UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM SEGURANÇA CIBERNÉTICA

MÁRIO INÁCIO NEIS

O profissional hacker ético e a efetividade de ferramentas de detecção de vulnerabilidades open source

Monografia apresentada como requisito parcial para a obtenção do grau de Especialista em Segurança Cibernética.

Orientador: Prof. Dr. Alberto Egon Schaeffer-Filho

Porto Alegre
2019

AGRADECIMENTOS

Meus mais sinceros agradecimentos ao Prof. Dr. Alberto Egon Schaeffer-Filho, por todo o auxílio e orientação ao longo deste projeto, por instigar o meu senso crítico e por nunca aceitar menos do que o meu melhor possível. Não é possível colocar em palavras tudo o que eu aprendi ao longo deste ano. Muito obrigado.

Agradeço ao colega Pedro Vagner, pela disponibilidade e pela ajuda nos momentos tortuosos durante a jornada. Agradeço também a todos os professores do Instituto de Informática da UFRGS. Este trabalho não seria possível sem tudo o que eu aprendi ao longo desses anos no INF. Foi uma experiência fantástica.

Meus profundos agradecimentos a minha mãe e meu pai (*in memoriam*), minha esposa, Michele, pelo companheirismo e suporte de sempre, e minha filha, Rafaela, por ser o grande motivador para seguir a jornada. E a todos que, direta ou indiretamente, fizeram parte da minha formação, o meu muito obrigado.

Se eu vi mais longe, foi por estar sobre ombros de gigantes.

Isaac Newton

À Gl.'.do Gr.'.A.'.do U.'.

RESUMO

A necessidade de um profissional especialista na área de segurança cibernética se mostra cada vez mais presente, haja vista o crescente número de incidentes de segurança ocorrendo no mercado. Este trabalho apresenta um perfil do profissional de segurança hacker ético, e a necessidade crescente deste profissional para o mercado da tecnologia da informação, de forma direta e indireta. Inspirado em pesquisas anteriores, este trabalho apresenta uma análise comparativa do desempenho de ferramentas open source, de busca de vulnerabilidades em ambientes Web utilizadas por esse profissional quando executadas em suas configurações padrões.

Palavras-chave: Cibersegurança. Hacking. Ethical Hacking. Pentest. Detecção de vulnerabilidades.

The ethical hacker professional and the effectiveness of open source vulnerability detection tools

ABSTRACT

The need for a specialist in the area of cyber security is increasingly present, given the increasing number of security incidents occurring in the market. This work presents a profile of the ethical hacker security professional, and the growing need of such professional in the information technology market, directly and indirectly. Inspired by previous researches, this paper also presents a comparative analysis of the performance of open source vulnerability search tools in Web environments used by ethical hackers when executed in their default settings.

Keywords: Cyber security. Hacking. Ethical hacking. Pentest. Vulnerability detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programing Interface
CTF	Capture the Flag
DB	Data base
HD	Hard Disk
DBMS	Data base management system
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ISECOM	Institute for Security and Open Methodologies
ISSAF	Information Systems Security Assessment Framework
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
OISSG	Open Information Systems Security Group
OSSTMM	Open Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
PKI	public key infrastructure
RFC	Request for Comments
SGDB	Sistema de gerenciamento de banco de dados
SMTP	Simple Mail Transfer Protocol
SO	Sistema Operacional
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
WASC	Web Application Security Consortium
XSS	Cross-site Scripting
XML	Extensible Markup Language
XPATH	XML Path Language

SUMÁRIO

1 INTRODUÇÃO

Este trabalho apresenta um perfil do profissional de segurança hacker ético, a necessidade crescente deste profissional para o mercado da tecnologia da informação, de forma direta e indireta, e o desempenho de ferramentas de busca de vulnerabilidades em ambientes Web utilizadas por esse profissional quando executadas em suas configurações padrões. Nas próximas subseções serão apresentados aspectos deste trabalho, tais como sua motivação, objetivos, contribuições e organização do documento.

1.1 Motivação

O termo hacker é amplo e aceita uma grande variedade de interpretações. No trabalho efetuado por PATIL et al., 2017, é proposta uma classificação destes profissionais em tipos e categorias como *white-hats*, que *defendem* sistemas, *black-hats* que atacam sistemas, e *hacktivistas*, que atacam por motivação ideológica, entre outros, de modo a esclarecer as diferenças entre eles. A pesquisa efetuada por CHANDRIKA (2014), além de classificar os tipos de hackers em categorias, busca nas fontes históricas do termo para esclarecer como essas diferentes interpretações ocorreram.

Nesse contexto, trabalhos como TRABELSI; IBRAHIM, 2013; BISHOP, 2007; e LONG, 2012 mostram a necessidade desses profissionais, a busca pelo aumento da segurança cibernética (por meio do ensino sobre falhas e suas formas de mitigação e do desenvolvimento de sistemas de forma segura) e os passos e ferramentas utilizados por esses profissionais para a execução de suas tarefas.

Muitos trabalhos, como DOUPÉ et al., 2010; JIMÉNEZ, 2016; HAFELE, 2004; WANG; YANG, 2017; e RAO et al., 2014, efetuaram testes de capacidade e/ou eficácia de ferramentas de detecção de vulnerabilidades Web, especialmente as do tipo *black box*, que é um teste do tipo cego (no qual, por exemplo, o testador não conhece detalhes sobre o alvo do teste, como infraestrutura, sistema operacional e afins,). Contudo, em nenhum deles foi aventada a eficácia destes aplicativos quando utilizados em sua configuração padrão, como se fossem executados por alguém com conhecimentos técnicos de informática, porém sem a expertise de um especialista, ou por um entusiasta de outra área técnica.

No trabalho proposto em DOUPÉ et al., 2010, por exemplo, é sugerido que estas ferramentas de detecção de falhas não terão uma eficácia boa se utilizadas por um usuário comum, ou seja, sem os conhecimentos técnicos necessários para configurar a melhor forma de execução destes testes.

1.2 Objetivos

O objetivo principal deste trabalho é traçar um perfil do profissional hacker ético e diferenciá-lo dos outros tipos de hackers existentes, haja vista sua crescente necessidade no mercado da tecnologia da informação e a grande carga de desinformação associada ao termo hacker, e mensurar o desempenho das ferramentas de pesquisa de vulnerabilidades Web de código aberto, quando utilizados em sua configuração padrão inicial.

Para isso, faremos um estudo bibliográfico sobre a história do termo hacker e suas variações de acordo com o nicho ao qual o especialista se vinculou.

A avaliação de desempenho das ferramentas se dará com o escaneamento de um ambiente previamente preparado com falhas mapeadas, utilizando a lista de dez falhas mais recorrentes na Web, disponibilizada pelo OWASP top 10.

1.3 Contribuições

Dados os pontos mencionados nas subseções anteriores, as principais contribuições deste trabalho são listadas a seguir.

- Classificar, de forma clara e objetiva, os tipos de hackers e diferenciá-los do profissional hacker ético;
- Mapeamento da necessidade do profissional hacker ético no mercado da tecnologia da informação, de forma direta e indireta;
- Estudo comparativo entre os aplicativos de pesquisa de vulnerabilidades Web de código aberto quando executados em suas configurações padrão.

1.4 Organização do documento

Este documento está organizado da seguinte maneira: o Capítulo 2 apresenta os conceitos principais da pesquisa, como: o que são hackers e como operam e o que são scanners de vulnerabilidades, bem como a fundamentação teórica deste trabalho. O Capítulo 3 reúne os detalhes sobre a metodologia dos testes, apresenta dados da OWASP sobre vulnerabilidades, suas execuções para testar sua efetividade e os resultados destas execuções. O Capítulo 4 apresenta as conclusões e considerações finais deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para que possamos estabelecer uma base inicial para esse trabalho, esse capítulo será dividido em três tópicos fundamentais. Começaremos traçando o perfil do hacker ético na Seção 2.1. Em seguida, mapearemos a crescente necessidade desse especialista no mercado na Seção 2.2. Na Seção subsequente, 2.3, faremos um breve resumo do processo de análise de vulnerabilidades. Na Seção 2.4 abordaremos com maior profundidade o *penetration test*. Para um aprofundamento mais organizado dos temas, a Seção 2.4 está dividida em três subseções: metodologia dos testes, tipos de teste e relatório final. Finalizamos na Seção 2.5 com a avaliação de desempenho das principais soluções, em código aberto, para mapeamento de vulnerabilidades em ambientes Web. Nas subseções da Seção 2.5 exploramos os detalhes da listagem das dez principais falhas em ambientes Web da OWASP.

2.1 Hacker Ético

Incidentes de falha de segurança cibernética deram origem a um tipo específico de profissional, que ficou conhecido como *hacker ético* (SMITH et al. 2001). Contudo, o termo hacker ético é, em verdade, um termo macro que engloba uma gama de especialidades vinculadas à disciplina inicial do especialista. Deste modo, torna-se impossível agrupar todos os ‘hackers’ em uma categoria única e bem definida. E com o crescimento do número destes incidentes, cresce também a necessidade de definir um perfil desse profissional especialista que irá identificar e corrigir essas lacunas de segurança (HAFELE, 2004). O hacker ético é um especialista em computadores e redes que ataca um sistema de segurança em nome de seus donos, procurando vulnerabilidades que atacantes maliciosos poderiam explorar (CHANDRIKA, 2014). Essa visão é corroborada por SMITH et al. (2001): hackers éticos encontram falhas e as corrigem, e neste processo testam o software para companhias em troca de acesso e informação. Como prestam um serviço de alta especialização e alto valor agregado, recebem pagamentos de acordo.

Como podemos observar, Hackers Éticos utilizam seus conhecimentos para aprimorar a segurança de sistemas uma vez descoberta uma vulnerabilidade. Eles não a exploram, mas sim a reportam aos mantenedores e usuários do sistema afetado, diferentemente de outros tipos de hackers, que utilizam seus conhecimentos para fins maliciosos.

Algo que precisa ser pontuado é que o hacker ético deve prover educação ao seu cliente (HAFELE, 2004). Ao final de um processo de mapeamento de vulnerabilidades, o profissional deve apresentar ao contratante um relatório detalhado sobre as falhas encontradas e as sugestões de correção. Observamos aqui que o profissional, em primeira ordem, não efetua correções, apenas informa o contratante sobre as falhas e possíveis correções (SMITH et al., 2001).

Algumas vezes o responsável pela TI ou pela segurança de uma empresa será informado que um ataque, chamado de teste de penetração (*penetration test*), será executado, e esses responsáveis podem até acompanhar o trabalho do hacker ético. Outras vezes, contudo, os profissionais não serão avisados, e o conhecimento de que um ataque será executado ficará acessível somente à alta gerência. Algumas vezes apenas duas ou três pessoas de alto escalão terão ciência do teste de penetração.

2.1.1 A necessidade da criação de um perfil

O'DONOHUE (2011) diz que a melhor forma de predizer um comportamento futuro é avaliar o comportamento passado, de modo que perfilar é entender como e por que alguém fez o que foi feito. LONG (2012), concorda com isso e acrescenta que um perfil é uma ferramenta psicológica que ajuda a entender a motivação de uma pessoa e pode ser usada em uma variedade de ocupações.

Em todos os casos, a criação de perfis estuda a motivação e metodologia para determinar se existe um padrão de comportamento que pode ser deduzido. Perfilar hackers é similar à criação de perfis de outras áreas. É preciso se manter atualizado em relação às mudanças tecnológicas, aos últimos ataques e às assinaturas do jeito de agir do cibercriminoso; assim, é possível antever não somente ataques planejados como também ataques do tipo *0day*. Deve-se conhecer a história do hacking de modo geral e de ataques famosos.

Se um grupo desconhecido ameaça um ataque a uma infraestrutura crítica aos EUA, por exemplo, os EUA sabiamente escutariam esse aviso, porém podem não dar total crédito, haja vista a baixa precedência histórica referente ao grupo desconhecido. Se, no entanto, forem hackers do governo chinês que façam essa ameaça, medidas protetivas serão tomadas por causa da prévia criação e conhecimento do perfil destes hackers (LONG, 2012).

2.1.2 Tipos de Hackers

Hoje a mídia utiliza os termos hacker ou cibercriminoso de uma forma mais ou menos intercambiável. Isso pode levar o leitor ao erro; enquanto seus significados se sobrepõem, eles não são sinônimos. Nesta Seção abordaremos uma lista não exaustiva dos subtipos de hackers existentes, focando nos três principais ao mercado: *white-hat*, *black-hat* e *gray-hat*.

O conceito de personagens usando chapéus brancos (*white-hat*) e pretos (*black-hat*) para designar a natureza de suas intenções, seja boa ou má, origina-se no cinema em filmes de faroeste. Podemos traçar seu primeiro uso em 1903, no filme "O grande roubo do trem", que, por ser um filme monocromático, precisou se apropriar da dualidade bem e mal de forma visual, facilmente identificável. Esse conceito foi posteriormente abraçado pela comunidade hacker para diferenciar os mocinhos dos bandidos de uma forma mais fácil para a população leiga em geral (CHANDRIKA, 2014).

2.1.2.1 White-hat *(chapéu branco)*

Um *white-hat*, também conhecido como hacker ético, é alguém que invade sistemas de segurança sem intuito malicioso. A grande maioria dos *white-hat* são especialistas em segurança e irão frequentemente trabalhar com uma empresa para detectar de forma legal as falhas de segurança e corrigir as que forem detectadas. Eles são comumente vistos como hackers que usam suas habilidades para beneficiar a sociedade. Deste modo, podem ser *black-hat* reformados ou simplesmente pessoas bem versadas nos métodos e técnicas utilizadas por hackers (CHANDRIKA, 2014).

Uma empresa preocupada com a segurança de seus ativos de informática pode contratar um hacker ético como consultor para efetuar testes em sua rede e aplicações e também ajudá-la, por meio da instrução de suas equipes nos quesitos de segurança, a implementar melhores práticas que a faça ser menos vulnerável a futuras tentativas maliciosas de hackers.

2.1.2.2 Black-hat *(chapéu preto)*

Um *black-hat*, também chamado de cracker em algumas bibliografias, é alguém que, no lado oposto de um hacker ético, tenta invadir um sistema com motivação maliciosa e sem autorização do proprietário daquela infraestrutura. Tipicamente esse hacker quer provar suas habilidades e irá cometer uma serie de cibercrimes, como roubo de identidade, fraude em cartões de crédito e pirataria (CHANDRIKA,2014).

Os *black-hats* são também conhecidos como *crackers* (do inglês *to crack*, rachar/quebrar) porque, enquanto *white-hats* criam coisas, os *black-hats* as *quebram*. São também conhecidos por serem os criadores de grande parte dos vírus que infestam as redes de computadores (CHANDRIKA, 2014). Em uma apertada síntese, podemos colocar tanto o *black* quanto o *white-hat* como o mesmo tipo de especialista em computadores, e a palavra-chave que os diferencia é *autorização*.

2.1.2.3 Gray-hat (chapéu cinza)

Como a cor do *chapéu* sugere, um *gray-hat* é alguém entre o *black* e o *white*, e essa pessoa exibe traços de ambos. Por exemplo, um *gray-hat* irá vasculhar a internet atrás de uma falha, e como um *white-hat* ele irá informar a empresa que possui essa falha, ao mesmo tempo que, como um *black-hat*, fez toda essa pesquisa sem autorização dessa empresa.

O termo *gray-hat* foi cunhado pelo grupo hacker *L0pht* em 1998, e foi originalmente usado para descrever hackers que reportavam as vulnerabilidades que encontravam em organizações que invadiam. Depois, em 2002, a comunidade de segurança *Anti-Sec* usou o termo para descrever aqueles que trabalhavam na comunidade de segurança durante o dia e agiam como *black-hats* durante seus períodos de folga (CHANDRIKA, 2014).

2.1.2.4 Outros tipos de hackers

Existem outros grupos de hackers que citaremos brevemente, pois não estão no escopo do nosso trabalho.

- *Blue-hats*: funcionários de empresas de consultoria de segurança contratados para testar um sistema atrás de falhas antes do seu lançamento no mercado. Eles também são associados a uma conferência anual comandada pela Microsoft, na qual engenheiros da Microsoft e hackers podem conversar abertamente sobre falhas e brechas de segurança.
- Hackers de elite: esse tipo de hacker tem a reputação de ser o melhor de sua área. Os hackers de elite são frequentemente considerados inovadores e especialistas. Foi esse grupo que criou a hoje conhecida *Leetspeak* para esconder seus sites dos motores de busca rudimentares de anos atrás. Pessoas desse grupo são chamadas assim pela comunidade de segurança porque estão, de fato, no ápice de ambas as indústrias, a de computadores e a de redes (CHANDRIKA, 2014).
- Hacktivistas: São hackers que invadem uma rede de computadores por motivação política ou social. Esse é um termo controverso, pois, para que um hacktivista se torne um terrorista basta que suas atividades deixem o campo da computação (não violento) para atividades violentas. O termo foi cunhado em 1996 pelo grupo chamado "The cult of the dead cow". Eles não podem ser colocados no mesmo grupo de *black-hats* ou *white-hats*, pois sua motivação é política e não o lucro (CHANDRIKA, 2014). Um dos grupos mais conhecidos recentemente é o "Anonymous", que em outubro de 2011 derrubou 40 sites de pornografia infantil e tornou público o nome de mais de 1500 pessoas que frequentavam esses sites.
- Ciberterroristas: Diferentemente dos hacktivistas, este grupo tem como motivação crenças religiosas ou políticas, e tentam criar medo e caos ao derrubar infraestruturas críticas para civilização (eletricidade, fornecimento de água, gerenciamento de semáforos etc.). O ciberterrorismo pode ser definido como o uso intencional de computadores, redes e outras redes públicas de internet para causar destruição e perigo por objetivos pessoais. De acordo com o FBI, ciberterrorismo é qualquer ataque premeditado e politicamente motivado, contra sistemas de informações, programas de computadores e dados que possam resultar em violência contra alvos não combatentes por grupos sub-nacionais ou agentes clandestinos (CHANDRIKA, 2014).
- Amadores/*Script kiddies*: Estes são aqueles que seguem manuais e usam scripts e códigos de terminal prontos, disponibilizados por outros hackers, sem ter o total conhecimento de cada passo tomado durante um ataque.

- Espiões: Grandes corporações recrutam hackers para se infiltrarem nas redes de seus competidores para roubar segredos industriais. Suas ferramentas são principalmente "cavalos de troia" e *keyloggers*, usadas de modo que possam observar o comportamento de seus alvos dentro da organização atacada.

2.2 A necessidade de profissionais especialistas

Com o crescimento populacional, crescem proporcionalmente os crimes, dentre eles os crimes cibernéticos. Neste cenário, a atividade do hacker ético se tornou uma poderosa estratégia no combate de ameaças online e, portanto, uma necessidade. HAFELE (2004) concorda, dizendo que o Hacking Ético é uma ferramenta que, se utilizada corretamente, prova-se útil para compreender as fraquezas de uma rede e como elas podem ser exploradas e, deste modo, combatidas. Essa premissa é corroborada por PATIL et al., (2017) quando os pesquisadores sustentam que a segurança no ciberespaço está crescendo e os atacantes estão munidos de ferramentas cada vez mais sofisticadas, e que a defesa deste espaço precisa se equalizar ou suplantar o conhecimento destes criminosos.

SINGER (2014) propõe que a cada dia mais 2.500 Petabytes são adicionados ao depósito global de informações digitais. Neste cenário, o setor da tecnologia da informação se encontra confrontado tanto pelo crescimento de suas redes de comunicação quanto pela necessidade crescente que essa comunicação se dê de forma segura, de modo que um profissional que responda pelos meios de segurança dessa massa de informação se torna cada vez mais necessário. Esse crescimento das taxas de dados trafegados é, em grande parte, decorrência do aumento da utilização do ciberespaço pela população para conduzir atividades do dia a dia. Os hackers maliciosos têm usado toda sorte de técnicas inovadoras para atingir seus objetivos. RAO et al. (2014) concordam com isso ao sustentar que, se o ataque está focado em roubar segredos, interromper sistemas ou algo pior, a ameaça é real.

Outros escopos que necessitam de um profissional qualificado para lidar com as circunstâncias da segurança cibernética emergem deste cenário inicial, como, por exemplo, a educação, tanto academicamente quanto dentro das corporações, versando sobre padrões seguros de desenvolvimento de aplicações. Nesse cenário da instrução corporativa, ou da falta dela, vemos muitos casos de vazamentos de dados ocorrendo. HAFELE (2004) cita que, virtualmente, todos os dias podemos tanto ler notícias como ver referências na Web sobre companhias e organizações sofrendo ataques contra suas redes. Essa visão é aceita por SMITH

et al. (2001) quando os pesquisadores afirmam que o nível da segurança na internet de modo geral é fraco e sua qualidade cresce de forma lenta.

Neste cenário de vazamentos de dados críticos podemos citar como exemplo o já infame incidente de vazamento maciço de dados críticos de usuários das redes de hotéis Marriott, geridos pelo grupo Starwood Hotels and Resorts (STARWOOD, 2018). Este vazamento, hoje considerado o segundo maior na história da tecnologia da informação, afetou mais de 500 milhões de clientes. De acordo com informações do *International Business Times*, o impacto nos ativos do grupo administrador dos hotéis já atinge quase 9% de perda nas ações da empresa, e já há ações coletivas, com pedidos de perdas e danos na ordem de 12,5 bilhões de dólares, mapeando essa falha de segurança como existente nos sistemas desde 2014 (MAJUMDER, 2018; STARWOOD, 2018). Neste cenário solidificamos a necessidade crescente por especialistas em segurança cibernética.

2.3 A análise de vulnerabilidades

O teste por falhas de segurança aparenta ser uma atração natural para muitos entusiastas: é desafiador, e expor vulnerabilidades de modo que elas sejam corrigidas contribui para o bem público. Testes manuais evoluíram para automatizações em programas para mapeamento de vulnerabilidades conhecidas. WANG e YANG (2017) afirmam que a verificação de vulnerabilidades é o processo de utilizar um computador para procurar falhas em outro computador. Ferramentas de mapeamento de vulnerabilidades normalmente são utilizadas em estações de trabalho próprias para testar outros computadores e outras redes conectadas a eles.

Antes de explorarmos as diferentes ferramentas que estão disponíveis, é necessário entendermos um pouco melhor sobre os conceitos básicos de vulnerabilidade de segurança e de testes de penetração. Vulnerabilidade em segurança cibernética é uma falha, seja de lógica durante a composição do sistema, seja de arquitetura que permite que um intruso explore e reduza a confiabilidade do sistema (WANG; YANG, 2017). Testes de penetração, por definição, são um grupo de testes utilizando um detector com o objetivo de mapear as vulnerabilidades de um sistema, tendo a clareza de que não há sistema 100% seguro (JIMENEZ, 2016).

Em uma simplificação, podemos dizer que um aplicativo de mapeamento de vulnerabilidades é composto por três módulos principais: um módulo de *crawler*, um módulo de ataque e um módulo de análise (DOUPÉ et al., 2010). O módulo *crawler* (rastejador em

inglês) tem por objetivo mapear todos os arquivos acessíveis em um endereço de rede. O módulo de ataque utiliza os endereços de arquivos descobertos pelo *crawler* para efetuar tentativas de ataques já mapeados para os arquivos/endereços correspondentes. Por fim, o módulo de análise sintetiza o resultado das tentativas de ataque em formato de relatório.

WANG e YANG (2017) e DOUPÉ et al. (2010) utilizaram métodos parecidos para suas avaliações de ferramentas de mapeamento de vulnerabilidades. As falhas testadas como métrica foram as listadas no top 10 da OWASP. Utilizaremos essa mesma métrica neste trabalho por ela ter se mostrado eficaz, já que a lista da OWASP é reconhecida como a mais verossímil em relação às vulnerabilidades mais críticas em ambientes Web.

2.4 *Penetration test*

Penetration test (também chamado de *pentest* ou *teste de penetração*) é uma técnica em que são utilizadas ferramentas e expertises de hackers para fazer análise de sistemas ou redes em busca de falhas de segurança. A verdadeira definição é que um *pentest* é um grupo de testes com o objetivo de detectar vulnerabilidades de um sistema, tendo em mente de forma clara que nenhum sistema é 100% seguro e inviolável (GUIRADO, 2009).

Um *pentest* exige um planejamento cuidadoso do escopo do que será testado, assim como a duração do processo e os itens contidos no relatório final. No início do planejamento, o hacker ético irá recolher as informações necessárias para a execução do teste. Destas informações podemos listar: o escopo do projeto, seu objetivo, duração, tarefas a serem realizadas, custo do serviço, forma de pagamento e as metas que devem ser alcançadas.

ASSUNÇÃO (2014) afirma que o cronograma de realização dos testes varia de acordo com três fatores: a quantidade de tarefas, a quantidade de equipamentos e o número de horas demandadas para realização do teste. JIMÉNEZ (2016) concorda com essa divisão e credita um tempo médio de duas a quatro semanas para a execução de cada uma destas fases, o que resulta num período médio de seis a doze semanas para a execução total de um teste do tipo *black box*, que é a proposta deste trabalho.

2.4.1 Metodologia de *penetration test*

Assim como ocorre em outras áreas da TI, diversas entidades desenvolvem normas para uma maior padronização das tarefas a serem realizadas em um *pentest*. Estes padrões são importantes, pois facilitam a organização de um processo de análise de vulnerabilidades em larga escala. Os padrões de teste referentes à segurança cibernética mais populares são o OSSTMM *(Open Source Security Testing Methodology Manual*) desenvolvido pela ISECOM (*Institute for Security and Open Methodologies*), o TGISTA (*Technical Guide to Information Security Testing and Assessment*), proposto como recomendação pelo NIST (*National Institute of Standards and Technology*), e o ISSAF (*Information Systems Security Assessment Open Methodologies*), proposto pelo OISSG (*Open Information Systems Security Group*).

O OSSTMM é a metodologia mais popular para a padronização de testes de penetração. A documentação é muito rica e detalhada e os testes padronizados são explicados com um alto nível de detalhes e incluem também a análise de diversos tipos de tecnologias de redes sem fio, como o *bluetooth*, infravermelho e WIFI. A documentação do OSSTMM (ISECOM, 2015) não é vinculada às ferramentas, portanto, as técnicas sugeridas podem ser utilizadas com qualquer ferramenta de análise de vulnerabilidades compatível com os testes propostos. Esta é uma das características de dupla mão do OSSTMM, e essa flexibilidade vem ao custo da curva de aprendizado ser maior.

O ISSAF é outro manual de padronização desenvolvido para a realização de auditoria de segurança de sistemas. Ele é fortemente baseado em listas de checagem, mas também aborda técnicas de análise de vulnerabilidades. A documentação do ISSAF é dividida em duas partes: Gerenciamento geral do processo (ISSAF 0.2.1A) e *Penetration testing* (ISSAF 0.2.1B). Uma característica do ISSAF, em contrapartida ao OSSTMM, é o fato dele não exigir conhecimentos prévios de ferramentas e sistema operacional para ser utilizado. A segunda parte do manual ISSAF aborda de forma detalhada, com imagens e exemplos, o uso das ferramentas necessárias para a realização dos testes. A documentação do ISSAF também propõe a divisão de um teste de penetração em cinco estágios sequenciais: planejamento, análise, tratamento, resultados e manutenção.

2.4.2 Tipos de teste

STUART et al. (2014) concordam com JIMÉNEZ (2016) e HAFELE (2004) quando afirmam que a definição do tipo de teste realizado afeta diretamente o resultado do processo de avaliação. Portanto, é muito importante que se entenda e escolha o teste correto para cada avaliação. Tanto o OSTMM quanto o ISSAF definem três tipos possíveis de *pentest*: os testes de caixa preta (*black box*), caixa branca (*white box*) e caixa cinza (*gray box*). Esses testes apresentam similaridades entre si, contudo, possuem suas particularidades quanto ao escopo e objetivo final.

JIMÉNEZ (2016) explica o funcionamento de um teste do tipo *black-box* como um teste no qual o avaliador não possui nenhum conhecimento do sistema que ele está disposto a testar. Por exemplo, neste teste, o avaliador somente sabe o que espera de saída do teste, mas não possui nenhuma ideia de como será o formato desta saída. Ele não examina nenhum código de programação. HAFELE (2004) complementa dizendo que, geralmente, o intuito do teste de invasão do tipo *black-box* é simular um ataque hacker por alguém externo à empresa, ou seja, alguém que realmente não tenha um acesso fácil às informações da mesma. É um teste "às cegas" em relação aos dados fornecidos antecipadamente, mas que pode identificar as vulnerabilidades que poderiam facilitar esse tipo de invasão.

Já os testes do tipo *white-box* são compostos por uma abordagem mais aberta, pois é executado com anuência do proprietário do sistema auditado. HAFELE (2004) nos ensina que um plano de ataque do tipo *white-box* é mais determinístico do que um do tipo *black-box*; isso quer dizer que um time de hackers éticos terá muito mais informação divulgada para eles sobre o sistema antes do teste ocorrer. JIMÉNEZ (2016) vai além ao afirmar que o *white-box* é um teste considerado como uma simulação de ataque de uma fonte interna, pois o atacante é totalmente ciente das especificidades do sistema alvo, tais como código fonte, sistema operacional, endereços IP etc.

Como o nome nos sugere, um teste *gray-box* é uma amálgama dos testes *white-box* e *black-box*, no qual o testador possui informações parciais dos detalhes internos do sistema alvo. HAFELE (2004) classifica esse tipo de teste como um ataque por um hacker externo que tenha conseguido acesso ilegítimo a documentos descritivos da infraestrutura alvo.

2.4.3 Relatório final

Após a conclusão da análise das vulnerabilidades, é necessário reunir e organizar todas as informações obtidas durante o processo, como os tipos de problemas encontrados e o nível de gravidade de cada um. Em geral as falhas são classificadas em três categorias: baixo, médio e alto risco (DOUPÉ et al., 2010). Essas informações devem estar contidas no relatório final, que é um documento composto pelas informações e falhas encontradas durante a execução do *pentest*. Deve seguir um modelo claro, preferencialmente seguindo tópicos, para que não existam informações confusas. ASSUNÇÃO (2014) recomenda que se demonstrem quatro itens essenciais no relatório: introdução, detalhes do trabalho, resultados obtidos e recomendações ao cliente.

Além do relatório final, HAFELE (2004) e CHANDRIKA (2014) sinalizam a necessidade de um termo de responsabilidade, que deverá ser firmado entre o prestador do serviço de testes e a empresa contratante. ASSUNÇÃO (2014) concorda com a necessidade deste documento, de forma a proteger legalmente o hacker ético, uma vez que um teste de penetração pode levar ao comprometimento de informações sigilosas, e no caso de ambientes Web, o possível comprometimento do sistema.

2.5 Vulnerabilidades em ambientes Web

BISHOP (2007) nos diz que as vulnerabilidades de uma aplicação Web podem ser reduzidas com a devida instrução dos desenvolvedores e/ou pelo uso correto de ferramentas que permitam a realização de testes de segurança. Essa visão é corroborada por HAFELE (2004) e por DOUPÉ et al. (2010), que frisam a necessidade de profissionalização dos procedimentos de avaliação de vulnerabilidades por parte dos desenvolvedores dos aplicativos.

2.5.1 O projeto OWASP

O projeto OWASP (*Open Web Application Security Project*) é uma organização internacional e uma comunidade aberta dedicada a permitir que as organizações concebam, desenvolvam, adquiram, operem e mantenham aplicativos confiáveis. Todas as ferramentas, documentos, fóruns e capítulos do OWASP são gratuitos e abertos a qualquer pessoa

interessada em melhorar a segurança das aplicações. SILVA et al. (2014) nos dizem que o OWASP não é afiliada com nenhuma empresa de tecnologia, portanto, as informações de segurança que eles geram em seus relatórios são imparciais e práticas. OLIVEIRA (2012) corrobora esta visão e acrescenta que os projetos do OWASP são divididos em duas categorias: a de desenvolvimento seguro e a de documentação dos riscos. Entre os projetos desenvolvidos, podemos elencar como os principais para a pesquisa proposta neste trabalho.

- *Development Guide* – é o projeto de documentação que foca na segurança durante o desenvolvimento de aplicações Web. Ele mostra vários aspectos pertinentes à segurança das soluções;

- *Testing Guide* – é um guia prático que norteia as seguintes questões: o que testar, por que testar, quando, onde e como testar as aplicações Web em busca de vulnerabilidades;

- *Code Review Guide* – é um guia que mostra boas práticas para a revisão segura de códigos com o objetivo de encontrar vulnerabilidades. É um pouco mais extenso e técnico que o *Testing Guide*;

- *Top Ten* – é o guia mais popular do OWASP. O *top ten* é basicamente um documento que referencia e explica as dez principais vulnerabilidades das aplicações Web. Mostra as consequências da exploração de cada uma dessas falhas, assim como algumas técnicas e/ou sugestões para mitigar estes problemas.

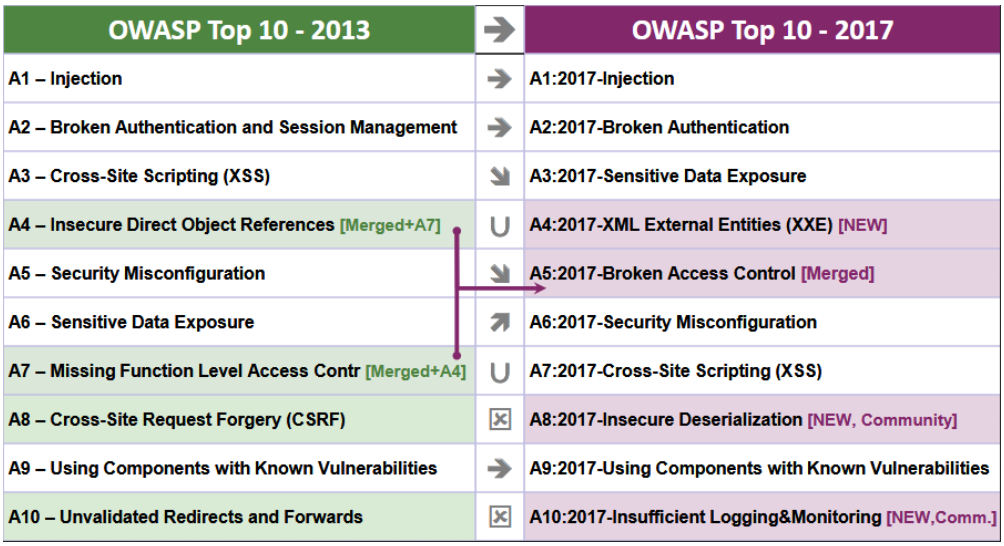
2.5.2 As dez principais vulnerabilidades mapeadas pelo OWASP *Top Ten*

A lista das dez falhas mais recorrentes da OWASP é um relatório regularmente atualizado que demarca as principais preocupações de segurança nas aplicações Web, focando nas dez com riscos mais críticos. O relatório é referenciado por um time de especialistas do mundo todo. O OWASP descreve o *top ten* como um "documento de conscientização" e recomendam que todas as empresas incorporem o relatório como um norte para seus processos, de modo a minimizar e/ou mitigar possíveis riscos de segurança. Na Figura 2.1 há uma comparação da versão 2013 com a versão 2017 da lista *top ten.*

De acordo com o OWASP, a maioria das vulnerabilidades ocorre durante o desenvolvimento da aplicação. SMITH et al. (2001) concordam com isso ao afirmar que a segurança da internet é um problema complexo que precisa de incentivos fortes para os

desenvolvedores serem treinados para fazer a coisa certa (aplicar os padrões de desenvolvimento). Essa visão é confirmada pelos estudos de IBRAHIM e TRABELSI (2013) quando citam que atualmente há uma falta visível de textos, livros etc. que descrevam a implementação de técnicas de desenvolvimento seguro que sejam práticas e não apenas factíveis em ambientes isolados de laboratório.

Figura 2.1 - Comparativo entre a lista de 2013 e 2017.



Fonte: <https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf>

Percebe-se que algumas vulnerabilidades mudaram de posição na classificação, outras deixaram de existir, sendo substituídas por novas classificações, e outras foram fundidas em uma nova classificação mais ampla. Na pesquisa utilizaremos a versão 2017, que é a mais atual disponibilizada pelo OWASP. Tendo como referência essa versão, discutiremos os principais riscos a seguir.

A1-Injection

KUMAR e PATERIYA (2012) demonstram que há quatro categorias principais quando se fala da falha por injeção: Manipulação de SQL, Injeção de código, Injeção de chamada de Funções e *Buffer Overflow*.

- Manipulação de SQL: é o processo de modificar uma declaração SQL se aproveitando das operações disponíveis no banco de dados, como *UNION*, por exemplo. Outra forma

de implementar uma injeção de SQL é por meio da manipulação de como uma cláusula *where* é criada na declaração SQL, de modo a gerar um resultado diferente.

- Injeção de código: é o processo de inserir uma nova declaração SQL. Uma das formas de ataque conhecido é de adicionar um comando SQL Server *EXECUTE* à declaração SQL comprometida.
- Injeção de chamada de função: é o processo de inserir várias chamadas de função de banco de dados dentro de uma declaração SQL comprometida, de modo que a função possa executar chamadas no sistema operacional ou efetuar alterações na base de dados.
- *Buffer Overflow*: esta vulnerabilidade é causada pelo uso da injeção de chamadas de funções; a grande maioria das bases de dados disponíveis no mercado já possui patch de correção disponível. Este tipo de ataque é possível quando o servidor não está atualizado.

A2-Broken Authentication

AL-KHURAFI e AL-AHMAD (2015) expõem que a causa raiz de um sistema de autenticação estar vulnerável está relacionada com a implementação insegura por parte dos desenvolvedores. Isso acontece quando estes criam versões customizadas de processos de autenticação que possuem falhas de implementação, que acarretam no comprometimento de senhas, chaves criptográficas e *tokens* de sessão, ou que exploram outras falhas de implementação, de modo a permitir que o atacante assuma a identidade de outros usuários de forma temporária ou permanente.

A3-Sensitive Data Exposure

Muitas aplicações Web e APIs não protegem seus dados sensíveis de forma apropriada, como dados financeiros, de saúde etc. Atacantes podem roubar ou modificar estes dados fragilizados de modo a conduzir uma fraude financeira, ou um roubo de identidade, ou outros crimes. Dados sensíveis podem ser comprometidos quando não possuem uma proteção extra, como a criptografia dos dados armazenados ou em transferência, e requerem precauções especiais quando trocadas via browser.

SHU, YAO e BERTINO (2015) concordam com as formas de comprometimento ao afirmarem que a detecção e prevenção de vazamento de dados requer um grupo de soluções

complementares, que podem incluir detectores de vazamento de dados, sistemas de confinamento dos dados, detectores escondidos de *malwares* e a aplicação de políticas de segurança.

A4-XML External Entities (XXE)

Muitos processadores de arquivos XML desatualizados ou mal configurados processam entidades externas internamente em seus documentos XML. Entidades externas podem ser usadas para tornar visíveis arquivos internos usando o *handler* do arquivo, compartilhamentos de arquivos interno, scanners de portas, execução de código remoto, e ataques de negação de serviço.

JAN, NGUYEN e BRIAND (2015) explicam que, se o processador de XML não estiver configurado para prevenir ou limitar entidades externas, pode ser forçado a acessar recursos especificados pela URI. Uma exploração bem-sucedida desta vulnerabilidade pode resultar em um vazamento de dados, uma negação de serviço, ou na obtenção de acesso não autorizado aos recursos do sistema.

A5-Broken Access Control

Restrições sobre o que um usuário autenticado pode ou não fazer devem ser reforçadas e implementadas apropriadamente. Atacantes podem explorar essas falhas para acessar funcionalidades e/ou dados não autorizados (como outras contas de usuários), visualizar arquivos sensíveis, modificar dados de usuário, modificar direitos de acesso etc.

A6-Security Misconfiguration

Recursos de segurança mal configurados são os problemas mais comumente vistos. Isso normalmente é resultado da utilização das configurações padrão dos aplicativos, configurações incompletas ou configurações ad hoc, áreas de armazenamento na nuvem que estão abertas, cabeçalhos HTTP mal configurados e mensagens de erro com informações sensíveis. Não apenas todos os sistemas operacionais, como frameworks, bibliotecas e aplicações devem ser configurados apropriadamente, como também devem ser mantidos atualizados com seus devidos patches/upgrades quando estes existirem.

A7-Cross-Site Scripting (XSS)

Falhas de XSS ocorrem quando uma aplicação requisita dados não confiáveis ou inclui estes dados não confiáveis em uma página Web sem a validação apropriada, ou sanitização, ou atualiza uma página existente com dados fornecidos pelo usuário usando a API de um navegador que consiga criar código HTML ou Javascript. XSS possibilita ao atacante executar scripts no navegador da vítima, o que pode ser usado para roubar uma sessão de usuário, redirecionar a sites falsos ou alterar a visualização do site (defacement attacks).

SHRIVASTAVA, CHOUDHARY e KUMAR (2016) concordam e afirmam que existe três tipos de ataques XSS: persistido ou armazenado, não persistente ou refletido e baseado em DOM. O ataque persistido é efetuado no momento que os dados são armazenado na base de dados; o não persistente e o DOM requerem que o usuário clique em um link malicioso ou visite um site malicioso, de modo que a requisição HTTP, automaticamente e sem conhecimento do usuário, envia uma requisição do tipo POST com código vulnerável que será executado novamente e mostrado na tela do usuário.

A8-Insecure Deserialization

DEHALWAR et al., (2017) explicam que serialização é um processo de gravar os dados como uma cadeia ou como um binário bruto para a transferência de dados através de uma rede. Por outro lado, a desserialização é o processo de recuperar os dados brutos de um arquivo ou soquete de rede para reconstruir o modelo de objeto. Os dados não confiáveis não podem ser desserializados sem verificar suficientemente se os dados resultantes são válidos/genuínos. Os dados/objetos serializados podem ser convenientemente usados, mas os dados não sanitizados podem ser modificados por um invasor se não estiverem protegidos pela função criptográfica.

A9-Using Components with Known Vulnerabilities

Componentes como bibliotecas, frameworks e outros módulos de terceiros são executados com os mesmos privilégios da aplicação que os consome. Se um componente vulnerável é explorado, o ataque pode ser usado para facilitar severamente a perda de dados, ou até mesmo a perda de controle de um servidor. Aplicações e APIs usando componentes com

vulnerabilidades conhecidas podem minar as defesas da aplicação e permitir vários tipos de ataques e impactos no negócio.

A10-Insufficient Logging & Monitoring

Registros, traços e monitoramento insuficientes, junto com integração ineficiente com o time de respostas a incidentes, permitem que agentes maliciosos ataquem os sistemas, mantenham ataques persistentes, utilizem o sistema como alavanca para ataques a outros sistemas e adulterem, extraiam ou mesmo excluam de dados. A maioria das brechas estudadas mostram que o tempo de detecção da brecha é de mais de 200 dias, tipicamente detectada por terceiros, externos ao sistema.

2.5.3 Ferramentas de análise de vulnerabilidades Web

Para detectar essas e outras falhas presentes em aplicações Web, existem hoje no mercado dezenas de ferramentas. Contudo, existem alguns impeditivos em relação aos seus usos, como a forma de acesso, o licenciamento e escopo/funcionalidades presentes na aplicação. Como posto por VIERA, ANTUNES e MADEIRA (2009), algumas destas ferramentas pagas possuem suas versões gratuitas disponíveis para testes, porém ao custo de terem limitações de recursos ou prazo de uso limitado, e o licenciamento destas ferramentas comerciais costumam ter valores que vão de algumas dezenas a milhares de dólares, muitas vezes com um custo muito elevado em comparação aos benefícios disponibilizados.

De acordo com ASSUNÇÃO (2014), um dos fatores relevantes na escolha de uma ferramenta de análise de vulnerabilidades em ambientes Web é a quantidade de falsos positivos e de falsos negativos gerados após uma varredura ser finalizada. Um falso positivo ocorre quando a falha detectada não existe de fato no sistema, mas o scanner detecta erroneamente. Já o falso negativo é o contrário, quando uma falha existe e não é detectada; consequentemente, é muito mais preocupante. Outro fator a ser considerado é a implementação de um perfil de testes preparados para verificação das falhas citadas no *OWASP Top Ten*. Como esta lista é periodicamente revista pela entidade devido às descobertas e pesquisas de cibersegurança acontecendo no mundo, a ferramenta de análise deve oferecer atualizações frequentes em sua relação de falhas detectáveis, de modo a se manter uma ferramenta relevante ao longo do tempo.

Alguns sites mantêm rankings de popularidade de ferramentas de segurança cibernética. Destes, podemos citar o *SecTools.org*, o *csoonline.com* e o *WebAppSec* (Wasc). Nesta pesquisa utilizaremos o ranking do site *SecTools.org* e o *WebAppSec* como fonte das ferramentas de teste mais populares, por serem citados como fonte pelos especialistas da área (MARTINELO; BELLEZI, 2014).

3 METODOLOGIA DE PESQUISA

Em relação ao desenvolvimento da pesquisa, ela tem caráter bibliográfico no que se refere à pesquisa do perfilamento do profissional hacker ético, e caráter experimental no que se refere ao teste de desempenho das ferramentas de análise de vulnerabilidades para conhecer a eficiência de cada uma das cinco ferramentas selecionadas na detecção das dez principais falhas da lista *OWASP Top Ten*, na versão 2017 do documento.

Na Seção 3.1, apresentamos a metodologia proposta para análise das ferramentas que serão testadas. Na Seção 3.2, listamos as características e os procedimentos para construção do ambiente simulado que será usado para o teste das ferramentas. Na Seção 3.3 exporemos o método de seleção das ferramentas que serão alvo do teste proposto neste trabalho, e nas subseções do item 3.3 serão apresentadas as ferramentas selecionadas, seguidas de breve descrição e de suas principais características.

3.1 Metodologia proposta para análise das ferramentas

A lista *OWASP Top Ten*, periodicamente atualizada, apresenta os dez riscos de segurança em sistemas Web mais frequentemente mapeados em uma aplicação Web. De acordo com CARVALHO (2014), a lista de vulnerabilidades disponibilizada pela *OWASP Top Ten* é desenvolvida e atualizada a partir de consultas feitas a especialistas em segurança da informação e organizações de portes e ramos diversos, em vários países, tanto na esfera privada como na esfera governamental. Por esta razão, sendo ela considerada uma referência na área de Segurança da Informação, decidiu-se utilizar essa lista como base para o experimento de análise de falhas.

Os modelos de teste foram selecionados inicialmente utilizando os estudos de WANG e YANG (2017), que fizeram uma avaliação de algumas ferramentas de verificação de vulnerabilidades em sistemas Web com foco no aprendizado em sala de aula. Estes conceitos de testes práticos e de ferramentas de teste são corroborados pelos estudos de JIMÉNEZ (2016), que fez um estudo focado nos conceitos das técnicas de *pentest* para sistemas Web, suas fases e os ataques mais comuns à época do estudo. A parte experimental deste trabalho se baseou nas pesquisas de WANG e YANG (2017) e JIMÉNEZ (2014), que conduziram experimentos com diferentes ferramentas em diferentes cenários. Diferenciamo-nos destes estudos por tratarmos o uso destas aplicações como usuários finais, sem conhecimento técnico sobre customizações,

de modo a verificar a real capacidade de detecção destes softwares quando estes forem executados em sua configuração padrão.

Durante o experimento foi utilizado o framework de testes *Mutillidae*, além das ferramentas de teste selecionadas. Este framework permite simular as vulnerabilidades descritas no documento *OWASP Top Ten* para fins de teste. O *Mutillidae* é um projeto da própria OWASP, que fornece um site completo com diversas páginas propositalmente vulneráveis com o propósito de demonstrar estas vulnerabilidades em um ambiente realista. Foram coletados os seguintes dados sobre as ferramentas, durante sua execução: quantidade de páginas descobertas automaticamente, as falhas detectadas e o tempo levado para detecção. Para fins da pesquisa, será considerada a ferramenta com melhor acurácia, aquela ferramenta que identificar o maior número de vulnerabilidades no ambiente, descartando-se os falsos positivos.

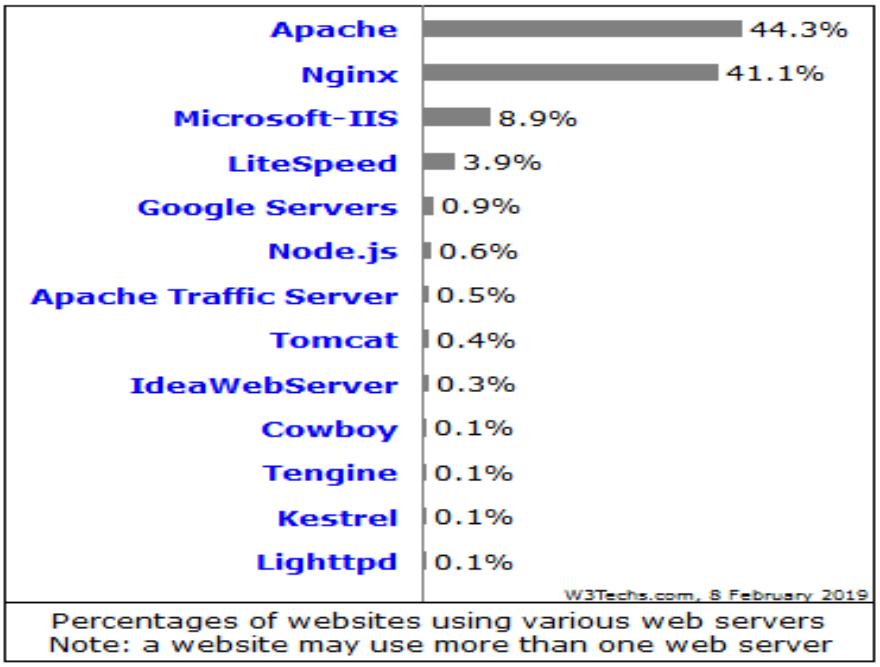
As análises dos resultados obtidos na etapa de testes experimentais da pesquisa verificaram as páginas evidenciadas automaticamente pelas ferramentas (através do processo de *crawling*), o número de vulnerabilidades detectadas e o tempo total de varredura gasto por ferramenta. Dos dados coletados, será considerada a relação entre as falhas corretamente detectadas e as falhas existentes como métrica para criação do ranking final. Os falsos positivos e falsos negativos serão apresentados apenas para fins de tabulação dos resultados, mas suas proporções não serão consideradas para fins de estabelecimento do ranking. Os dados obtidos foram tabulados e comparados, de modo a permitir que se responda à questão inicial e motivadora da pesquisa, que é mensurar a eficácia dos scanners de vulnerabilidades de código aberto quando executadas em suas configurações iniciais (sem refinamento).

3.2 Construção do ambiente simulado

O sistema *Mutillidae* é um ambiente de teste e/ou treinamento para *pentest* fornecido pelo OWASP que conta mais de 40 vulnerabilidades documentadas. De modo a encorajar a melhoria dos processos de segurança e de treinamento de desenvolvedores, os criadores deste sistema disponibilizam uma série de manuais, vídeos de suporte e funcionalidade de resets da base de dados e tutoriais. O OWASP possui outros sistemas com falhas mapeadas disponíveis para uso da comunidade externa, cada um com foco em propósitos distintos. Entre os disponíveis podemos citar o *Juice Shop* como uma ótima ferramenta para ser utilizada em aulas de cibersegurança por possuir um sistema de métricas e um formato de *gamificação* no qual o sistema deve ser invadido e o invasor adquire pontos de cumprimento de tarefas.

O *Mutillidae* foi escolhido por melhor se adequar à necessidade da pesquisa, já que possui uma vasta documentação, é de fácil customização caso necessário e é construído dentro das vulnerabilidades mapeadas no *Top Ten*. Para a construção do ambiente de realização dos testes, uma máquina virtual foi preparada com todos os recursos necessários para a utilização do *Mutillidae.* Para que o teste fosse mais próximo da realidade, efetuamos pesquisas de quais linguagens e servidores de aplicações eram os mais utilizados pelo mercado no momento da elaboração deste trabalho. Apesar do crescimento do uso do *Nginx*, o servidor de dados mais utilizado ainda é o Apache, como demonstrado pela pesquisa efetuada pela *W3Techs*. A Figura 3.1 nos mostra um gráfico comparativo de popularidade dos servidores Web, de modo que o Apache foi escolhido como servidor de aplicação Web para o teste proposto.

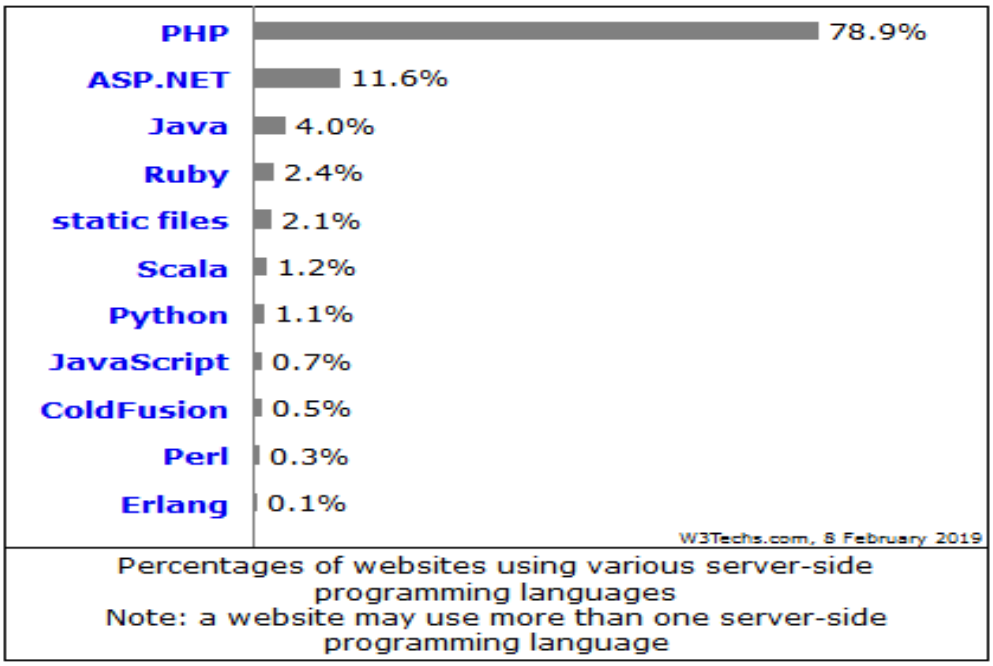
Figura 3.1 - Comparativo de popularidade ente servidores



Fonte: https://w3techs.com/technologies/overview/Web_server/all

Sobre as linguagens de programação mais utilizadas no mercado, a W3Techs efetuou pesquisa sobre a popularidade delas quando utilizadas para criação de sites e aplicações Web. Na Figura 3.2 podemos acompanhar o gráfico demonstrando esses dados, de modo que a linguagem PHP foi selecionada para uso do sistema com falhas.

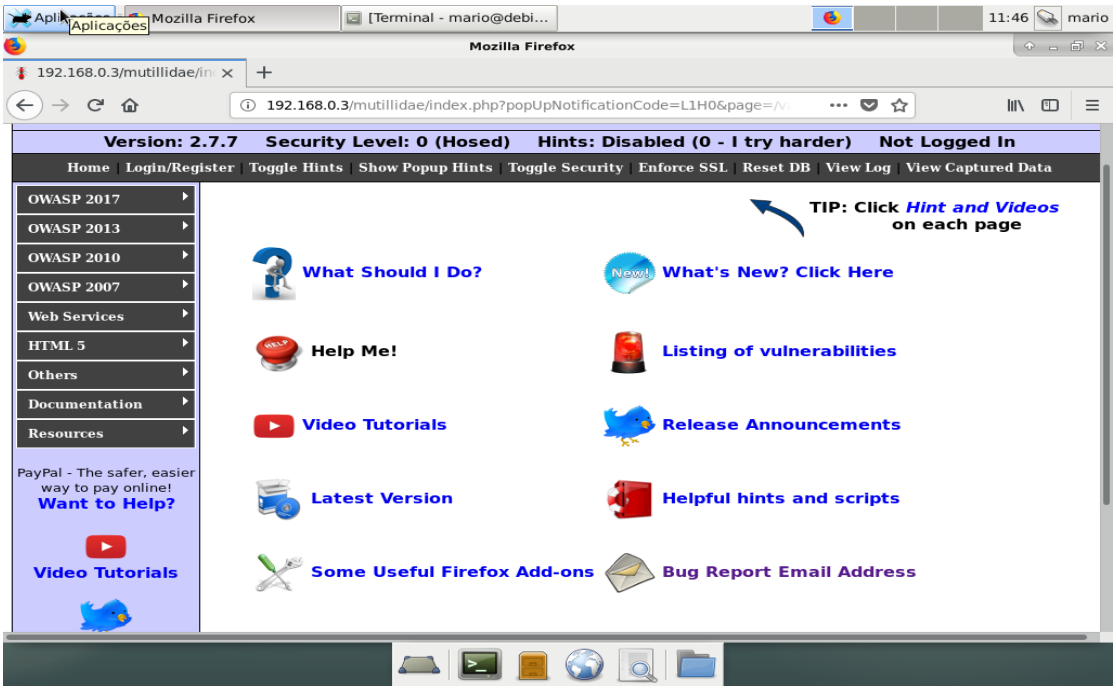
Figura 3.2 - Comparativo de popularidade de linguagens de desenvolvimento Web



Fonte: https://w3techs.com/technologies/overview/programming_language/all

Para a utilização do *Mutillidae 2* utilizamos a distribuição Linux Debian. A preferência pela escolha do Debian se deu pela nossa experiência pessoal; consideramos o Debian uma distribuição estável e reconhecida pela sua comunidade ativa. A Figura 3.3 mostra a tela inicial do sistema *Mutillidae 2* sendo acessado por um navegador.

Figura 3.3 - Tela inicial do *Mutillidae* 2



Fonte: acervo do autor.

3.3 Seleção das ferramentas *open source* para testes de vulnerabilidade Web

O processo de seleção dos *scanners* de vulnerabilidades foi feito segundo quatro critérios para garantir soluções adequadas para o propósito e o escopo do experimento proposto:

1- Ferramentas contidas na listagem geral da OWASP;

2- Categoria do software (*open source*);

3- Posição no ranking *SecTools* e *WebAppSec* (Wasc);

4- Possuir interface gráfica amigável e/ou relatório final gráfico;

No primeiro critério foi utilizada a listagem geral de ferramentas de pesquisa de vulnerabilidades disponibilizada pela OWASP. Por ser uma entidade reconhecida por muitos consórcios e entidades governamentais de segurança cibernética, consideramos a listagem fornecida por ela como uma fonte confiável de ferramentas para testes de vulnerabilidades.

A partir da lista de ferramentas selecionadas pela OWASP, utilizamos o ranking do site *SecTools.org* da *WebAppSec* (Wasc) como filtro para elencar quais são as mais utilizadas pelos profissionais da área de segurança. O ranking se baseia na opinião de mais de dois mil profissionais da área de cibersegurança, que concedem notas às ferramentas de acordo com seus recursos e popularidade. O *WebAppSec* é uma organização sem fins lucrativos composta por especialistas, autoridades da indústria e representantes de organizações da cibersegurança que produzem documentação e padronizações para o campo da segurança cibernética. Desta listagem, utilizando o segundo critério, filtramos apenas os softwares que se enquadram na categoria *open source*. Durante a pesquisa, notou-se a existência de uma ampla gama de ferramentas para análise de falhas em ambientes Web, muitas delas gratuitas, porém, de código fechado; por este motivo, foram desconsideradas.

O terceiro critério utilizado foi a verificação da posição das ferramentas por meio dos sites *SecTools* e *WebAppSec*. O *SecTools* faz uma enquete quantitativa para estabelecer um ranking das soluções de software mais populares para utilização em segurança de redes e sistemas, de acordo com profissionais da área. O *SecTools* faz uma listagem das 125 ferramentas de segurança utilizadas e reconhecidas pela comunidade de Segurança da Informação. A listagem da Wasc é efetuada por uma lista de pesquisadores e colaboradores, que vão de profissionais especialistas da *WhiteHat Security* a especialistas da indústria, como a HP, por exemplo.

Dessas listas, filtrou-se as dez ferramentas mais bem ranqueadas e excluiu-se do resultado as ferramentas que não se enquadravam no critério quatro (possuir interface gráfica e/ou relatório final gráfico). Essa análise revelou que as ferramentas *OWASP ZAP*, *W3AF*, *GOLISMERO*, *ARACHNI* e *VEGA* tiveram melhor avaliação no ranking combinado, tendo sido estas as escolhidas para pesquisa. O *OpenVAS* não foi escolhido pois, de acordo com MARTINELO (2014), é considerado um software de análise de vulnerabilidades genérico, que verifica superficialmente falhas em diversos tipos de serviços em vez de focar apenas no ambiente Web e suas falhas relacionadas. No ranking do *SecTools*, ele também não está listado como uma ferramenta especializada em análise de vulnerabilidades Web.

Algumas destas ferramentas possuem outras incorporadas, como o *Golismero*, que incorpora o uso do *nmap*, do *sqlmap*, do *nikto*, *Grabber*, entre outros. Estas tiveram preferência na escolha do teste para acompanhar o quarto critério de escolha. A Tabela 3.1 apresenta a seleção das ferramentas segundo os critérios previamente apresentados.

Tabela 3.1 - Critérios para seleção das ferramentas

	Critério 2 (*open source*)	**Critério 3 (sectools/Webappsec)**	**Critério 4 (GUI / Relatório gráfico)**
OWASP ZAP	Sim	14° (Webappsec)	Possui GUI e relatório gráfico
GoLismero	Sim	10° (Webappsec)	Não possui GUI, mas possui relatório gráfico
Arachni	Sim	1° (Webappsec)	Possui GUI e relatório gráfico
Vega	Sim	7° (Webappsec)	Possui GUI e relatório gráfico somente interno
W3AF	Sim	18° (sectools) / 8° (Webappsec)	Possui GUI e relatório gráfico
Acunetix	Sim com limitações	41° (sectools)	Possui GUI e relatório gráfico
Nessus	Sim até a versão 3.0	3° (sectools)	Possui GUI e relatório gráfico
Nexpose	Sim com limitações	36° (sectools)	Possui GUI e relatório gráfico
Burp	Sim com limitações	13° (sectools)	Possui GUI e relatório gráfico somente interno
Nikto	Sim	14° (sectools)	Não possui GUI e não possui relatório gráfico
Grabber	Sim	2° (Webappsec)	Não possui GUI e não possui relatório gráfico

Fonte: Construído pelo autor.

3.3.1 OWASP ZAP

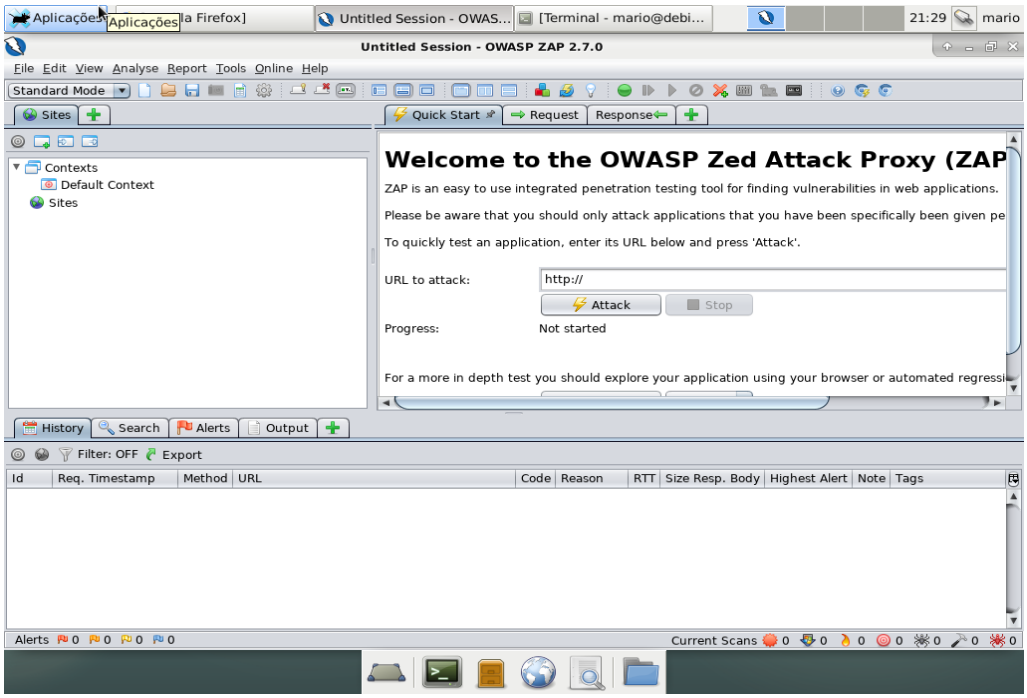
O OWASP ZAP[1] é uma ferramenta *open source* criada pelo OWASP que age como um servidor proxy, com a finalidade de detectar vulnerabilidades em aplicações Web. Este é um dos projetos mais ativos da OWASP, já sendo traduzido para mais de 25 línguas diferentes. De acordo com SILVA et al. (2014), a ferramenta fornece scanners automatizados, bem como um

[1] Disponível em: <https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project>. Acesso em: jan. 2019.

conjunto de ferramentas que permitem encontrar vulnerabilidades de segurança manualmente. TORRES (2014) complementa que o OWASP ZAP é muito semelhante a outros programas de teste de segurança, o que deixa a curva de aprendizado da aplicação um pouco menor.

A ferramenta foi desenvolvida em Java, o que permite que ela rode em diversos sistemas operacionais como Windows, Linux ou Mac OS. Entre os recursos fornecidos pelo ZAP destacamos o proxy ativo, que permite capturar trafego em tempo real; o *Web Crawler*, que percorre o site alvo tentando encontrar todas as páginas que compõem o ambiente; o scanner automatizado, que efetua as pesquisas básicas oferecidas pelo zap sem necessidade de prévia configuração; e o módulo *fuzzer*, que permite a geração de tráfego aleatório, de modo a verificar o comportamento do sistema quando recebe tipos de dados diferentes do esperado. A Figura 3.4 mostra a interface principal do programa OWASP ZAP em execução.

Figura 3.4 – Interface gráfica do OWASP ZAP



Fonte: acervo do autor.

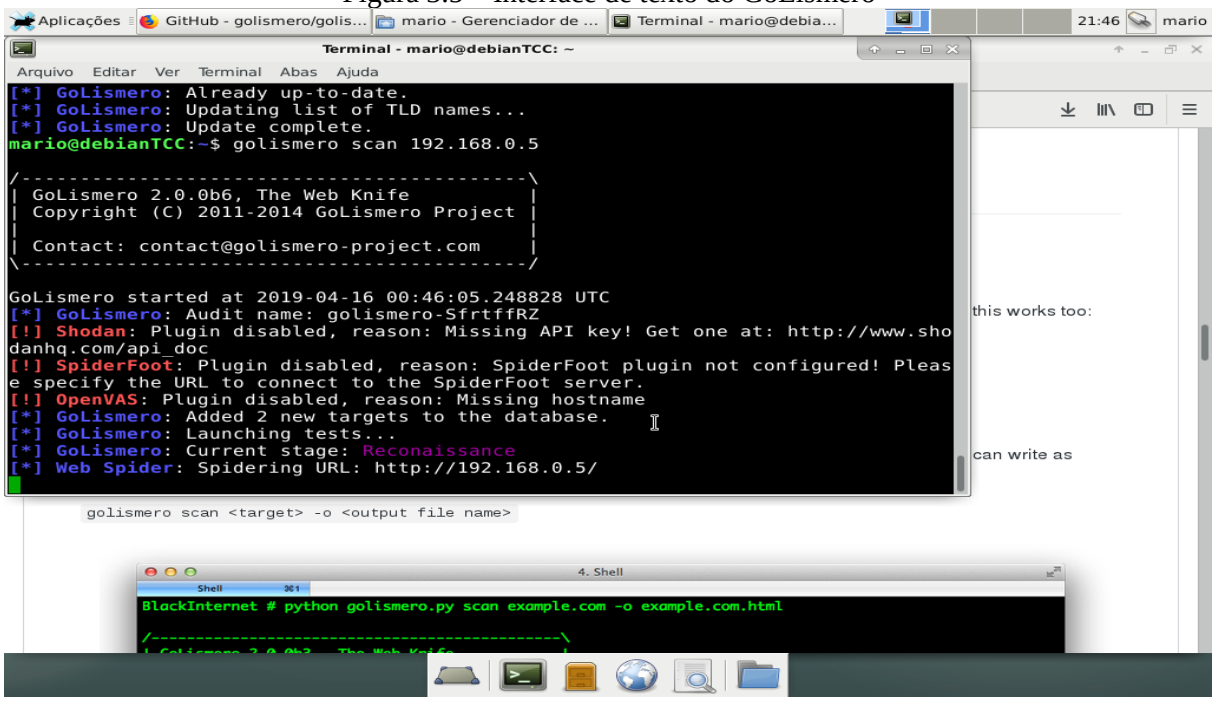
3.3.2 GoLismero

O GoLismero[2] é uma suíte de teste de segurança que agrupa várias ferramentas de código aberto, como o *Nikto*, o *Nmap*, o *SQLmap* etc., e tem possibilidade de utilização de ferramentas

[2] Disponível em: <http://www.golismero.com/>. Acesso em: jan. 2019.

proprietárias por meio do uso de plug-ins, como o *Shodan* ou o *OpenVAS*, por exemplo. Por ser de código aberto e totalmente escrito em Python, permite que possa ser expandido para outros tipos de *scan* de forma facilitada. Um ponto positivo notado no GoLismero é a geração de relatório final em formato html, que possui uma formatação bem executada, deixando muito fácil a leitura dos resultados por parte do usuário. A Figura 3.5 demonstra a interface de texto do GoLismero.

Figura 3.5 – Interface de texto do GoLismero



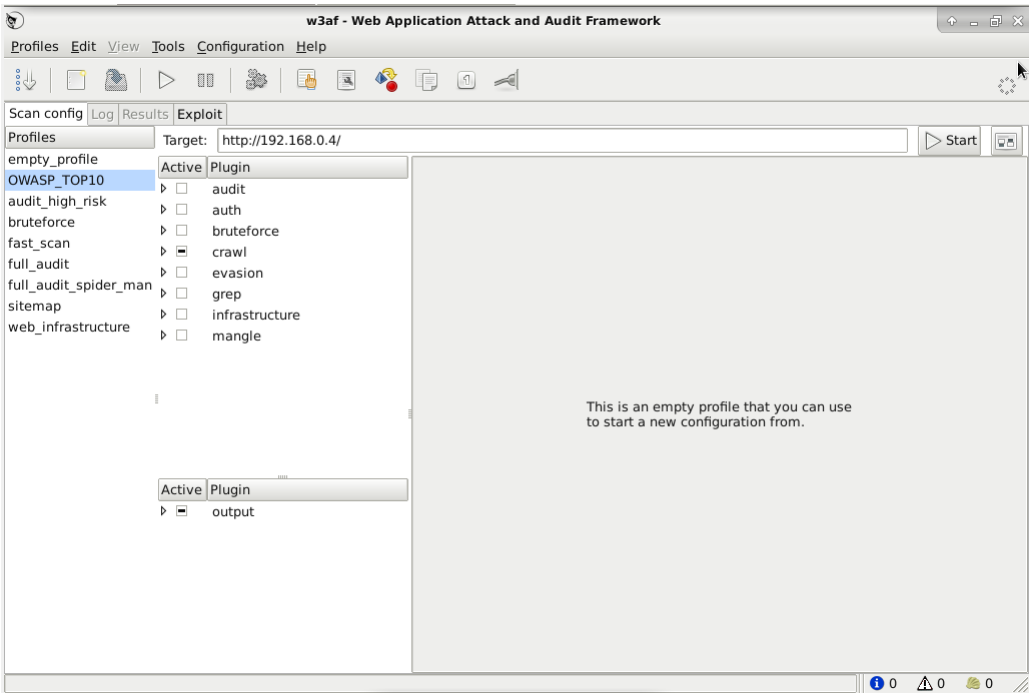
Fonte: acervo do autor.

3.3.3 W3af

W3af[3] é a abreviação de *Web Application Attack and Audit Framework*. De acordo com TORRES (2014), o W3af é uma ferramenta *open source* que faz testes de vulnerabilidades contra servidores Web. Assim, o software tem como objetivo fazer a varredura e identificação de falhas em um ambiente Web. Além de permitir o *crawling* para identificação de diversos tipos de arquivos e páginas do site, o programa tem mais de 200 plug-ins para análise de diversas categorias de falhas, como injeção de SQL, XSS e XSRF. SILVA et al. (2014) também explicam que a ferramenta pode ser utilizada para testes de injeção de código. A Figura 3.6 demonstra a Interface do W3af.

Figura 3.6 – Tela inicial da interface gráfica do W3af

[3] Disponível em: <http://w3af.org/>. Acesso em: jan. 2019.



Fonte: acervo do autor.

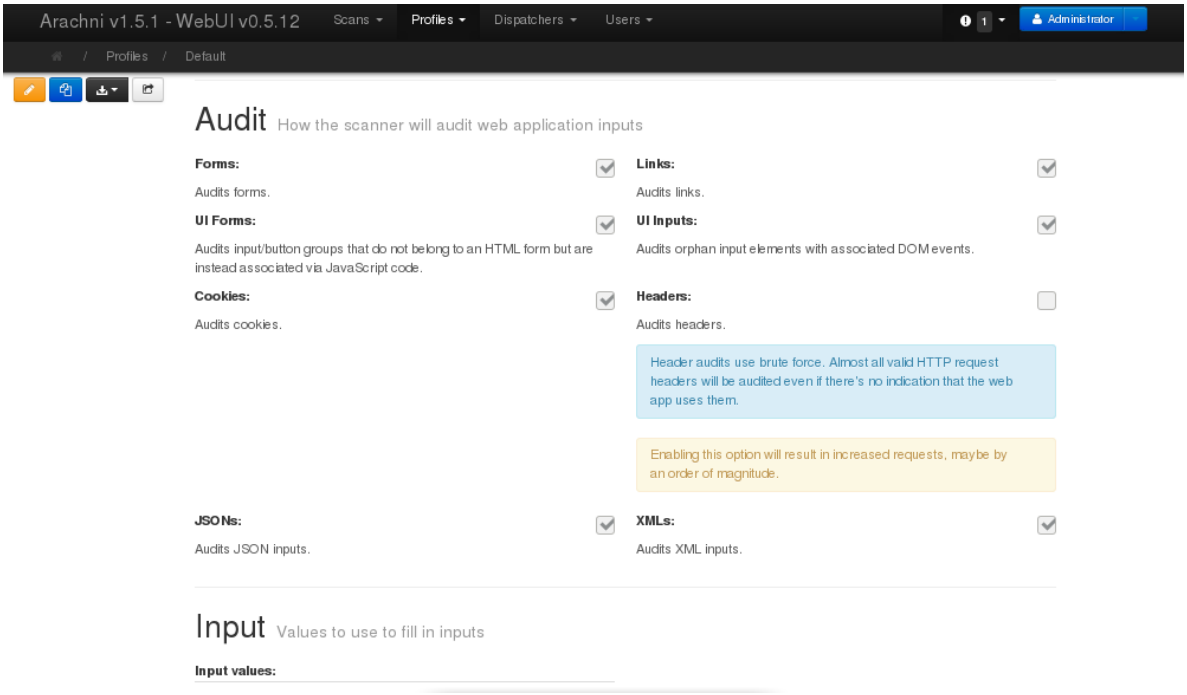
3.3.4 Arachni

O Arachni[4] é um framework de auditoria de sistemas Web muito completo, totalmente modular e que suporta múltiplas instâncias para aumentar a perfomance. Ele foi desenvolvido na linguagem Ruby e tem seu código aberto. Por ser modular, possibilita uma fácil customização e implementação de novos testes. Ele tem suporte a multiplataformas, suportando a maioria dos sistemas operacionais comerciais (Windows, OS X, Linux etc.), além de possibilitar a distribuição *empacotada*, facilitando a utilização em ambientes isolados.

Ele possui uma lista realmente extensa de opções de auditoria, fingerprint e detecção de vulnerabilidades. Contudo, uma caracteristica que o diferencia é o seu relatorio final com os resultados do escaneamento, tendo uma categoria própria para as vulnerabilidades da lista *OWASP Top Ten* . Ele possui uma interface Web que pode ser adicionada ao ambiente de uso; essa integração com o browser permite que ele consiga efetuar testes mais elaborados, que utilizem tecnologias como JavaScript, HTML5, manipulação de DOM e AJAX. Na Figura 3.7 podemos observar a tela para criação e edição da lista parcial de parâmetros de auditoria suportados pelo Arachni.

[4]Disponível em: <https://www.arachni-scanner.com/>. Acesso em: jan. 2019.

Figura 3.7 – Tela com a edição das opções de criação de perfis para auditoria de sistemas



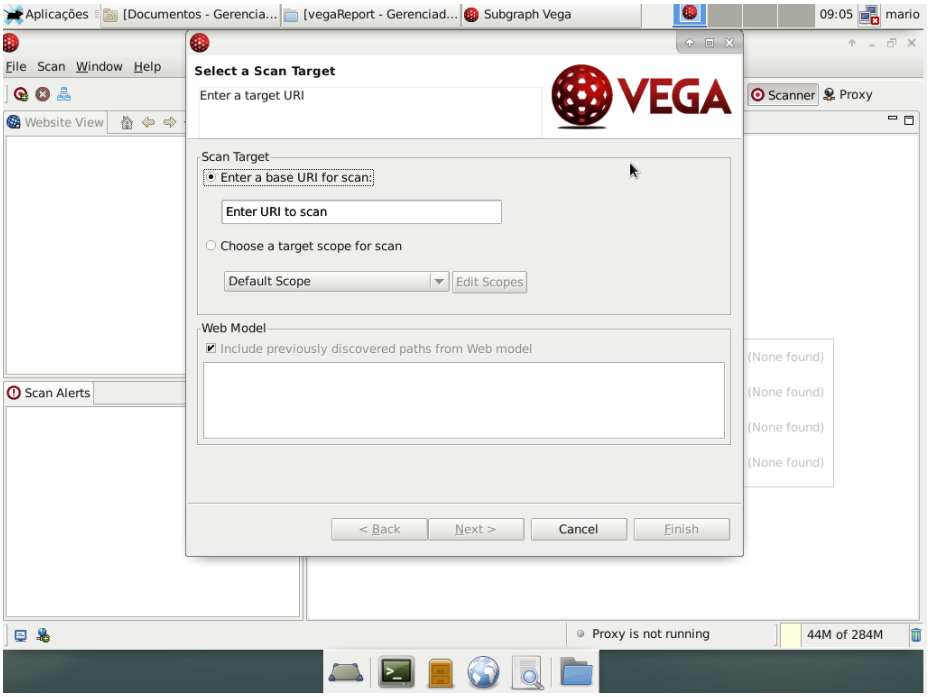
Fonte: acervo do autor.

3.3.5 Vega

O Vega[5] é um scanner de sistemas Web gratuito e de código aberto desenvolvido na linguagem Java, permitindo que ele seja executado no Linux, OS X e Windows. Ele permite encontrar e validar possibilidades de *SQL Injection, Cross Site Scripting (XSS)* e exposição de dados sensíveis, entre outras vulnerabilidades. Inclui um scanner automatizado para execução de testes rápido e um proxy para interceptação e inspeção de tráfego. O Vega suporta também a pesquisa por TLS/SSL de modo a identificar possíveis falhas para aprimorar a segurança de servidores TLS. Uma característica positiva do Vega é poder ser estendido via módulos criados em javascript. Uma questão negativa é o relatório final com os resultados das pesquisas de vulnerabilidades ainda não ter possibilidade se ser exportado para arquivos externos. A Figura 3.8 demonstra a interface do Vega.

[5] Disponível em: <https://subgraph.com/vega/>. Acesso em: jan. 2019.

Figura 3.8 – Tela inicial da interface gráfica do Vega



Fonte: acervo do autor.

4 RESULTADOS

Esta Seção apresenta as informações obtidas após a execução dos testes experimentais propostos no ambiente do *Mutillidae*. Na Seção 4.1 estão todas as vulnerabilidades existentes no cenário simulado. Na Seção 4.2, e suas subseções, são apresentados os resultados obtidos na execução de cada uma das ferramentas selecionadas para o teste. Na Seção 4.3 apresentamos a tabulação dos resultados obtidos. E na Seção 4.4 discutimos estes resultados.

4.1 Vulnerabilidades existentes no *Mutillidae*

Para oferecer um procedimento que seja padronizado e permita uma comparação do resultado gerado pelas ferramentas de forma crível e eficiente, compilamos as informações de todas as páginas que compõem o framework *Mutillidae* e as vulnerabilidades contidas em cada uma delas. Estas informações estão disponíveis de forma mais extensa na própria documentação do *Mutillidae* (em *Documentation/vulnerabilities*) e foram tabuladas para deixar a visualização facilitada para posterior comparação com o resultado dos testes realizados. O objetivo é que a Tabela 4.1 sirva de referência para a análise dos resultados.

Tabela 4.1 - Páginas e vulnerabilidade do *Mutillidae*

Arquivo comprometido	Vulnerabilidades mapeadas	Núm. falhas
add-to-your-blog.php	• Injeção de SQL • XSS • CSRF • Injeção de HTML • Injeção de LOG • Erro da aplicação visível ao usuário	8
arbitrary-file-inclusion.php	• Arquivo do sistema comprometido • Injeção HTML • XSS • Carregar qualquer página da aplicação por URL comprometida	4
back-button-discussion.php	• XSS • Injeção de HTML e Javascript • Redirecionamento de página via comprometimento de cabeçalho	4
browser-info.php	• XSS • Injeção de HTML e Javascript	4
capture-data.php	• XSS • Injeção de SQL, HTML e LOG	4
captured-data.php	• XSS • Injeção de HTML	2
client-side-control-challenge.php	• XSS • Injeção de HTML • Validação de Javascript falha	3

conference-room-lookup.php	• Injeção de dados do LDAP	1
dns-lookup.php	• Validação de Javascript falha • XSS • Injeção de SQL e HTML • Execução de comando do SO	7
document-viewer.php	• XSS • Injeção de HTML • Modificação de LOG • Comprometimento de HTTP GET/POST	4
echo.php	• XSS • Injeção de comandos do SO, HTML, LOG • Comprometimento de HTTP GET/POST • Validação de Javascript falha	8
edit-account-profile.php	• Injeção de SQL, HTML, LOG • XSS • Referência direta a objeto insegura	4
footer.php	• XSS	1
header.php	• XSS	2
html5-storage.php	• Injeção DOM	1
index.php	• XSS • Injeção de SQL • Comentários visíveis no HTML • HTTP splitting • Elevação de privilégios	5
log-visit.php	• Injeção de SQL • XSS	4
login.php	• Injeção de SQL, HTML, LOG • XSS • Falha de autenticação	6
password-generator.php	• Injeção de Javascript	1
pen-test-tool-lookup.php	• Injeção de JSON	1
pen-test-tool-lookup-ajax.php	• Injeção de JSON	1
phpinfo.php	• Informações de configuração visíveis • Acesso à plataforma e caminhos da aplicação	2
phpmyadmin.php	• Informações de configuração visíveis • Acesso à plataforma e caminhos da aplicação	2
process-commands.php	• Manipulação de cookies	1
process-login-attempt.php	• O mesmo de Login.php (esta é a página de ação)	6
redirectandlog.php	• Redirecionamento sem validação	1
register.php	• XSS • Injeção de SQL, HTML e LOG	6
repeater.php	• Injeção de HTML • XSS • Buffer overflow	3
robots.txt	• Lista diretórios que deveriam ser privados	1
secret-administrative-pages.php	• Libera informações de configuração do servidor	1
set-background-color.php	• XSS • Injeção de CSS	2
show-log.php	• XSS • Injeção de HTML • DoS através do LOG	4
site-footer-xss-discusson.php	• XSS	1
source-viewer.php	• XSS	3

	• Injeção de HTML • Carregamento arbitrário de arquivos	
styling.php	• Acesso à plataforma e caminhos da aplicação Injeção • HTML • XSS	3
text-file-viewer.php	• Carregamento arbitrário de arquivos • Injeção HTML • XSS	3
upload-file.php	• Carregamento arbitrário de arquivos • Injeção HTML • XSS	3
user-agent-impersonation.php	• XSS • Injeção de Javascript • Furto de credencial	3
user-info.php	• Injeção de SQL • XSS • Validação de Javascript falha	4
user-info-xpath.php	• XSS	3
user-poll.php	• Injeção de SQL, HTML • XSS	4
view-someones-blog.php	• XSS	1
view-user-privilege-level.php	• CBC bit flipping attack	1
Webservices/rest/ws-user-account.php	• Injeção de SQL • Enumeração de usuários	2
Webservices/soap/ws-lookup-dns-record.php	• Injeção de comandos SOAP • Enumeração de usuários	2
Webservices/soap/ws-user-account.php	• Injeção SQL • Enumeração de usuários	2
xml-validator.php	• *Parsing* de entidade externa de XML • Injeção de XML • XSS	4

Fonte: elaborado pelo autor.

Algumas páginas do *Mutillidae* foram ignoradas pelo experimento, pois não possuem erros internos catalogados ou as vulnerabilidades existentes não estão no escopo do teste proposto, já que não se encaixam nas categorias propostas pelo *Top Ten* 2017 da OWASP. São elas:

- Autorization-required.php
- Cliente-side-comments.php
- Database-offline.php
- Framer.php
- Framing.php
- Home.php
- Installation.php
- Page-not-found.php

- Php-errors.php
- Privilegie-escalation.php
- Rene-megritte.phpset-up-database.php
- Sqlmap-targets.php
- Usage-instructions.php
- Notes.php

Feitas as varreduras usando as ferramentas selecionadas, foram verificadas e tabuladas as vulnerabilidades encontradas, quantas de forma correta, quantas em falso positivo e quantas em falso negativo.

4.2 Análise das ferramentas

Nesta Seção do trabalho, são apresentados os resultados dos testes realizados no ambiente Web do *Mutillidae* preparado para o cenário. Quatro categorias de métricas foram empregadas para comparação das ferramentas: número de vulnerabilidades detectadas, quantos falsos positivos ocorreram, quantos falsos negativos e o tempo total da varredura realizado. Notou-se em todas as ferramentas que o log dos resultados é normalmente muito extenso devido a uma característica comum, que é a de reportar avisos (*warnings*) da possibilidade de uma falha ocorrer, além das vulnerabilidades em si. As falhas encontradas são comumente categorizadas em *high risk* (alto risco), *medium risk* (risco médio), *low risk* (baixo risco) e *informational* (informativo). Normalmente, as aplicações encontram poucas vulnerabilidades de alto e médio risco, e a maior parte de baixo risco (TORRES, 2014). Além dessas vulnerabilidades, uma grande massa de dados informacionais é reportada depois de uma execução do aplicativo, como avisos sobre versões de interpretadores de *scripts* desatualizadas.

É importante pontuarmos que, durante os testes, as categorias *A8 - Insecure Deserialization* e *A10 - Insufficiente Logging & Monitoring*, por se tratarem de falhas de segurança que são intrínsecas ao desenvolvimento do sistema (A8) e ao gerenciamento dele (A10), não são mapeadas apropriadamente por *scanners* de vulnerabilidades. Portanto, não serão consideradas para fins de contabilidade das vulnerabilidades proposta.

A categoria *informational*, deste modo, não se refere às vulnerabilidades, e sim às informações obtidas durante a varredura, podendo ser úteis à melhoria da segurança da aplicação. Como exemplo, temos uma entrada de log detectada pela ferramenta OWASP ZAP:

Informational (warning): X-Frame-Options header not set

Description: X-Frame-Options header not included in the HTTP response to protect against 'ClickJacking' attacks

URL: http://192.168.0.5/mutillidae/styles/global-styles.css

Como as informações da categoria *informational* não são relacionadas aos riscos descritos no *OWASP Top Ten*, esses dados foram ignorados na análise final. As falhas de baixo risco que também não se encaixam em nenhuma categoria do *Top Ten* foram igualmente desconsideradas.

Para tabulação dos resultados, considerou-se:

- **Falhas existentes:** apresenta a quantidade de falhas já configuradas e catalogadas do framework *Mutillidae*;
- **Falhas detectadas:** Total de falhas encontradas pelo sistema, considerando falsos positivos;
- **Falhas detectadas corretamente:** Total de falhas identificadas corretamente pelo *scanner*;
- **Falso positivo:** quantidade de dados indevidamente apontados como erro em relação ao framework *Mutillidae*;
- **Falso negativo:** quantidade de vulnerabilidades existentes no *Mutillidae* que não foram identificadas durante a varredura;
- **Acurácia:** relação percentual entre as falhas detectadas corretamente *versus* falhas existentes, sem inferência de falsos negativos e falsos positivos;
- **Relação detecção/erro:** mostra o percentual da associação entre a quantidade de falhas detectadas pelo *scanner* e, dentre essas, a perspectiva desse total possuir algum erro, ou seja, falso positivo ou falso negativo.

4.2.1 Execução de teste da ferramenta OWASP ZAP

Inicialmente foi utilizado o módulo de Web Crawler para fazer o mapeamento de todas as páginas existentes no ambiente testado através da página principal (index.php). Durante essa etapa, as falhas ainda não foram analisadas, pois este primeiro passo é feito apenas para identificação de recursos.

Após a finalização do *crawler*, que levou cerca de seis minutos para ser completado, verificou-se que todas as páginas do site foram corretamente mapeadas. O próximo passo foi a execução do módulo Active Scan, que fez a varredura à procura de vulnerabilidades em todas as páginas do ambiente, previamente mapeadas. O processo de análise de falhas levou 29 minutos e 20 segundos para ser finalizado e retornou um total de 91 falhas identificadas. O teste foi executado mais duas vezes para verificar se os resultados apresentariam diferenças, o que não ocorreu. O processo todo, incluindo as etapas de *crawling* e varredura, durou 35 minutos e 20 segundos. A ferramenta se saiu bem na detecção das vulnerabilidades de todas categorias, com exceção da categoria A7 (*Cross-Site Scripting XSS)*. Nas categorias A2 (*Broken Authentication*), A4 (*XML External Entities* XXE) e A6 (*Security Misconfiguration*), todas as falhas foram corretamente identificadas.

Um problema detectado ao final da varredura foi a enorme quantidade de falsos positivos reportados pelo OWASP ZAP na categoria de vulnerabilidade que se refere à injeção de código (A1). No final da varredura, 91 falhas foram identificadas pelo scanner; após análise manual, verificou-se que 41 delas eram falsos positivos. Portanto, das 107 existentes no ambiente, apenas 50 foram corretamente detectadas. A Tabela 4.2 apresenta os resultados obtidos ao final do teste de varredura.

Tabela 4.2 - Informações obtidas após a varredura com o OWASP ZAP

ID	TIPO DE FALHA	FALHAS EXISTENTES	FALHAS DETECTADAS	FALHAS DETECTADAS CORRETAMENTE	FALSOS POSITIVOS	FALSOS NEGATIVOS	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
A1	Injection	61	68	31	37	30	50,82%	98,53%
A2	Broken Authentication	1	1	1	0	0	100,00%	0,00%
A3	Sensitive Data Exposure	2	1	1	0	1	50,00%	100,00%
A4	XML External Entities (XXE)	5	6	5	1	0	100,00%	16,67%
A5	Broken Access Control	3	2	1	1	2	33,33%	150,00%
A6	Security Misconfiguration	4	4	4	0	0	100,00%	0,00%
A7	Cross-Site Scripting (XSS)	29	8	6	2	23	20,69%	312,50%
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	1	1	0	1	50,00%	100,00%
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.2.2 Execução de teste da ferramenta GoLismero

O GoLismero, assim como o Arachni e o Vega, realiza o processo de *crawling* em simultâneo com a identificação de vulnerabilidades quando executado com suas configurações iniciais. Por se tratar de uma suíte que utiliza internamente outras ferramentas para executar certos passos da busca de páginas e de vulnerabilidades, acaba herdando as fraquezas destas ferramentas quando executadas em configurações iniciais. O tempo total utilizado pelo processo de identificação de páginas e mapeamento de vulnerabilidades foi de 22 minutos e 3 segundos.

O GoLismero não conseguiu identificar nenhuma falha da categoria A3 (*Sensitive Data Exposure*) e teve um desempenho muito baixo nas outras categorias, com exceção das categorias A2 (*Broken Authentication*) e A9 (*Using Components with Known Vulnerabilities*), nas quais conseguiu identificar todas as vulnerabilidades com precisão.

Das 107 vulnerabilidades mapeadas no sistema alvo, ela conseguiu identificar corretamente apenas 27, considerando o escopo do *Top Ten*. Um ponto considerado positivo é o relatório final gerado pela ferramenta, que é bem detalhado no quesito de opções de visão, disponibilizando visões técnica, de vulnerabilidades e sintética. As informações geradas pela execução da ferramenta podem ser vistas na Tabela 4.3.

Tabela 4.3 - Informações obtidas após a varredura com GoLismero

ID	TIPO DE FALHA	FALHAS EXISTENTES	FALHAS DETECTADAS	FALHAS DETECTADAS CORRETAMENTE	FALSOS POSITIVOS	FALSOS NEGATIVOS	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
A1	Injection	61	14	10	4	51	16,39%	392,86%
A2	Broken Authentication	1	1	1	0	0	100,00%	0,00%
A3	Sensitive Data Exposure	2	0	0	0	0	0,00%	0,00%
A4	XML External Entities (XXE)	5	6	4	2	1	80,00%	50,00%
A5	Broken Access Control	3	3	2	1	1	66,67%	66,67%
A6	Security Misconfiguration	4	3	1	2	3	25,00%	166,67%
A7	Cross-Site Scripting (XSS)	29	10	7	3	22	24,14%	250,00%
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	2	2	0	0	100,00%	0,00%
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.2.3 Execução de teste da ferramenta W3af

Para o teste do software W3AF foi escolhido o perfil OWASP_TOP10, que faz uma varredura somente das vulnerabilidades categorizadas de acordo com a documentação *Top Ten*. Primeiramente, foi iniciado o processo de *crawling* para identificação das páginas Web que constituem o site. Notou-se que a ferramenta se comportou de forma instável durante a execução dos testes, chegando a travar em uma das execuções, durante a etapa de *crawling*, que foi completado efetivamente apenas depois da terceira tentativa.

Algumas páginas também não foram identificadas corretamente durante esse processo, e tiveram de ser testadas manualmente após a execução do teste principal. Outro problema mapeado durante a execução dos testes utilizando o perfil da OWASP_TOP10 foi que ele possui uma incompatibilidade interna de plugins na versão mais recente do software, contudo, a própria mensagem de erro do programa já instrui o usuário sobre como desabilitar o plugin referido e continuar o teste proposto. O processo de *crawling* levou 14 minutos para ser completado e a etapa de identificação das vulnerabilidades levou mais 28 minutos e 15 segundos. No total, o processo todo levou 42 minutos e 15 segundos. Após o problema inicial com o reconhecimento das páginas, a identificação dos problemas ocorreu sem dificuldades.

A ferramenta gerou alguns falsos positivos, em maior quantidade na categoria de risco A7 (*Cross-Site Scripting XSS*), e uma quantidade expressiva de falsos negativos na categoria A1

(*Injection*). Apesar dos falsos positivos, o W3af foi a ferramenta que melhor detectou falhas da categoria A7, tendo também identificado todas as falhas na categoria A5 (*Broken Access Control*). Das 107 falhas existentes no ambiente, 48 foram corretamente identificadas pelo W3af. Apenas 25 falsos positivos foram detectados, valor que pode ser considerado razoável em relação ao total de problemas. Todos os resultados obtidos após análise das vulnerabilidades com a ferramenta podem ser verificados na Tabela 4.4.

Tabela 4.4 - Informações obtidas após a varredura com o W3af

ID	TIPO DE FALHA	FALHAS EXISTENTES	FALHAS DETECTADAS	FALHAS DETECTADAS CORRETAMENTE	FALSOS POSITIVOS	FALSOS NEGATIVOS	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
A1	Injection	61	25	20	5	41	32,79%	184,00%
A2	Broken Authentication	1	2	1	1	0	100,00%	50,00%
A3	Sensitive Data Exposure	2	2	1	1	1	50,00%	100,00%
A4	XML External Entities (XXE)	5	6	3	3	2	60,00%	83,33%
A5	Broken Access Control	3	3	3	0	0	100,00%	0,00%
A6	Security Misconfiguration	4	3	2	1	2	50,00%	100,00%
A7	Cross-Site Scripting (XSS)	29	31	17	14	12	58,62%	83,87%
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	1	1	0	1	50,00%	100,00%
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.2.4 Execução de teste da ferramenta Arachni

O Arachni realiza o processo de *crawling* em simultâneo com a identificação das vulnerabilidades e mapeou todas as páginas disponíveis no sistema vulnerável. Por ser um software modular, que possui plugins que efetuam cada um dos passos de verificação de vulnerabilidade, possuir uma lista bem grande de opções, e o perfil padrão de execução possuir quase a totalidade das opções ativadas, causou uma execução lenta da varredura. O processo de identificação das páginas e mapeamento das vulnerabilidades levou 120 horas, 24 minutos e 15 segundos.

A ferramenta apresentou bons resultados de forma geral, apesar da baixa performance, conseguindo identificar um total de 103 falhas. Contudo, destas falhas, 35 foram mapeadas como falsos positivos; das 107 falhas contidas no sistema vulnerável, 68 foram mapeadas

corretamente. Um ponto considerado negativo foi a geração excessiva de informações adicionadas no log de execução pós varredura. Um ponto muito positivo foi a formatação do resultado gerado, que disponibiliza os dados de forma agrupada por categoria do *Top Ten* e apresenta de forma textual as sugestões de correção para vulnerabilidade. As informações encontradas após o processo de *crawling* e *scan* da ferramenta podem ser vistas na Tabela 4.5.

Tabela 4.5 - Informações obtidas após a varredura com o Arachni.

ID	TIPO DE FALHA	FALHAS EXISTENTES	FALHAS DETECTADAS	FALHAS DETECTADAS CORRETAMENTE	FALSOS POSITIVOS	FALSOS NEGATIVOS	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
A1	Injection	61	75	43	32	6	70,49%	50,67%
A2	Broken Authentication	1	1	1	0	0	100,00%	0,00%
A3	Sensitive Data Exposure	2	1	1	0	1	50,00%	100,00%
A4	XML External Entities (XXE)	5	2	2	0	3	40,00%	150,00%
A5	Broken Access Control	3	1	1	0	2	33,33%	200,00%
A6	Security Misconfiguration	4	3	3	0	1	75,00%	33,33%
A7	Cross-Site Scripting (XSS)	29	18	15	3	14	51,72%	94,44%
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	2	2	0	0	100,00%	0,00%
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.2.5 Execução de teste da ferramenta Vega

A ferramenta Vega efetua o processo de *crawling* e busca de vulnerabilidades de forma simultânea. Em sua configuração inicial o Vega não conseguiu identificar nenhuma vulnerabilidade da categoria A2 (*Broken Authentication*) e teve um desempenho muito fraco de forma geral, apresentando uma quantidade expressiva de falsos positivos na categoria A3 (*Sensitive Data Exposure*). A aplicação também teve problemas em identificar todas as páginas disponíveis no sistema alvo durante o processo de *crawling*,

O tempo consumido para execução dos processos de crawling e de reconhecimento de vulnerabilidades resultou em um total de 20 minutos e 40 segundos. Um ponto negativo da ferramenta foi o excesso de registros do tipo *Informational*, gerado após a varredura. Removendo todos estes avisos e vulnerabilidades de baixo nível que não se encaixavam nas

categorias do OWASP, pouco foi aproveitado dos resultados gerados. As informações resultantes da execução da ferramenta podem ser vistas na Tabela 4.6.

Tabela 4.6 - Informações obtidas após a varredura com o Vega

ID	TIPO DE FALHA	FALHAS EXISTENTES	FALHAS DETECTADAS	FALHAS DETECTADAS CORRETAMENTE	FALSOS POSITIVOS	FALSOS NEGATIVOS	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
A1	Injection	61	6	6	0	55	9,84%	916,67%
A2	Broken Authentication	1	0	0	0	0	0,00%	0,00%
A3	Sensitive Data Exposure	2	39	2	37	0	100,00%	94,87%
A4	XML External Entities (XXE)	5	3	2	1	3	40,00%	133,33%
A5	Broken Access Control	3	2	2	0	1	66,67%	50,00%
A6	Security Misconfiguration	4	1	1	0	0	25,00%	0,00%
A7	Cross-Site Scripting (XSS)	29	3	3	0	0	10,34%	0,00%
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	3	2	1		100,00%	33,33%
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.3 Tabulação dos resultados

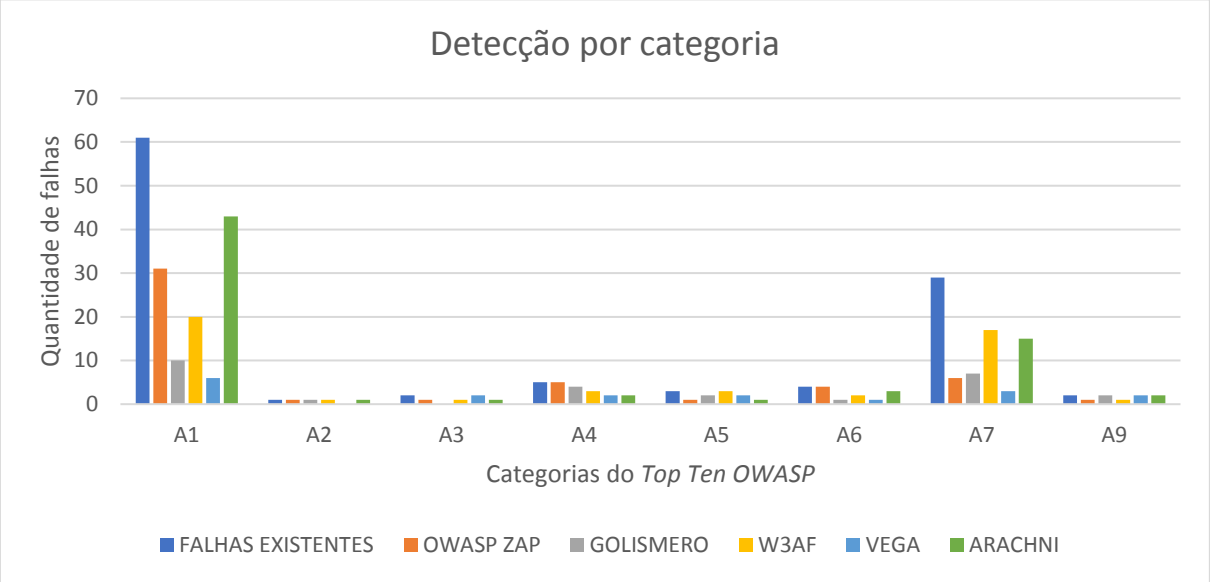
A Tabela 4.7 exibe o total de falhas existentes e o número de falhas corretamente detectadas, divididos por categorias. A Figura 4.1 apresenta a análise de desempenho das ferramentas nesta detecção por categoria.

Tabela 4.7 - Falhas corretamente detectadas pelas ferramentas

ID	NOME DA FALHA	FALHAS EXISTENTES	OWASP ZAP	GOLISMERO	W3AF	VEGA	ARACHNI
A1	Injection	61	31	10	20	6	43
A2	Broken Authentication	1	1	1	1	0	1
A3	Sensitive Data Exposure	2	1	0	1	2	1
A4	XML External Entities (XXE)	5	5	4	3	2	2
A5	Broken Access Control	3	1	2	3	2	1
A6	Security Misconfiguration	4	4	1	2	1	3
A7	Cross-Site Scripting (XSS)	29	6	7	17	3	15
A8	Insecure Deserialization	N/A	N/A	N/A	N/A	N/A	N/A
A9	Using Components with Known Vulnerabilities	2	1	2	1	2	2
A10	Insufficient Logging & Monitoring	N/A	N/A	N/A	N/A	N/A	N/A

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

Figura 4.1 - Desempenho de detecção das ferramentas por categoria *OWASP Top Ten*.



Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

A Tabela 4.8 mostra o tempo total de varredura despendido por cada ferramenta para obter os resultados finais apresentados nesse trabalho. Como, por definição padrão, a ferramenta Arachni possui todos os seus módulos de busca ativados, ela foi a ferramenta que consumiu mais tempo para a realização de todo processo.

Tabela 4.8 – Tempo de execução das ferramentas.

FERRAMENTA	**TEMPO TOTAL DE EXECUÇÃO**
OWASP ZAP	35 minutos e 20 segundos
GOLISMERO	22 minutos e 3 segundos
W3AF	42 minutos e 15 segundos
VEGA	20 minutos e 40 segundos
ARACHNI	120 horas, 24 minutos e 15 segundos

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

A análise dos dados totalizados gerou a Tabela 4.9, que contém cinco colunas que representam os seguintes itens obtidos pela análise das cinco ferramentas: o total de falhas existentes no *framework Mutillidae*; o número de falsos positivos e falsos negativos gerados pelas ferramentas; o número total de falhas detectadas; e o número total de falhas detectadas corretamente.

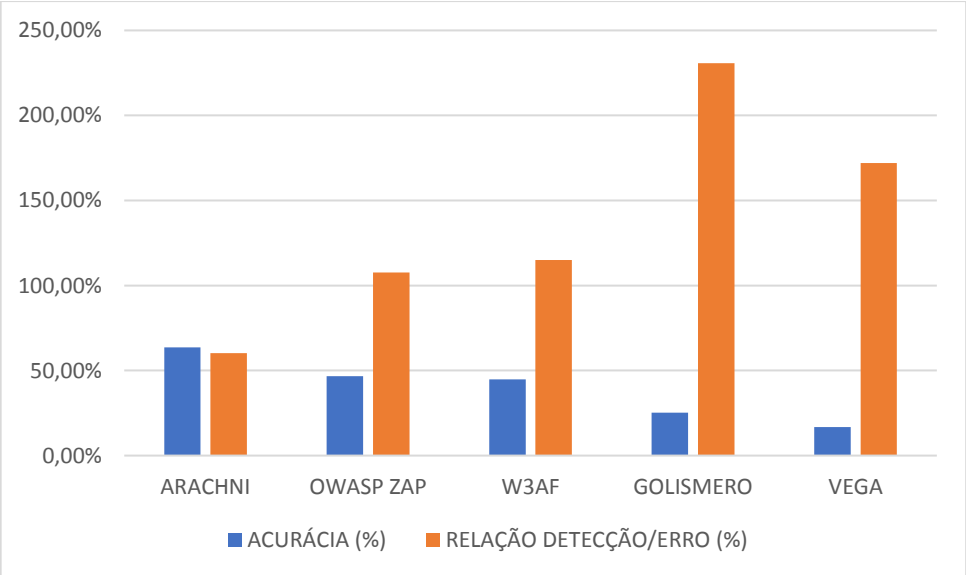
Tabela 4.9 – Total de falhas existentes *vs*. detectadas *vs.* falsos positivos *vs.* falsos negativos.

FERRAMENTA	TOTAL DE FALHAS EXISTENTES	TOTAL DE FALHAS DETECTADAS	TOTAL DE FALSOS POSITIVOS	TOTAL DE FALSOS NEGATIVOS	TOTAL DE FALHAS DETECTADAS CORRETAMENTE	ACURÁCIA	RELAÇÃO DETECÇÃO/ERRO
ARACHNI	107	103	35	27	68	63,55%	60,19%
OWASP ZAP	107	91	41	57	50	46,73%	107,69%
W3AF	107	73	25	59	48	44,86%	115,07%
GOLISMERO	107	39	12	78	27	25,23%	230,77%
VEGA	107	57	39	59	18	16,82%	171,93%

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

A Tabela 4.9, bem como a Figura 4.2, trazem dados denominados “acurácia”, que calcula a porcentagem de falhas detectadas corretamente pelo software sob o número de falhas existentes no *framework Mutillidae*; desta forma, não considera o número de falsos positivos e falsos negativos. Já a coluna “relação detecção/erro” mostra o percentual da associação entre a quantidade de falhas detectadas pelo software e, dentre estas, a perspectiva deste total possuir algum erro, ou seja, falso positivo ou negativo.

Figura 4.2 - Análise percentual de acurácia *vs*. relação detecção/erro



Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

4.4 Discussão dos resultados

Após a compilação e interpretação dos resultados referentes às oito categorias de falhas da Web que foram contempladas pelo trabalho, identificou-se o padrão de comportamento dos softwares analisados para a detecção de cada um dos diferentes tipos de riscos considerados para o teste. A Tabela 4.10 apresenta a relação das categorias analisadas no teste e o nome das soluções que apresentaram melhor resultado, identificando falhas em cada uma delas.

Tabela 4.10 – Relação de acurácia por categoria do *Top Ten*.

ID	NOME DA FALHA	FERRAMENTAS			
A1	Injection	OWASP ZAP	ARACHNI		
A2	Broken Authentication	OWASP ZAP	GOLISMERO	W3AF	ARACHNI
A3	Sensitive Data Exposure	VEGA			
A4	XML External Entities (XXE)	OWASP ZAP	GOLISMERO		
A5	Broken Access Control	GOLISMERO	W3AF	VEGA	
A6	Security Misconfiguration	OWASP ZAP	ARACHNI		
A7	Cross-Site Scripting (XSS)	W3AF	ARACHNI		
A8	Insecure Deserialization	N/A			
A9	Using Components with Known Vulnerabilities	GOLISMERO	VEGA	ARACHNI	
A10	Insufficient Logging & Monitoring	N/A			

Fonte: Elaborado pelo autor com base nos resultados experimentais obtidos.

De acordo com as informações da Tabela 4.10, pode-se verificar que a única categoria que todos os aplicativos (com exceção do Vega) foram capazes de identificar as vulnerabilidades existentes foi a A2 (*Broken Authentication*). A ferramenta Arachni foi a que melhor se saiu em seu desempenho geral, sendo contabilizada em cinco das oito categorias abordadas no trabalho, seguido do OWASP ZAP, com quatro ocorrências.

O W3af, apesar de ter apresentado instabilidade durante o processo de *crawling* para detecção de arquivos, foi a ferramenta com o terceiro melhor resultado no quesito acurácia, detectando corretamente 48 vulnerabilidades (que representa 44,86% do total de falhas existentes). A suíte GoLismero apresentou um desempenho muito fraco e uma alta taxa de falsos negativos; das 107 falhas mapeadas do sistema, ele detectou corretamente apenas 27 (representando uma taxa percentual de 25,23%), o que o colocou em quarto lugar do ranking. A ferramenta Vega não teve um desempenho satisfatório por ter identificado uma quantidade muito pequena das vulnerabilidades, apenas 18 das 107 existentes no ambiente (deixando uma taxa de acurácia de 16,82%), ficando em último lugar no ranking proposto.

Portanto, o Arachni foi a solução que apresentou melhor relação da taxa de falhas identificadas corretamente e falsos positivos quando comparada com as outras ferramentas, além de ter uma interface amigável de fácil utilização. O único ponto negativo foi o tempo

excessivo de execução que o aplicativo demandou para efetuar uma avaliação completa do sistema alvo.

Neste contexto, a necessidade do profissional hacker ético se mostrou verdadeira também como requisito para utilização dos softwares de análise de vulnerabilidades Web. Foi demonstrado que, sem o conhecimento técnico necessário, os aplicativos não terão desempenho aceitável para uma avaliação confiável de um sistema. Incluindo-se nesse aspecto, a necessidade prévia de configuração customizada e/ou capacidade de interpretação dos resultados obtidos pela varredura.

5 CONCLUSÃO

A criação de um perfil do profissional hacker ético permite um melhor entendimento do que é o hacker ético e, consequentemente, a seleção mais aprimorada deste profissional por uma empresa. A avaliação de ferramentas de detecção de vulnerabilidades em suas configurações iniciais permite que tenhamos maior ciência das vantagens e desvantagens destas ferramentas quando utilizadas por profissionais capacitados, ou seja, que consigam calibrá-las apropriadamente para cenários específicos.

Na Seção 5.1, são apresentadas as contribuições deste trabalho. As propostas para trabalhos futuros são descritas na Seção 5.2.

5.1 Contribuições

Neste trabalho, propusemos a criação de um perfil do profissional hacker ético para que este se diferencie dos demais subtipos de hackers nomeados nas bibliografias e na mídia. Para essa diferenciação utilizamos as categorizações propostas por SMITH et al. (2001) e pela pesquisa desenvolvida por CHANDRIKA (2014).

Propusemos também a avaliação de scanners de vulnerabilidades Web quando executados em sua configuração padrão, ou seja, sem customizações das características da varredura. A análise comparativa foi feita usando uma aplicação vulnerável previamente preparada para os testes e uma métrica de avaliação, a lista das dez vulnerabilidades mais recorrentes em sistemas Web fornecido pelo OWASP, na versão 2017. Foram selecionados scanners open source para avaliação. Os resultados do experimento demonstraram que os aplicativos, quando executados em sua configuração inicial, não possuem uma acurácia aceitável, criando muitos falsos negativos e falsos positivos. Como são soluções de código aberto, o ranking estabelecido demonstra onde cada ferramenta pode ter seu código fonte aperfeiçoado, contribuindo, assim, com a comunidade open source da área de segurança.

5.2 Trabalhos futuros

Como trabalhos futuros, propomos avaliar o desempenho dos aplicativos de detecção de vulnerabilidades em outros ambientes alvo, como ambientes compostos por *smart devices*,

plataformas de jogos focando em vulnerabilidades dos consoles e das redes que os sustentam. Também propomos a melhoria da acurácia dos softwares testados neste trabalho, aprimorando os perfis básicos disponibilizados (como é o caso do W3af, que possui uma lista de perfis propostos para uso), ou criando perfis nos aplicativos que não os possuam, a fim de diminuir a quantidade de falsos positivos e melhorar a taxa de detecção/erro. Também se sugere um trabalho sobre a inclusão, no processo de análise, de outras categorias de risco de vulnerabilidades em sistemas Web, além das contempladas pelo *OWASP Top Ten*.

REFERÊNCIAS

AL-KHURAFI, O. B.; AL-AHMAD, M. A. Survey of Web Application Vulnerability Attacks. **2015 4th International Conference On Advanced Computer Science Applications And Technologies (acsat)**, [s.l.], p.154-158, dez. 2015. IEEE.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 27001:2013**: Tecnologia da Informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação. Rio de Janeiro: ABNT, 2013.

ASSUNÇÃO, M. F. **Segredos do hacker ético.** 5. ed. Florianopolis: Visual Books, 2014.

BISHOP, M. About Penetration Testing. **Ieee Security & Privacy Magazine**, [s.l.], v. 5, n. 6, p.84-87, nov. 2007. Institute of Electrical and Electronics Engineers (IEEE).

BIZTECH Africa. **Cybercrime Losses Up By 56%.** 2011. Disponível em: <http://biztechafrica.com/article/cybercrime-?losses-?56/977/>. Acesso em: 10 mar. 2019.

CARVALHO, A. H. P. Segurança de aplicações Web e os dez anos do relatório OWASP Top Ten: o que mudou? **Fasci-Tech – Periódico Eletrônico da FATEC-São Caetano do Sul**, São Caetano do Sul, v.1, n. 8, Mar./Set. 2014, p. 6 a 18.

CHANDRIKA, V. Ethical hacking: types of ethical hackers. **International Journal Of Emerging Technology In Computer Science & Electronics (ijetcse): ISSN: 0976 - 1353.** Vijayawada, Andhra Pradesh, India, p. 43-48. nov. 2014.

CIRCLEID. **Cybercrime Losses Overestimated, Say Researchers.** 2011. Disponível em: <http://www.circleid.com/posts/cybercrime_losses_overestimated_say_researchers/>. Acesso em: 10 mar. 2019.

COSTA, D. G. **Administração de Redes com scripts: Bash Script, Python e VBScript.** Rio de Janeiro: Brasport, 2010.

DEHALWAR, V. et al. Review of Web-based information security threats in smart grid. **2017 7th International Conference On Power Systems (icps)**, [s.l.], p.849-853, dez. 2017. IEEE.

DEITEL, H. M.; DEITEL, P. J. **C++ Como Programar.** 3. ed. São Paulo: Bookman, 2001.

DOUPÉ, A.; COVA, M.; VIGNA, G. Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners. **Detection Of Intrusions And Malware, And Vulnerability Assessment**, [s.l.], p.111-131, 2010. Springer Berlin Heidelberg.

FATIMA, H.; WASNIK, K. Comparison of SQL, NoSQL and NewSQL databases for internet of things. **2016 Ieee Bombay Section Symposium (ibss)**, [s.l.], p.1-6, dez. 2016. IEEE.

GAO/AIMD-96-84. **AGENCIES Need to Improve Implementation of Federal Approach to Securing Systems and Protecting against Intrusions.** 2018. Disponível em: <https://www.gao.gov/products/GAO-19-105>. Acesso em: 05 jan. 2019.

GAO/AIMD-96-84. **COMPUTER Attacks at Department of Defense Pose Increasing Risks.** 1996. Disponível em: <https://www.gao.gov/products/AIMD-96-84>. Acesso em: 26 dez. 2018.

GRABO, C. **Anticipating Surprise:** Analysis for Strategic Warning. Lantham: University Press Of America, 2004.

GROSS, D. '**Mafiaboy' breaks silence, paints 'portrait of a hacker'.** 2011. Disponível em: <http://www.cnn.com/2011/TECH/Web/08/15/mafiaboy.hacker/index.html?iref=obnetwork>. Acesso em: 10 mar. 2019.

GUIRADO, R. Penetration Testing, General Concepts and Current Situation, CISA, CGEIT, 2009.

HAFELE, D. **Three Different Shades of Ethical Hacking: Black, White and Gray.** 2004. Disponível em: <https://www.sans.org/reading-room/whitepapers/hackers/paper/1390>. Acesso em: 02 maio 2004.

HAHANOV, V.; HAYFORD, A.; AHMETOGLU, A. H. Pentesting and vulnerability diagnosis. In: International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 12., 2013, Polyana Svalyava, Ukraine. **Etc.** Polyana Svalyava, Ukraine: Ieee, 2013. p. 127 - 131.

HARRIS, S. **@War: The Rise of the Military-Internet Complex.** Nova York: Eamon Dolan / Marine, 2015. 288 p.

HOMELAND SECURITY NEWSWIRE. **U.S. Cybercrime Losses Double.** Disponível em: <http://www.homelandsecuritynewswire.com/us--?cybercrime--?losses--?double>. Acesso em: 10 mar. 2019.

JAN, S.; NGUYEN, C. D.; BRIAND, L. Known XML Vulnerabilities Are Still a Threat to Popular Parsers and Open Source Systems. **2015 Ieee International Conference On Software Quality, Reliability And Security**, [s.l.], p.233-241, ago. 2015. IEEE.

JIMENEZ, R. E. L. Pentesting on Web applications using ethical - hacking. **2016 Ieee 36th Central American And Panama Convention (concapan Xxxvi)**, [s.l.], p.1-6, nov. 2016. IEEE.

KAHANER, L. **Competitive Intelligence.** New York: Touchstone, 1996.

KROLL INC (Usa). **STARWOOD Guest Reservation Database Security Incident.** 2018. Disponível em: <https://info.starwoodhotels.com>. Acesso em: 17 dez. 2018.

KUMAR, P.; PATERIYA, R. K. A survey on SQL injection attacks, detection and prevention techniques. **2012 Third International Conference On Computing, Communication And Networking Technologies (icccnt'12)**, [s.l.], p.1-5, jul. 2012. IEEE.

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet:** uma abordagem top-down. 3. ed. São Paulo: Pearson, 2005.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de metodologia científica.** 5. ed. São Paulo: Atlas, 2003.

LIMA, João P. **Administração de Redes Linux.** Goiania: Terra, 2003.

LONG, L. **Profiling Hackers.** 2012. Disponível em: <https://www.sans.org/reading-room/whitepapers/hackers/paper/33864>. Acesso em: 07 fev. 2012.

MAJUMDER, B. G. **Marriott facing lawsuits after data breach impacting up to 500 million customers.** 2018. Disponível em: <http://www.ibtimes.sg/marriott-facing-lawsuits-after-data-breach-impacting-500-million-customers-28426>. Acesso em: 03 dez. 2018.

MARTINELO, C.; BELLEZI, M. A. **Análise de Vulnerabilidades com OpenVAS e Nessus.** São Carlos: T.i.s., 2014.

MCCLURE, S.; SCAMBRAY, J.; KURTZ, G.. **Hackers Expostos.** 7. ed. São Paulo: Bookman, 2014.

O'DONOHUE, W. **Difficult Personalities:** It's Not You; It's Them. Reno, Nevada: Lucky Bat Books, 2011. 215 p.

OLIVEIRA, T.. **Testes de Segurança em Aplicações Web segundo a metodologia OWASP**. Universidade Federal de Lavras. 2012. Disponível em: <http://www.bcc.ufla.br/wp-content/uploads/2013/09/TESTES-DE-SEGURAN%C3%87A-EM-APLICA%C3%87%C3%95ES-WEB-SEGUNDO-A.pdf>

PATIL, S. et al. Ethical hacking: The need for cyber security. **2017 Ieee International Conference On Power, Control, Signals And Instrumentation Engineering (icpcsi)**, [s.l.], p.1-5, set. 2017. IEEE.

RAO, G. S. et al. Security assessment of computer networks -an ethical hacker's perspective. **International Conference On Computing And Communication Technologies**, [s.l.], p.1-5, dez. 2014. IEEE.

SCHILDT, H.; SKRIEN, D. **Programação com Java:** Uma Introdução Abrangente. São Paulo: Mcgraw-hill, 2013.

SHRIVASTAVA, A.; CHOUDHARY, S.; KUMAR, A. XSS vulnerability assessment and prevention in Web application. **2016 2nd International Conference On Next Generation Computing Technologies (ngct)**, [s.l.], p.850-853, out. 2016. IEEE.

SHU, X.; YAO, D.; BERTINO, E. Privacy-Preserving Detection of Sensitive Data Exposure. **Ieee Transactions On Information Forensics And Security**, [s.l.], v. 10, n. 5, p.1092-1103, maio 2015. Institute of Electrical and Electronics Engineers (IEEE).

SICILIANO, R. **Seven Types of Hacker Motivations.** 2011. Disponível em: <https://www.infosecisland.com/blogview/12659--?Seven--?Types--?of--?Hacker--?Motivations.html>. Acesso em: 10 mar. 2019.

SILVA, R. R. T. et al. Investigação de segurança no Moodle. **Renote**, [s.l.], v. 12, n. 2, p.1-10, 15 fev. 2015. Universidade Federal do Rio Grande do Sul.

SINGER, P. W.; FRIEDMAN, A. **Cybersecurity and Cyberwar:** What Everyone Needs to Know. Oxford: Oxford University Press, 2014. 320 p. (What Everyone Needs To Know).

SMITH, B.; YURCIK, W.; DOSS, D. Ethical hacking: the security justification redux. **Ieee 2002 International Symposium On Technology And Society (istas'02). Social Implications Of Information And Communication Technology. Proceedings (cat. No.02ch37293)**, [s.l.], p.374-379, ago. 2001. IEEE.

TORRES, A. F. F. **Os referenciais de segurança da informação e a melhoria contínua:** um caso exploratório. 2014. 92 f. Dissertação (Mestrado) - Curso de Engenharia de Informática, Instituto Superior de Engenharia do Porto, Porto, 2014.

TRABELSI, Z.; IBRAHIM, W. Teaching ethical hacking in information security curriculum: A case study. **2013 Ieee Global Engineering Education Conference (educon)**, [s.l.], p.130-137, mar. 2013. IEEE.

TURVEY, B. **Criminal Profiling:** An Introduction to Behavorial Evidence Analysis. Oxford: Elsevier, 2011.

VIEIRA, M.; ANTUNES, N.; MADEIRA, H. Using Web security scanners to detect vulnerabilities in Web services. **2009 Ieee/ifip International Conference On Dependable Systems & Networks**, [s.l.], p.566-571, jun. 2009. IEEE.

WANG, Y.; YANG, J. Ethical Hacking and Network Defense: Choose Your Best Network Vulnerability Scanning Tool. **2017 31st International Conference On Advanced Information Networking And Applications Workshops (waina)**, [s.l.], p.1-4, mar. 2017. IEEE.

W^{3}TECHS. **Usage of Web servers**. (s.d). Disponível em: <https://w3techs.com/technologies/overview/Web_server/all>. Acesso em: 10 mar. 2019.

YUN, J.; AHN, S.; CHUNG, J. W. Fault diagnosis and recovery scheme for Web server using case-based reasoning. **Proceedings Ieee International Conference On Networks 2000 (icon 2000). Networking Trends And Challenges In The New Millennium**, [s.l.], p.495-495, jul. 200. IEEE Comput. Soc.

ZICK, C. **Microsoft Report Challenges Convention Wisdom on Cybercrime Losses.** 2011. Disponível em: <http://www.securityprivacyandthelaw.com/>. Acesso em: 10 mar. 2019.