IDENTIFICANDO VULNERABILIDADES WEB COM SOFTWARE LIVRE

Paulo da Silva Pereira[1,2], Guilherme Álvaro Rodrigues Maia Esmeraldo[2]

[1] Escola Estadual de Educação Profissional Presidente Médici, Campos Sales – CE

[2] Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Crato

paulo.s.p@live.com, guilhermealvaro@ifce.edu.br

RESUMO*: A Internet na atualidade conecta o mundo, trazendo benefícios como a comunicação, o comércio eletrônico, redes sociais, entretenimento, entre outros. Mas toda essa infraestrutura pode esconder problemas que colocam a vida de organizações e os usuários em risco, pois as aplicações utilizadas na internet nem sempre estão minimamente aceitáveis no que diz respeito à segurança da informação. A ação de atacantes fazem uso dessas vulnerabilidades, realizando procedimentos que alterem a lógica da programação de aplicações, permitindo a manipulação e roubo de dados, identificação de senhas, obtenção de acesso restrito, dinheiro e informações privilegiadas. O presente trabalho vem apresentar as 10 principais vulnerabilidades encontradas em aplicações web, segundo o documento OWASP top 10 2013. Traz também uma lista de ferramentas software livre e que podem detectar problemas de segurança em aplicações web por meio de auditorias de segurança. E, por fim, ainda apresenta formas de correção adequadas e práticas que devem ser adotadas por organizações e desenvolvedores criadores de software web.*

Palavras-chave: *Aplicações Web. Vulnerabilidade. OWASP. Segurança da Informação. Software Livre.*

ABSTRACT: *Today's Internet connects to the world, bringing benefits such as communication, e-commerce, social networks, entertainment, among others. But all this infrastructure can hide problems that put the lives of associations and users at risk, as the applications used on the internet are not always minimally acceptable with regard to information security. Attackers make use of these vulnerabilities, performing procedures that alter the logic of application programming, allowing data manipulation and theft, password identification, obtaining restricted access, money and privileged information. The present work presents the 10 main vulnerabilities found in web applications, according to the document OWASP top 10 2013. It also brings a list of free software tools that can detect security problems in web applications through security audits. And, finally, it still presents forms of correction found and practices that must be adopted by associations and creators of web software developers.*

Keywords: *Web Applications. Vulnerability. OWASP. Information Security. Open Source.*

1. INTRODUÇÃO

Desde a criação da ARPANet em 1969, houve uma evolução crescente onde foram adicionados novos recursos e tecnologias, possibilitando o surgimento de uma rede complexa e gigantesca, chamada de Internet, que proporcionou a comunicação entre computadores (KUROSE e ROSS, 2010).

Tim Berners-Lee de 1989 à 1991 deu origem a *web* com a criação de três tecnologias: HTML (*HyperText Markup Language*), URI (*Uniform Resource Identifier*) comumente chamado de URL e HTTP (*Hypertext Transfer Protocol*) (WEBFOUNDATION, 2016; KUROSE e ROSS, 2010), que formaram a base fundamental para as tecnologias usadas atualmente em *sites*, *blogs*, redes sociais, fóruns *on-line* entre outros sistemas que utilizam tecnologias *web*.

Toda essa infraestrutura nem sempre está minimamente segura o que coloca organizações e usuários em risco, com o intuito minimizar esses fatores negativos esse artigo vem apresentar um documento “OWASP top 10”, que identifica os dez riscos de segurança mais críticos em aplicações

web. Este artigo trata das 10 principais vulnerabilidades, em ordem de risco e prevalência, descrevendo suas características e os danos causados quando a aplicação sofre ataques. Na sequência, serão apresentadas ferramentas livres que buscam detectar e bloquear as referidas falhas por meio de auditorias de software, desta forma, alcançando confidencialidade, integridade e disponibilidade, que são os objetivos de segurança em aplicações *web* (SCP, 2012).

2. TOP 10 VULNERABILIDADES

A classificação da OWASP Top 10 2013 denomina as seguintes vulnerabilidade em aplicações *web*: A1 – Injeção de Código, A2 – Quebra de autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting (XSS), A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança, A6 – Exposição de Dados Sensíveis, A7 – Falta de Função para Controle do Nível de Acesso, A8 – Cross-Site Request Forgery (CSRF), A9 – Utilização de Componentes Vulneráveis Conhecidos e A10 – Redirecionamentos e Encaminhamentos Inválidos. Esta seção descreve cada uma dessas vulnerabilidades.

2.1. Vulnerabilidade A1: Injeção de Código

Representado pelo top 10 2013 como "A1" como a vulnerabilidade mais frequente em aplicações *web*. A aplicação vulnerável a ataques de injeção de código pode sofrer acesso irrestrito do atacante ao sistema e servidor *web*, bem como a perda, roubo, exclusão de dados sigilosos de clientes e empresas (SILVA, 2014). Ela ocorre quando a aplicação permite que dados não confiáveis sejam encaminhados para um interpretador, que pode receber consultas do tipo: SQL, LDAP, Xpath, NoSQL e comandos do Sistema Operacional (TOP 10, 2013). O atacante malicioso realiza injeção de código manipulando caracteres de entrada da aplicação para obter acesso ao sistema e informações sensíveis (MOZZAQUATRO, 2012). A correção para injeção de código baseia-se em tratamento de entrada de dados dos usuários, seja ele interno ou externo ao sistema. A biblioteca ESAPI oferece rotinas de filtragem e validação de entradas de caracteres (TOP 10, 2013), e pode ser associada à seção "Validação dos Dados de Entrada" do guia SPC, que contem orientações sobre o tratamento adequado de entradas na aplicação (SPC, 2012). O resultado deve ser avaliado conforme a seção V5 (Requisitos de Verificação da Validação de Entradas), do documento ASVS, que analisa a eficácia do tratamento de entradas da aplicação (ASVS, 2009).

2.2. Vulnerabilidade A2: Quebra de Autenticação e Gerenciamento de Sessão

Permitem que atacantes obtenham acesso ao sistema burlando a autenticação, e possam realizar manipulações em sessões de todas as contas de usuários, inclusive as privilegiadas (AMARO, 2012). Ações de atacantes maliciosos ocorrem devido a erros de codificação no processo de desenvolvimento da aplicação (SILVA, 2008). A exploração é caracterizada por um atacante possuir acesso aos dados e configurações de um usuário, seja ele do tipo comum ou administrativo (MARCOS, 2013). Podem ser exploradas utilizando um *sniffer* – farejador de pacotes de redes – capturando identificadores de sessões e senhas, ou realizando manipulações na execução de funções de cadastro de usuário, recadastramento e alteração de senhas com ajuda de um *proxy* (servidor intermediário entre as requisições de clientes e recursos providos por servidores web) (TOP 10, 2013). Existem ainda outras formas de exploração como XSS e CSRF que serão tratados neste trabalho nas seções 2.4 e 2.9 respectivamente. A correção recomendada é um conjunto forte de

autenticação e controle de sessão, que pode ser alcançado utilizando ESAPI Authenticator (TOP 10, 2013), com as seções de "Autenticação" e "Gerenciamento de Credenciais", que orientam no processo de validação e reconhecimento dos usuários, bem com "Gerenciamento de Sessões", que trata do início, todo o gerenciamento e finalização da sessão, presentes no guia SCP (SCP, 2012). As áreas V2 (Autenticação) e V3 (Gerenciamento de Sessão do ASVS) devem ser usadas para parametrizar a qualidade da autenticação e gerenciamento de sessão da aplicação (ASVS, 2009).

2.3. Vulnerabilidade A3: Cross-Site Scripting (XSS)

Sistemas vulneráveis a XSS permitem que atacantes capturem informações sigilosas como roubo de sessão do usuário e dados confidenciais, ou ataques de desfiguração de páginas e redirecionamento de usuários para sites de conteúdo malicioso (NUNAN et al., 2012). Existem três formas conhecidas de exploração XSS, persistente, refletido e baseado no DOM (MOZZAQUATRO, 2012). O Persistente deixa um *script* armazenado na aplicação, que é carregado toda vez que o usuário a acessa; já o refletido consiste em um atacante enviar *links* manipulados com injeção de código para usuários (NUNAN et al., 2012); por fim, o DOM (*Document Object Model*) produz elementos de HTML utilizando JavaScript e pode fazer alterações diretamente no navegador sem realizar consultas ao servidor de aplicações (SILVA, 2014). A correção de XSS é a mesma apresentada na Vulnerabilidade A1 (Injeção de Código). Injeção de Código e XSS são causados por tratamento inadequado de entradas de usuários internos ou externos ao sistema (SILVA, 2014).

2.4. Vulnerabilidade A4: Referência Insegura e Direta a Objetos

Uma aplicação está vulnerável a Referência Insegura e Direta a Objetos quando permite que um usuário ou atacante possa acessar objetos da aplicação como arquivos, pastas ou registros da base de dados sem a devida autorização (MARQUES, 2014). A alteração de caminho de pastas, nome de arquivos e variáveis na URL da aplicação, permitem manipular as referências para os objetos do sistema que estão sem verificação de autorização, facilitando a ação de atacantes na obtenção e manipulação de objetos (HOLANDA e FERNANDES, 2011). Todo acesso a objetos internos do sistema deve ter a autorização verificada. O guia SCP, na seção "Controle de Acessos", informa as características de proteção e controle de acessos a serem implantados na aplicação (SCP, 2012). No ASVS, seção V4 (Controle de Acesso), é descrito o perfil de qualidade esperado em uma aplicação segura em relação ao controle de autorização para acesso a dados (ASVS, 2009).

2.5. Vulnerabilidade A5: Configuração Incorreta de Segurança

Ocorre devido a configuração incorreta em servidores de aplicação, banco de dados, *frameworks*, uso de contas padrão, componentes desatualizados e serviços desnecessários (TOP 10, 2013). De acordo com Mozzaquatro (2012), o desconhecimento ou desinteresse de desenvolvedores e administradores de sistemas podem proporcionar grandes possibilidades para que um atacante faça uso de contas padrão, componentes desatualizados e falhas de configurações para ter acesso ao sistema. O atacante encontra na aplicação a listagem de diretórios ativada, banco de dados, servidor *web* e *framework* desatualizados ou com configurações padrão. Ele pode acessar diretórios e arquivos, realizar tentativas de logar na aplicação utilizando os usuários e senhas padrões dos sistemas (AMARO, 2012) ou pesquisar sobre falhas conhecidas presentes em versões desatualizadas dos componentes instalados. Utilizando um escâner, que são programas de varredura de redes, utilizados para detectar vulnerabilidades em sistemas web, é possível identificar falta de

atualização, configurações incorretas, contas padrão e serviços desnecessários em aplicações vulneráveis (TOP 10, 2013). De acordo com o guia SCP, se forem seguidas as recomendações da seção "Configuração do Sistema", que trata dos procedimentos referentes a atualização e configuração de componentes utilizados pela aplicação (SCP, 2012), o *software* atenderá os requisitos necessários de correção desta vulnerabilidade.

2.6. Vulnerabilidade A6: Exposição de Dados Sensíveis

Ocasionado por uma implementação de segurança de dados sensíveis equivocada ou até mesmo com a inexistência de algoritmos de criptografia e *hashing*. O que permite a exposição de dados de cartão de crédito, senhas e dados pessoais que estão armazenados no servidor ou em trânsito (TOP 10, 2013). De acordo com Marcos (2013) os desenvolvedores devem adotar uma configuração correta de cifras fortes de encriptação como AES, RSA ou SHA-256, abandonando algoritmos provados como fracos (e.g. MD5, SHA-1, RC3 e RC4). Ocorre normalmente por captura de pacotes utilizando um ataque conhecido como *man-in-the-middle* – em que o atacante intercepta informações trocadas entre o servidor e o cliente, realizando a captura de dados sensíveis em texto puro ou cifrados (MONTEVERDE e CAMPIOLO, 2014). Em caso de dados capturados em *hashing* ou criptografados, deve ser realizado a decifração para obter as informações contidas. Os dados sensíveis devem ser protegidos com eficiência conforme orienta o guia SCP, nas Seções "Praticas de Criptografia" e "Segurança nas Comunicações". Elas orientam sobre os métodos adequados no armazenamento e transporte de dados de forma criptografada (SCP, 2012). Cabe ainda verificar a qualidade dessas implementações de acordo com as seções V7 (Criptografia), V9 (Proteção de dados) e V10 (Segurança da Comunicação) do ASVS (ASVS, 2009).

2.7. Vulnerabilidade A7: Falta de Função para Controle do Nível de Acesso

Desenvolvedores utilizam técnicas que escondem caminhos para arquivos, funções e páginas com privilégios, acreditando que a ocultação dessas informações torne aplicação protegida contra-ataques (ASSUNÇÃO, 2014). Mas nada impede que um atacante utilize ferramentas e técnicas para encontrar URLs e *links* que podem levar a funcionalidades críticas da aplicação (SILVA, 2008), e posteriormente, transformarem-se em fontes de ataques de manipulação de funções da aplicação, acesso às páginas restritas e obtenção de acesso administrativo sem a devida autorização. Essa vulnerabilidade pode ser identificada com a utilização de um *proxy* capturando dados da aplicação em execução (TOP 10, 2013). Confirmada a falha de validação de autorização, é possível executar um ataque utilizando técnicas de forca bruta e adivinhação de *links* em busca de páginas desprotegidas (SILVA, 2008). Outra possibilidade seria alterar os caminhos e atributos na URL de forma manual, com o objetivo de encontrar funções e páginas privilegiadas ou tentar acessar a aplicação como administrador. A correção de Falta de Função para Controle do Nível de Acesso seque a mesma linha de implementação descrita na seção 4 (Corrigindo Referência Insegura e Direta a Objetos). Essas vulnerabilidades possuem o mesmo problema de permitem acessos a dados internos da aplicação sem autorização.

2.8. Vulnerabilidade A8: Cross-Site Request Forgery (CSRF)

As vítimas de CSRF estão sujeitas a enviar credenciais de acesso para atacantes e sites maliciosos, realizar transferências financeiras e compras online com quantidade de produtos alterada e endereços de entrega desconhecidos (SILVA, 2014). Segundo Uto e Melo (2009), um ataque de

CSRF consiste na alteração de parâmetros da URL e posterior disponibilização desta URL forjada, induzindo a vítima a acessar e executar ações em uma aplicação vulnerável sem que ela saiba. Para um ataque bem-sucedido é necessário que a vítima esteja autenticada na aplicação vulnerável (SILVA, 2008). O atacante altera uma URL com parâmetros reconhecidos pela aplicação e envia para a vítima, que executará de forma forçada o *link* ou imagem contendo uma URL forjada, podendo realizar diversas ações de forma legítima de acordo com a solicitação do usuário autenticado. Uma solução viável está no documento ASVS, seção V11 (Segurança do HTTP), especificamente no requisito de verificação V11.7, que trata da criação de *token* (cadeia de caracteres que identifica um usuário, aplicativo ou página) aleatório para *links,* formulários com transações, informações sensíveis e verificação da presença desse *token* pela aplicação (ASVS, 2009). Também é possível exigir que o usuário realize nova autenticação ou prove ser um usuário com uso do CAPTCHA para executar funções sensíveis (TOP 10, 2013).

2.9. Vulnerabilidade A9: Utilização de Componentes Vulneráveis Conhecidos

A utilização de componentes como *frameworks*, bibliotecas e outros módulos oferecem praticidade no desenvolvimento de aplicações, mas se estiverem desatualizados ou com vulnerabilidades conhecidas podem trazer grandes riscos à aplicação e a seus usuários (MARQUES, 2014). Aplicações que apresentam componentes desatualizados ou com vulnerabilidades conhecidas estão criticamente vulneráveis, pois módulos e complementos quase sempre são executados com privilégios elevados (MARQUES, 2014). O *software* estará sujeito a diversos tipos de ataques, incluindo os já mencionados neste trabalho e a outras práticas que variam de acordo com nível da vulnerabilidade encontrada em componentes utilizados. Assim como na seção 2.5, deste trabalho, que trata da Configuração Incorreta de Segurança, a Utilização de Componentes Vulneráveis Conhecidos também apresenta problemas na administração correta dos componentes utilizados. Sua solução pode ser encontrada, também neste trabalho, na seção 2.5 (Corrigindo Configuração Incorreta de Segurança).

2.10. Vulnerabilidade A10: Redirecionamentos e Encaminhamentos Inválidos

Este tipo de vulnerabilidade coloca em risco o usuário através do redirecionamento para sites maliciosos que usam técnicas de *phishing* (golpe eletrônico cujo objetivo é o furto de dados pessoais, tais como número de CPF e dados bancários) ou realizam a instalação de *malware* no sistema operacional. Essa falha ainda permite encaminhar atacantes para funções internas e administrativas, as quais não deveriam ter acesso dentro da aplicação vulnerável. Esses problemas ocorrem devido uma validação equivocada de redirecionamento e encaminhamento (MOZZAQUATRO, 2012). A exploração consiste na inserção de URL forjada em uma aplicação vulnerável, a qual realiza o redirecionando de usuários para sites de atacantes ou encaminha um atacante para funções internas da aplicação, contornando o controle de acesso (SILVA, 2014). Uma alternativa de solução é realizar a checagem dos encaminhamentos e redirecionamentos por meio de lista branca ou negra (MARCOS, 2013). É recomendado utilizar mapeamento para o caminho de destino evitando expor a URL real, e o valor mapeado será traduzido no lado do servidor da aplicação identificando o destino correto. O uso da biblioteca ESAP pode garantir que os destinos de redirecionamento estão seguros (TOP 10, 2013).

3. FERRAMENTAS LIVRES PARA DETECÇÃO DE VULNERABILIDADES

Na literatura, há uma grande disponibilidade de ferramentas para uso em auditoria de sistemas de informação. Nesta seção, apresenta-se as principais ferramentas livres para detecção de vulnerabilidades em aplicações *web*. São elas:

- Stranger: é uma ferramenta de linha de comando, distribuída sob a licença GNU GPL v2, que pode encontrar falhas em aplicações *web* com a realização de um escaneamento recursivo no diretório da aplicação em todos os arquivos de forma automática. Possibilitando a identificação das vulnerabilidades A1 – Injeção de Código e A3 – Cross-Site Scripting (XSS) em sistemas *web* (STRANGER, 2016). No processo de varredura em busca de vulnerabilidades essa ferramenta verifica a sanitização na validação de dados de entrada, utilizando uma implementação baseada em autômatos finitos determinísticos (YU et al., 2010), que fazem a leitura de um gráfico de controle de fluxo gerado para um *script* ou diretório de *scripts* PHP.

- Wireshark: é um analisador de protocolo de rede, que permite verificação minuciosa dos dados capturados (WIRESHARK, 2016). Ele realiza capturas de pacotes de dados na rede de forma automática e permite várias opções de configuração de filtros que classificam os dados para observação em tempo real ou leitura de dados armazenados em arquivos. Pode ser útil para identificar das vulnerabilidades, A2 – Quebra de Autenticação e Gerenciamento de Sessão e A6 – Exposição de Dados Sensíveis.

- Ettercap: distribuído sob os termos da licença GNU GPL v2, é um analisador de protocolos de rede, que possui interfaces de linha de comando e gráfica. Desenvolvida por comunidade aberta, suporta vários tipos de *plugins* que adicionam ou estendem novos recursos a ferramenta, que possui uma politica organizada de implementação de funções e correção de *bugs*. Está disponível para vários sistemas operacionais e em diferentes arquiteturas (32/64-bits*)* (ETTERCAP, 2016). O Ettercap pode identificar a ocorrência das vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão e A6 – Exposição de Dados Sensíveis. Por conta do seu comportamento de captura de dados na rede, também pode ser utilizada em ataques do tipo *man-in-the-middle*.

- OWASP Zed Attack Proxy (ZAP): e uma das ferramentas de detecção de vulnerabilidades mais populares do mundo, e seu desenvolvimento é realizado em comunidade por centenas de voluntários internacionais (OWASP, 2016). Sua distribuição ocorre sob os termos da Licença Apache versão 2.0, possui interface gráfica, é multiplataforma e pode funcionar de forma automática e manual. É muito utilizada no processo de auditoria em aplicações *web* e na identificação de vulnerabilidades como: A1 – Injeção de Código, A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança, A7 – Falta de Função para Controle do Nível de Acesso, A8 – Cross-Site Request Forgery e A10 – Redirecionamentos e Encaminhamentos Inválidos. A ferramenta ZAP funciona como um *proxy*, capturando informações de aplicações *web*, e utiliza métodos de testes de invasão (SCHEFFER et al., 2015), com o objetivo de encontrar falhas de implementações equivocadas no tratamentos de entradas de caracteres, configurações inadequadas dos componentes utilizados, disponibilidade de acesso a arquivos e páginas administrativas.

- W3AF: é distribuída sob os termos da licença GNU GPL v2, possui interface gráfica e de linha de comando, é multiplataforma e funciona de forma automática e manual na busca de falhas de segurança em sistemas *web*. Essa ferramenta é muito utilizada para realizar auditorias em aplicações *web*, e permite adicionar uma grande variedade *plugins,* que agregam funções que facilitam na identificação de vários tipos de vulnerabilidades, entre elas estão: A1 – Injeção de Código, A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança, A7 – Falta de Função para Controle do Nível de Acesso, A8 – Cross-Site Request Forgery e A10 – Redirecionamentos e Encaminhamentos Inválidos. Durante um processo de auditoria, esta ferramenta realiza o escaneamento de todos os arquivos e diretórios de uma aplicação *web* (SCHEFFER et al., 2015), com o objetivo de encontrar vulnerabilidades que possam comprometer a segurança do sistema analisado.

- NIKTO 2: de acordo com o site cirt.net, Nikto 2 é um *scanner* que realiza testes abrangentes contra servidores *web*, faz a checagem em mais de 6.700 arquivos potencialmente perigosos, verifica versões desatualizadas de mais de 1250 tipos de serviços e problemas específicos de versão em mais de 270 tipos de servidores. É multiplataforma, possui interface de linha de comando, distribuído sob os termos da licença GNU GPL, permite adição de novas funcionalidades por meio de *plugins* e pode identificar vulnerabilidades *web* como: A5 – Configuração Incorreta de Segurança e A9 – Utilização de Componentes Vulneráveis Conhecidos. Após a análise de uma aplicação com a ferramenta Nikto 2, é possível reportar todas as informações obtidas em relatórios de saída nos formatos XML, HTML e CSV.

- HYDRA: Essa ferramenta é desenvolvida em comunidade e distribuída sob os termos da licença GNU GPL v3, é multiplataforma e possui as interfaces gráfica e linha comando. Muito utilizada para ataques automatizados de quebra de autenticação remota, com uso de técnicas de força bruta. Pode identificar e auxiliar na exploração das vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão e A5 – Configuração Incorreta de Segurança. Entre os ataques mais utilizados da ferramenta HYDRA, o uso da técnica de força bruta de tentativa e erro percorrendo uma *wordlist* em busca da informação correta para descoberta de senhas em servidores. Os ataques desta ferramenta podem ser utilizados com diversos protocolos de rede contra servidores mal configurados e que contenham senhas fracas.

- SQLMap: desenvolvida em comunidade por voluntários e distribuída sob a licença GNU GPL v2, possui interface de linha de comando e permite incorporação de *plugins* adicionando novas características a ferramenta. Age de forma automatizada no processo de detecção e exploração de vulnerabilidades (SQLMAP.ORG, 2016), pode ser usada para identificar as falhas, A1 – Injeção de Código, A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança e A6 – Exposição de Dados Sensíveis. SQLMap é possui suporte para exploração dos sistemas de gerenciamento de banco de dados e pode executar as principais técnicas de injeção SQL conhecidas (SCHEFFER et al., 2015), reconhece automaticamente formatos de *hash* de senhas, podendo decifrá-los usando técnicas baseados em dicionário, permite listagem de usuários, senhas, privilégios, *download* e *upload* de arquivos.

- Parsero: é um *script* distribuído sob a licença GNU GPL v2 que faz a leitura dos arquivos robots.txt presentes em vários sistemas *web*. O arquivo robots.txt identifica os caminhos de arquivos e pastas de uma aplicação *web* que devem ou não serem indexados por motores de buscas (e.g. Google, Bing e Yahoo), esses caminhos podem direcionar usuários para diretórios e arquivos internos da aplicação, permitindo acesso indevido e a exploração das vulnerabilidades A4 – Referência Insegura e Direta a Objetos, A7 – Falta de Função para Controle do Nível de Acesso e A10 – Redirecionamentos e Encaminhamentos Inválidos.

- Dirsearch: distribuída sob a licença GNU GPL v2, é executada em linha de comando e projetada para encontrar diretórios e arquivos em servidores *web* de forma automatizada (DIRSEARCH, 2016), podendo auxiliar na identificação e exploração das vulnerabilidades A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança e A7 – Falta de Função para Controle do Nível de Acesso. Essa ferramenta utiliza *multithreading* e técnicas de força bruta de forma recursiva, realizando varreduras em servidores *web* em busca de acesso irrestrito a arquivos e diretórios.

- John the Ripper: é um *software* livre de código aberto distribuído sob a licença GNU GPL v2, possui interface de linha de comando e o seu objetivo principal é decifrar senhas em *hash* ou criptografadas, de forma automatizada e utilizando ataques de força bruta e com suporte de dicionário. Atualmente está disponível para os sistemas operacionais Unix, Windows e GNU/Linux (OPENWALL, 2016). Pode ser utilizado para identificar as vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão e A6 – Exposição de Dados Sensíveis.

- Dsniff: consiste de uma coleção de ferramentas para auditoria de rede e testes de penetração (QUEIROZ, 2007), possui código aberto e distribuição livre sob os termos da licença BSD, está disponível apenas para sistemas GNU/Linux e UNIX. Essa coleção de ferramentas pode facilmente auxiliar na identificação das vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão e A6 – Exposição de Dados Sensíveis. Dsniff é formado pelas ferramentas: dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf e webspy, que monitoram de forma passiva uma rede de dados, arpspoof, dnsspoof e macof, que facilitam a intercepção de tráfego de rede, e por fim, sshmitm e webmitm, que auxiliam em ataques *monkey-in-the-middle*.

- Wapiti: é um *software* livre distribuído a sob a licença GNU GPL v2 e é executado em linha de comando para auditar a segurança de aplicações *web* realizando *teste de caixa-preta* com varreduras automatizadas (WAPITI, 2016). Pode identificar a existência das vulnerabilidades A1 – Injeção de Código, A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança, A6 – Exposição de Dados Sensíveis, A7 – Falta de Função para Controle do Nível de Acesso e A9 – Utilização de Componentes Vulneráveis Conhecidos. Durante sua execução, Wapiti realiza buscas de *scripts*, para realizar injeções de dados e cargas para encontrar falhas de referência direta a arquivos XSS refletido e permanente, detectar arquivos potencialmente perigosos usando à base de dados do Nikto, injeção de banco de dados e exposição de arquivos de *backup* com informações confidenciais.

- Tamper Data for FF Quantum: é um complemento do navegador Mozilla Firefox e derivados que pode capturar e alterar requisições enviadas e recebidas através do protocolo HTTPS (TAMPERDATA, 2016). Possui interface gráfica e distribuição livre sob termos da licença GNU GPL v2, funciona de forma automática e manual, podendo ser usado para identificar as vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A4 – Referência Insegura e Direta a Objetos e A7 – Falta de Função para Controle do Nível de Acesso. Com esse complemento, é possível realizar manipulações de parâmetros das requisições e respostas contidas nos cabeçalhos do protocolo HTTPS (MENDES, 2014). Essa ação permite que usuários mal-intencionados obtenham acesso a arquivos, diretórios e funções administrativas das aplicações.

- Firebug: é um complemento para o navegador Mozilla Firefox e derivados, possui um conjunto de ferramentas de desenvolvimento *web* que permite editar, depurar e monitorar CSS, HTML e JavaScript em qualquer página da *web* (GETFIREBUG, 2016). Possui interface gráfica e distribuição sob os termos da licença BSD. Essa ferramenta pode auxiliar na identificação das vulnerabilidades A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A6 – Exposição de Dados Sensíveis e A8 – Cross-Site Request Forgery. Esse complemento permite realizar inspeção, manipulação e identificação de erros de codificação em elementos HTML, CSS, JavaScript, *cookies e* o Document Object Model (DOM), que pode ter suas funções facilmente manipuladas por chamadas do JavaScript.

- OWASP Mantra: é um navegador *web* projetado especialmente para testes de segurança de aplicações *web*. Distribuído sob a licença Creative Commons Attribution-ShareAlike 3.0, está disponível para GNU/Linux, Windows e Mac OS. Possui interface gráfica e pode funcionar de forma automática ou manual de acordo com os complementos utilizados. Em execução esse navegador pode identificar a existência das vulnerabilidades A1 – Injeção de Código, A2 – Quebra de Autenticação e Gerenciamento de Sessão, A3 – Cross-Site Scripting, A4 – Referência Insegura e Direta a Objetos, A5 – Configuração Incorreta de Segurança, A6 – Exposição de Dados Sensíveis, A7 – Falta de Função para Controle do Nível de Acesso, A8 – Cross-Site Request Forgery e A10 – Redirecionamentos e Encaminhamentos Inválidos. OWASP Mantra é composto de vários complementos que auxiliam na auditoria de aplicações *web,* estão classificados em categorias denominadas: auditores de aplicação, utilitários de rede, editores, coletores de informações e *proxies* (GETMANTRA, 2016). Os complementos permitem modificar informações de cabeçalhos HTTPS, manipular cadeias de caracteres de entrada, realizar solicitações GET e POST, editar *cookies*, arquivos CSS, JavaScript e HTML, alternar rapidamente entre vários *proxys* e forçar redirecionamentos e encaminhamentos.

4. ANÁLISE DAS FERRAMENTAS LIVRES

As ferramentas estudadas foram distribuídas no Quadro 1 e identificadas de acordo com as vulnerabilidades que podem detectar em aplicações *web*.

Quadro 1. Ferramentas livres e as suas respectivas vulnerabilidades tratadas.

Ferramenta	Vulnerabilidade									
	A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
Stranger	x	-	x	-	-	-	-	-	-	-
Wireshark	-	x	-	-	-	x	-	-	-	-
Ettercap	-	x	-	-	-	x	-	-	-	-
OWASP Zed Attack Proxy (ZAP)	x	x	x	x	x	-	x	x	-	x
W3AF	x	x	x	x	x	-	x	x	-	x
NIKTO 2	-	-	-	-	x	-	-	-	x	-
HYDRA	-	x	-	-	x	-	-	-	-	-
SQLMap	x	-	-	x	x	x	-	-	-	-
Parsero	-	-	-	x	-	-	x	-	-	x
Dirsearch	-	-	-	x	x	-	x	-	-	-
John the Ripper	-	x	-	-	-	x	-	-	-	-
Dsniff	-	x	-	-	-	x	-	-	-	-
Tcpdump	-	x	-	-	-	x	-	-	-	-
Wapiti	x	x	x	x	x	x	x	-	x	-
Tamper Data for FF Quantum	-	x	x	x	-	-	x	-	-	-
Firebug	-	x	x	-	-	x	-	x	-	-
OWASP Mantra	x	x	x	x	x	x	x	x	-	x

Da análise do Quadro 1, percebe-se que as ferramentas estudadas conjuntamente, permitem identificar todas as vulnerabilidades OWASP top 10 2013. Entretanto, pode ser notado que nenhuma é capaz de identificar todas as vulnerabilidades de forma individual. Nesse caso, existe a necessidade de utilizar mais de uma ferramenta para que seja possível auditar um software *web* de forma completa, para obter mais qualidade nos resultados.

O Quadro 2, a seguir, sumariza as principais características das ferramentas estudadas. Entre as características estudadas estão: 1) **Interface**: Tipo de interface do usuário utilizada; 2) **Automatização**: Forma de identificação de vulnerabilidades: automática, manual ou ambas; 3) **Portabilidade (Multiplataforma):** Suporte a diferentes sistemas operacionais; 4) **Suporte Comunitário**: Possui desenvolvimento aberto pela comunidade; 5) **Eficiência**: Caracteriza análise com baixo consumo de recursos (tempo, memória, processador, rede, etc.); 6) **Eficácia:** Permite identificar a vulnerabilidade em quaisquer tipo de sistema *web*; 7) **Acesso (Remoto ou Local)**: Forma de análise do sistema *web*; 8) **Tipo de Licença**: GPL v2, GPL v3, BSD, Creative Commons Attribution-ShareAlike 3.0, LGPL v3; 9) **Extensibilidade**: Permite adição de novas funções ou subsistemas (*plugins*).

Quadro 2. Comparativo entre as ferramentas livres estudadas.

Ferramenta	Características								
	Interface	**Automatização**	**Portabilidade**	**Suporte Comunitário**	**Eficiência**	**Eficácia**	**Acesso**	**Licença**	**Extensibilidade**
Stranger	Lin. de Comando	Sim	Não	Não	Sim	Não	Local	GPL v2	Não
Wireshark	Gráfica	Não	Sim	Sim	Sim	Sim	Remoto	GPL v2	Não
Ettercap	Gráfica e Lin. de Comando	Não	Não	Sim	Sim	Sim	Remoto	GPL v2	Sim
OWASP Zed Attack Proxy (ZAP)	Gráfica	Sim /Não	Sim	Sim	Não	Sim	Remoto e Local	Apache v 2.0	Sim
W3AF	Gráfica e Lin. Comando	Sim/Não	Sim	Sim	Sim	Sim	Remoto e Local	GPL v2	Sim
NIKTO 2	Lin. Comando	Sim	Sim	Não	Sim	Sim	Remoto	GPL	Sim
HYDRA	Gráfica e Lin. de Comando	Sim	Sim	Sim	Não	Sim	Remoto	GPL v3	Não
SQLMap	Lin. de Comando	Sim	Sim	Sim	Sim	Sim	Remoto	GPL v2	Sim
Parsero	Lin. de Comando	Sim	Sim	Não	Sim	Sim	Remoto	GPL v2	Não
Dirsearch	Lin. de Comando	Sim	Sim	Sim	Sim	Sim	Remoto	GPL v2	Não
John the Ripper	Lin. de Comando	Sim	Sim	Sim	Não	Sim	Local	GPL v2	Não
Dsniff	Lin. de Comando	Não	Não	Não	Sim	Sim	Remoto	BSD	Não
Wapiti	Lin. de Comando	Sim /Não	Sim	Não	Sim	Sim	Remoto e Local	GPL v2	Não
Tamper Data for FF Quantum	Gráfica	Não	Sim	Sim	Sim	Sim	Remoto	GPL v2	Não
Firebug	Gráfica	Não	Sim	Sim	Sim	Sim	Local	BSD	Sim
OWASP Mantra	Gráfica	Sim /Não	Sim	Não	Não	Sim	Remoto e Local	CC 3.0	Sim

De acordo com o Quadro 2, pode ser observado que a maioria das ferramentas estudadas dão suporte ao seu uso por linha de comando. A maioria delas pode ser automatizada na busca por vulnerabilidades e utilizada em mais de um sistema operacional (multiplataforma). Em sua maior parte, são assistidas por uma comunidade de desenvolvedores que contribuem continuamente para seu desenvolvimento e correções de falhas. Observou-se ainda que um pequeno grupo dessas ferramentas pode consumir uma quantidade maior de recursos de hardware, necessitando assim de plataformas mais robustas para sua execução. Já no quesito eficácia, apenas uma, (Stranger) não atendeu a esse critério de forma satisfatória, pois somente identifica vulnerabilidades em sistemas *web* produzidos com a linguagem de programação PHP. No geral, as formas de acesso variaram entre local, remoto ou ambas as opções. Todas as ferramentas analisadas são disponibilizadas como software livre e suas restrições podem variar de acordo com a licença com quais são distribuídas. Por fim, verificou-se que menos da metade desses sistemas oferece suporte a *plugins*, que podem ser adicionados para assim acrescentar novos recursos às ferramentas.

5. CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo a orientação, com sugestões de materiais de apoio, sobre as formas de correção das vulnerabilidades mais frequentes em aplicações *web*.

Observa-se que não se identificou uma solução completa, que pudesse identificar todas as vulnerabilidades estudadas, entretanto foram apresentados recursos de segurança que podem ser adotados para melhor produção do *software* web ou correção de um já existente. As soluções propostas seguem um padrão estabelecido e testado pela OWASP, onde boa parte do que foi exposto neste trabalho pode solucionar problemas de vulnerabilidades não tratados em muitas aplicações.

Os trabalhos futuros incluem: elaboração de trabalhos que possam auxiliar diretamente no ciclo de desenvolvimento de aplicações *web*, detalhando os melhores métodos de implantação de segurança, voltados especificamente para as linguagens de desenvolvimento *web* mais utilizadas; estudos para aumento de segurança no desenvolvimento de sistemas que utilizam informações sigilosas, sem prejuízos à usabilidade da aplicação. Estas aplicações requerem maior atenção por serem mais visadas por atacantes; e, por fim, desenvolver uma nova ferramenta ou modificar alguma existente para que seja possível detectar todas as principais vulnerabilidades mencionadas neste trabalho. De forma que a auditoria da aplicação *web* seja realizada por um único *framework* de segurança.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARO, R. G. J. D. **dgs.SGA-Vulntracking Sistema automatizado de rastreio e gestão de vulnerabilidades Dognaedis**. 2012. 156 f. Dissertação (Mestrado em Engenharia Informática) Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra, 2012.

ASVS. **Application Security Verification Standard.** Disponível em : <https://www.owasp.org/images/2/2b/OWASP_ASVS_2009_Web_App_Std_Release_PT-BR.pdf>. Acesso em 21 de jun. 2016.

HOLANDA, M. T. de; FERNANDES, J. H. C. **Segurança no Desenvolvimento de Aplicações:** Gestão da Segurança da Informação e Comunicações - 2009-2011. Brasília: UNB. Departamento de Ciência da Computação, 2013. 43 p. Disponível em: <http://www.cic.unb.br/~jhcf/MyBooks/cegsic/2009_2011/GSIC701_Seguranca_Desenvolvimento_Aplicacoes.pdf>. Acesso em 19 de jun. 2016.

KUROSE, J.; ROSS, K. W. **Redes de Computadores e a Internet:** Uma Abordagem Top-down. 5ª edição São Paulo: Pearson, 2010.

MARCOS, D. G. **CodeV - Security Management Tool: Dognædis**. 2013. 129 f. Dissertação (Mestrado em Engenharia Informática) Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra, 2013.

MARQUES, M. M. **Abordagem Ontológica para Mitigação de Riscos em Aplicações Web**. 2014. 92 f. Dissertação (Mestrado em Informática) Departamento de Ciência da Computação, Instituto de Ciências Exatas, Universidade de Brasília – UnB, Brasília, 2014.

MENDES, D. C. **Técnicas de Hacking para Anonimização na Internet**. 2014. 80 f. Dissertação (Mestrado em Segurança de Sistemas de Informação) Universidade Católica Portuguesa, 2014.

MONTEVERDE, W. A; CAMPIOLO, R. **Estudo e Análise de Vulnerabilidades Web**. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, 14., 2014. Anais. Campo Mourão – PR: Universidade Tecnológica Federal do Paraná (UTFPR), 2014. p. 415 – 423.

MOZZAQUATRO, B. A. **Aplicando a Transformada Wavelet Bidimensional na Detecção de Ataques Web**. 2012. 88 f. Dissertação (Mestrado em Computação) Universidade Federal de Santa Maria, Rio Grande do Sul, 2012.

NUNAN, E.; SOUTO, E.; SANTOS, E. M. dos.; FEITOSA, E. **Classificação Automática de Cross-Site Scripting em Páginas Web**. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 30., 2012. Anais. Ouro Preto: MG, 2012. p. 450 – 463.

SCP. **OWASP Secure Coding Practices - Quick Reference Guide**. Disponível em: <https://www.owasp.org/images/b/b3/OWASP_SCP_v1.3_pt-BR.pdf>. Acesso em: 24 out. 2020.

OWASP. **OWASP Top 10 – 2013:** Os dez riscos de segurança mais críticos em aplicações web. 2013. Disponível em: <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013>. Acesso em: 24 out. 2020.

QUEIROZ, C. da C. **Segurança Digital: Um estudo de caso**. 2007. 71 f. Monografia (Bacharelado em Ciência da Computação) Faculdade Lourenço Filho, Fortaleza, 2007.

SCHEFFER, L. C., DE LIMA, M. C., MALLMANN, J., & ANDERLE, D. F. **Análise de Vulnerabilidades em Servidor Web: uma comparação de desempenho das ferramentas Nessus, OWASP ZAP e w3af**. Organizadores Vanderlei Freitas Junior Lucyene Lopes da Silva Todesco Nunes Thales do Nascimento da Silva Gerson Luis da Luz, 2015. p.148 – 178.

SILVA, C. M. R. da. **Aegis: Um Modelo de Proteção à Dados Sensíveis em Ambientes Client-side**. 2014. 130 f. Dissertação (Mestrado em Ciência da Computação) Universidade Federal de Pernambuco. Cin. Ciência da computação, Pernambuco, 2014.

SILVA, L. de S. **Uma metodologia para detecção de ataques no tráfego de redes baseada em redes neurais.** 2008. 256 f. Tese (Doutorado em Computação Aplicada) Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2008.

UTO, N.; MELO, S.P. **Vulnerabilidades em Aplicações Web e Mecanismos de Proteção**. In: Minicursos SBSeg, Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, 9., 2009. Campinas: São Paulo, 2009. p. 237-283.

WEBFOUNDATION. **History of the Web**: Sir Tim Berners-Lee invented the World Wide Web in 1989. Disponível em: <http://webfoundation.org/about/vision/history-of-the-web/>. Acesso em: 24 out. 2020.

YU, Fang; ALKHALAF, Muath; BULTAN, Tevfik. Stranger: An automata-based string analysis tool for PHP. In: **International Conference on Tools and Algorithms for the Construction and Analysis of Systems**. Springer Berlin Heidelberg, 2010. p. 154-157.