimento web. Durante a etapa exploratória da metodologia especificada neste ensaio, foi encontrado um documento, chamado de OWASP Top 10 [OWASP 2021b], que fez a descrição de dez riscos e ataques a aplicações web, visando à implementação de códigos mais seguros, reduzindo os riscos que culminem em ataques. Por meio dos instrumentos de pesquisa não-experimental, é realizada uma discussão técnica sobre cada risco ou ataque elencado na etapa exploratória, identificando como os aspectos de implementação, codificação ou uso de tecnologias podem reduzir os riscos e ataques.

Os ataques por injeção de SQL ocorrem, popularmente, quando o atacante consegue concatenar parâmetros maliciosos, não previstos pelo programador, em consultas SQL embutidas no código do programa de aplicação e alcança o nível de execução no Sistema Gerenciador de Banco de Dados (SGBD). Portanto, é muito importante que o programador faça uma validação prévia dos tipos de dados e padrão (expressões regulares podem ser úteis) dos parâmetros recebidos antes da operação de concatenação dos parâmetros na consulta SQL, observando se existem caracteres de comentários SQL, uso de aspas, tentativa de inclusão dos separadores de instruções SQL, condições lógicas, elaboração de novas instruções SQL etc.

Outro fator importante é observar o nível de permissão do usuário criado pelo SGBD para que o programador possa fazer a conexão da aplicação web com o Banco de Dados. É válido ressaltar que um SGBD pode gerenciar Bancos de Dados de aplicações distintas. Assim, um atacante pode ter acesso aos dados de outra aplicação, pois uma aplicação vulnerável, que compartilha o mesmo SGBD, permitiu acesso aos dados de outra aplicação pelo fato do usuário ter um alto nível de permissão. Dependendo do nível experiência do atacante e também do nível de permissão do usuário no SGBD da aplicação vulnerável, os ataques podem ser propagados para o nível de Sistema Operacional (SO), pois muitos SGBDs possuem funções de interação com o SO do servidor, assim tornando este ataque mais perigoso. Ataques de injeção LDAP ocorrem, basicamente, quando as entradas de dados não sofrem um bom tratamento e validação, sendo possível modificar as instruções LDAP por meio de técnicas semelhantes à injeção SQL.

Já os ataques que visam a quebra de autenticação, exigem cuidados para que os atacantes não se passem por usuários. Como um método de prevenção, seria interessante implementar a autenticação em multifator, para evitar ataques automatizados, de preenchimento de credenciais, de força bruta e de reutilização de credenciais roubadas. Verificações de senhas fracas também podem mitigar esse tipo de ataque, uma vez que senhas fortes são mais difíceis de serem reveladas diante destes ataques. Observar a periodicidade e quantidade de ações de autenticação mal sucedidas, assim identificando um possível ataque de quebra de autenticação por diversas tentativas. Por fim, uma outra situação que pode ser implementada para reduzir o ataque é gerenciar os identificadores de sessão, pois os identificadores de sessão não devem estar no URL, devem ser armazenadas com segurança e invalidadas diante de ações de *logout*, ociosidade e *timeout*.

A principal técnica que pode ser adotada para mitigar ataques que permeiam a exposição de dados sensíveis é classificar quais dados são confidenciais de acordo com as leis de privacidade, requisitos regulatórios ou regras de negócios. Com isso, aplicando uma técnica mais robusta de proteção aos dados sensíveis, como anonimização, ofuscação, criptografia forte, distribuição ou embaralhamento dos dados. Uma outra solução técnica é evitar o armazenamento de dados sensíveis sempre que puder. Usar *tokens* compatível com *Payment Card Industry Data Security Standard* (PCI DSS) ou até mesmo truncamento. Assim, os dados que não são retidos não podem ser roubados. É muito importante também criptografar todos os dados que são trafegados com protocolos seguros, como *Transport Layer Security* (TLS) com cifras de sigilo de encaminhamento perfeito (PFS), priorização de cifras pelo servidor e parâmetros seguros. Por fim, é interessante aplicar a criptografia usando diretivas como *HTTP Strict Transport Security* (HSTS).

Em ataques do tipo entidades externas de XML (XXE) é importante usar, sempre que possível, formatos de dados menos complexos, como *JavaScript Object Notation* (JSON), e evite a serialização de dados confidenciais. Outro ponto é ter uma maior atenção com os processadores e bibliotecas XML, verificando se existem correções e atualizações destes processadores e bibliotecas. Também é importante implementar validação, filtragem ou sanitização de entrada positiva, para evitar dados hostis em documentos, cabeçalhos ou nós XML. Por fim, é sempre bom verificar se a funcionalidade de *upload* de arquivo XML ou *eXtensible Stylesheet Language* (XSL) valida XML de entrada usando validação *XML Schema Definition* (XSD) ou similar.

É importante considerar, para ataques de quebra de controle de acesso, que o controle de acesso só é eficaz se aplicado em implementação no lado servidor confiável ou API sem servidor, onde o invasor não pode modificar a verificação de controle de acesso ou metadados. Implementar uma solução de permissão para que usuários tenham perfis e, estes perfis, especificam as ações que o usuário pode realizar na aplicação. Sempre antes de realizar uma ação por parte do usuário, observar o nível de permissão ao recurso desejado na ação, sessão do usuário etc. É sempre bom manter o *log* dos acessos, registrando falhas de controle de acesso, alertando os administradores quando apropriado, por exemplo falhas repetidas.

Para excluir ou reduzir ao máximo os riscos de ataques por configurações de segurança incorretas é preciso ficar atento com as configurações padrões dos sistemas ou *frameworks* utilizados na aplicação. Alterar as credenciais para senhas mais robustas

ou desativar contas padrões é o primeiro passo para uma aplicação mais segura. Identificar vulnerabilidades de versões de ferramentas ou de linguagens de programação também se faz necessário a fim de manter a segurança em dias, pois versões atualizadas destes componentes são, em sua grande maioria, melhorias e correções de falhas identificadas. Além de todos esses passos, não se pode deixar de lado uma boa varredura nas regras de negócio da aplicação com o intuito de verificar o que já não é mais utilizado para poder ser descartado do sistema, pois páginas obsoletas também são grandes chamativos para os atacantes.

Os ataques por XSS por serem muito famosos, existem muitas formas de prevenção. Por se tratar de um ataque que ocorre, geralmente, no navegador do cliente, é normal imaginar que apenas ele deve tomar os devidos cuidados para não ser alvo de um ataque. Entretanto, o servidor da aplicação também deve ser protegido por meio de bibliotecas Anti-XSS. *Encodings* e *validations* podem ser muito úteis na prevenção de ataques por XSS, tendo em vista que o primeiro tem a função de filtrar dados que o usuário insere no navegador, fazendo-o interpretar estes dados não como código. Já o segundo, tem a função de filtrar todas as entradas do usuário que podem vir a ser maliciosas para a aplicação. O *Content Security Policy* (CSP) também pode ser uma forma de manter a aplicação segura por permitir o fornecimento de uma lista de parâmetros do que poderá ser ou não carregado no site. Existem ainda métodos como o HttpOnly, que protege os *cookies* do cliente, mesmo que ele acesse um *link* malicioso e o seu navegador seja atacado e existe o *Cross Site Scripting Protection* (X-XSS) que protege o navegador contra ataques de Reflected XSS.

Para a prevenção de ataques de desserialização insegura é importante que a aplicação não aceite objetos serializados de fontes não confiáveis. Faz-se necessário a implementação de sistemas de verificações de integridade como assinatura digital, por exemplo, a fim de evitar a alteração dos dados e a criação de objetos maliciosos. O monitoramento da rede de entrada ou de saída e a restrição do que passa na rede também é de suma importância para a segurança da aplicação. Monitorar as desserializações dos arquivos, informando sempre ao sistema caso algum usuário esteja praticando essa ação muitas vezes, e tentar ao máximo manter a parte do código que executa a desserialização isolada em áreas de baixo privilégio mantém a aplicação com riscos mínimos de ataques.

Uma aplicação com componentes vulneráveis torna-se facilmente alvo de ataques. É importante ressaltar que para manter a aplicação segura e livre de qualquer ação criminosa os componentes que fazem parte da construção do sistema são imprescindíveis estarem atualizados. As atualizações são uma forma de, em sua grande maioria, corrigir falhas identificadas anteriormente e prevenir que as aplicações não sofram danos irreparáveis. Buscar continuamente identificar quais componentes estão ficando obsoletos e quais impactos eles podem causar para a aplicação mantém um padrão de segurança alto para o software, pois obtendo conhecimento dessas pequenas falhas fica mais fácil tratar esses erros o mais breve possível.

Ataques causados por registro e monitorização insuficiente podem ser facilmente mitigados com pequenas, mas importantes ações dentro dos aplicativos web. Uma técnica que bloqueie o número ilimitado de tentativas de autenticação de um usuário, por exemplo, é uma pequena implementação que pode bloquear um exaustivo número de requisições de autenticação, evitando que a aplicação fique inoperável ou que o atacante

consiga descobrir senhas do sistema por inúmeras tentativas. Monitorar e alertar eficazmente a aplicação sobre detecção de atividades suspeitas para que sejam tomadas atitudes em tempo hábil. Certificar que todos os logins e falhas no sistema sejam registradas a fim de serem estudadas para identificar contas suspeitas ou mal-intencionadas e isolar essas contas ou falhas para análise e medidas cabíveis.

5. Conclusão

Este trabalho apresentou um levantamento exploratório dos principais riscos e ataques em aplicações web, elaborou uma discussão sobre o levantamento feito e analisou soluções técnicas no intuito de diminuir ou eliminar os riscos e ataques apresentados. Para isso, utilizou-se os aspectos teóricos sobre aplicações web, protocolo de comunicação HTTP, arquitetura cliente-servidor, páginas web, servidores, conteúdos estáticos e geração de conteúdo dinâmico. Além disso, comentou-se, por meio do documento OWASP Top 10, sobre dez tipos críticos de riscos e ataques em aplicações web. Em termos metodológicos, este ensaio serviu-se da pesquisa exploratória e da pesquisa não-experimental. Na etapa da pesquisa exploratória, em particular, foi utilizado o levantamento bibliográfico para reunir os conceitos teóricos necessários para o entendimento do trabalho. Já a pesquisa não-experimental foi adotada para tirar conclusões a partir de um arcabouço teórico reunido na etapa da pesquisa exploratória.

Como resultados e discussão desta pesquisa foram identificados como ocorrem dez tipos críticos de riscos e ataques em aplicações web. Diante dos resultados e discussões, pode-se perceber que foi possível responder à questão do problema de pesquisa definida na Seção 1. Por meio da etapa exploratória foi comentado como dez tipos críticos de riscos e ataques em aplicações web. Além disso, ao final deste estudo, foi possível confirmar a hipótese que foi formulada. Na Seção 4 foi discutido como soluções de implementação e ações podem mitigar ou eliminar os riscos e ataques em aplicações web. Assim pode-se comentar, como maior conclusão do trabalho, que existem diversos tipos de ataques e riscos em aplicações web, podendo reduzir ou eliminar tais ataques e riscos quando os desenvolvedores conhecem os comportamentos destas vulnerabilidades e aplicam soluções de implementação ou tecnologias.

Como trabalhos futuros, almeja-se: i) realizar um estudo mais aprofundado no intuito de elencar mais ataques e enriquecer a pesquisa; ii) elaborar protótipos ou provas de conceito para realizar uma simulação para que desenvolvedores possam visualizar, de forma mais didática, a ocorrência dos ataques; e iii) realizar uma avaliação com desenvolvedores para descobrir como eles tratam os riscos e ataques em aplicações web.

Referências

Conallen, J. (2003). *Building Web Applications with UML.* Addison-Wesley Object Technology Series. Addison-Wesley.

Corrêa, M. V. H. and Gonçalves, R. C. (2019). Mecanismos para Quebra de Políticas de Acesso. Monografia de Graduação. Graduação em Ciência da Computação, Universidade Federal Rural de Pernambuco, Garanhuns, Brasil.

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Bookman Editora, 5a. edition.

Gil, A. C. (2002). *Como Elaborar Projetos de Pesquisa*. Atlas, São Paulo, 4a. edition.

Jahromi, H. Z., Delaney, D. T., and Hines, A. (2020). Beyond first impressions: Estimating quality of experience for interactive web applications. *IEEE Access*, 8:47741–47755.

Kurose, J. F. and Ross, K. W. (2010). *Redes de computadores e a Internet: uma abordagem top-down*. Addison Wesley, 5a. edition.

Maciel, D. P. (2019). Detecção e Análise de Vulnerabilidades em um Servidor OJS utilizando Teste de Penetração. Monografia de Graduação. Tecnólogo em Redes de Computadores, Campus de Quixadá, Universidade Federal do Ceará.

Mocelin, B. V., Farias, K., Gonçales, L., and Bischoff, V. (2018). Melhorias no processo de identificação de componentes vulneráveis: Decidindo sobre atualizações. In *Anais do XIV Simpósio Brasileiro de Sistemas de Informação*, pages 333–340, Porto Alegre, RS, Brasil. SBC.

Moreira, L. O. (2014). *Abordagem para Qualidade de Serviço em Banco de Dados Multi-Inquilinos em Nuvem*. PhD thesis, Doutorado em Ciência da Computação, Centro de Ciências, Universidade Federal do Ceará, Fortaleza, Brasil.

OWASP (2021a). OWASP Foundation - Open Source Foundation for Application Security. Disponível em: https://owasp.org/. Acessado em 06 de junho de 2021.

OWASP (2021b). OWASP Top Ten Web Application Security Risks - OWASP. Disponível em: https://owasp.org/www-project-top-ten/. Acessado em 11 de junho de 2021.

Pacheco, F. G. (2019). Análise das Técnicas de Segurança do Framework Laravel contra Ataques as Aplicações Web. Monografia de Graduação. Graduação em Ciência da Computação, Universidade Federal Rural de Pernambuco, Garanhuns, Brasil.

Pereira, P. d. S. and Esmeraldo, G. A. R. M. (2019). Identificando vulnerabilidades web com software livre. *Revista Acta Kariri - Pesquisa e Desenvolvimento*, 4(1):33–46.

Prokhorenko, V., Choo, K.-K. R., and Ashman, H. (2016). Web application protection techniques: A taxonomy. *Journal of Network and Computer Applications*, 60:95–112.

Reis, A. P. (2018). Análise de vulnerabilidades de segurança em sistemas de Internet Banking utilizando ferramentas de código aberto. Monografia de Graduação. Bacharelado em Sistemas de Informação, Faculdade de Computação, Universidade Federal de Uberlândia.

Souza, W. C. (2018). Construtor de Sistemas Web. Monografia de Graduação. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Instituto Federal de Santa Catarina, Gaspar, Brasil.

Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall, 2nd edition.

Vora, P. (2009). *Web Application Design Patterns*. Interactive Technologies. Elsevier Science.

Wazlawick, R. S. (2009). *Metodologia de Pesquisa para Ciência da Computação*. Elsevier Editora.