Análise e Levantamento de Vulnerabilidades de Segurança da Informação nos Portais ZZ da ZZ

Analysis and Survey of Information Security Vulnerabilities in ZZ Portals of ZZ

Análisis y levantamiento de vulnerabilidades de seguridad de la información en portales ZZ de ZZ

Ronaldo Luiz Ransan
Universidade de Caxias do Sul
rlransan@ucs.br

Marcos Vinicius Rossetto
Universidade de Caxias do Sul
mvrossetto@ucs.br

Scheila de Avila e Silva
Universidade de Caxias do Sul
sasilva6@ucs.br

Resumo

A Segurança da Informação é uma preocupação tanto em ambientes empresariais como em pesquisas científicas. Nesse sentido, o presente trabalho teve como objetivo identificar a existência de vulnerabilidades web nos portais ZZZZ, por meio de ferramentas open source. Para isso, foi realizado o escaneamento dos sites utilizando estas seis ferramentas: OpenVas, OWASP ZAP, SQLMap, Nikto, Skipfish e W3af. O documento Top Ten OWASP, junto com algumas das demais metodologias apresentadas pelo projeto OWASP, foi tomado como referência a fim de identificar o grau de risco de cada uma das cinco vulnerabilidades detectadas, bem como para sugerir uma solução genérica às implementações futuras nos sites. Verificou-se que cada uma das soluções testadas apresentou um desempenho diferente em relação à outra, sugerindo que ao realizar este tipo de trabalho é necessário utilizar o máximo de ferramentas possíveis para abranger o maior número possível de tipos de vulnerabilidades existentes.

Palavras-chave: Top Ten OWASP. Ferramentas Open Source. PenTest. Análise de Vulnerabilidades. Segurança da Informação.

Abstract

Information Security is a concern in both business environments and scientific research. In this sense, the present paper aimed to identify web vulnerabilities in the ZZZZ portals through open-source tools. For this, the sites were scanned using six selected tools: OpenVas, OWASP ZAP, SQLMap, Nikto, Skipfish, W3af. The Top Ten OWASP document, along with other methodologies presented by the OWASP project, were taken as a reference to identify the degree of risk for each of the five vulnerabilities detected. Additionally, some suggestions for generic solutions for future implementations on the sites were also presented. It was found that each of the tested solutions performed differently compared to the other, suggesting that it is necessary to use as many tools as possible to cover the largest possible number of types of existing vulnerabilities when performing this type of work.

Keywords: Top Ten OWASP. Open-Source Tools. PenTest. Vulnerability Analysis. Information security.

Resumen

La seguridad de la información es una preocupación tanto en los entornos empresariales como en la investigación científica. En ese sentido, el presente trabajo tuvo como objetivo identificar la existencia de vulnerabilidades en los portales ZZZZ, por medio de herramientas de código abierto. Para ello, los sitios fueron escaneados utilizando estas seis herramientas: OpenVas, OWASP ZAP, SQLMap, Nikto, Skipfish y W3af. El documento Top Ten OWASP, junto con algunas de las otras metodologías presentadas por el proyecto OWASP, se tomó como referencia para identificar el grado de riesgo de cada una de las cinco vulnerabilidades detectadas. Además, se buscó presentar sugerencias para futuras implementaciones en los sitios. Se encontró que cada una de las soluciones probadas se comportó de manera diferente en comparación con la otra, sugiriendo que al realizar este tipo de trabajo es necesario emplear la mayor cantidad de herramientas como sea posible para cubrir los posibles tipos de vulnerabilidades existentes.

Palabras clave: Top Ten OWASP. Herramientas de código abierto. PenTest. Análisis de vulnerabilidad. Seguridad de la Información.

Introdução

A informação é considerada um bem capaz de agregar valor. Porém nos casos em que não é bem administrada se torna onerosa para quem a detém. Considerando que o seu acesso passou a ser mais dinâmico, devido a popularização das tecnologias de informação e comunicação (TIC's), as informações estão expostas a um crescente número de ameaças e vulnerabilidades. Nesse sentido, percebe-se a necessidade de adotar estratégias que garantam a integridade e a proteção dos dados armazenados sem prejudicar sua disponibilidade (FERREIRA, 2008).

O conceito de segurança da informação é abrangente, visto que, ela está presente em toda e qualquer atividade mediada por sistemas computacionais e é influenciada por aspectos culturais, tecnológicos e legais (MAMEDE, 2006). Sua complexidade pode ser notada, por exemplo, quando se analisa a

quantidade e diversidade de indivíduos envolvidos e/ou fatores tecnológicos relacionados ao seu processo. Segundo a ABNT NBR ISO/IEC 27002:2013:

> Segurança da informação é a proteção que a informação tem de vários tipos de ameaças para garantir a continuidade do negócio, minimizando riscos ao negócio. Ela é obtida a partir de implementações de conjuntos e controles adequados, que incluem: políticas, processos, procedimentos, estruturas organizacionais e infraestrutura de TI. Esses controles precisam ser adequados, implementados, monitorados e analisados continuamente para garantir que os objetivos de segurança esperados para aquela informação sejam atendidos (ABNT NBR ISO/IEC 27002, 2013, p.??).

As análises de critérios para validação de procedimentos de segurança, a serem seguidos em meios digitais, são essenciais para proteção das informações e dados armazenados em meios computacionais. Considerando este cenário, para exemplificar a importância da análise de segurança da informação, neste trabalho foram analisados dois portais, o ZZZ1[1] e o ZZZ2[2], desenvolvidos por acadêmicos do curso de Ciência da Computação e Sistemas de Informação da ZZZ. As primeiras implementações iniciaram em 2011 e poderão ainda, passar por mudanças futuras. Os portais estão hospedados em um serviço pago, que prevê seu armazenamento e disponibilidade. A linguagem utilizada para o desenvolvimento foi PHP. O público-alvo dos portais são pesquisadores que necessitam de dados genômicos e análises relacionadas às sequências regulatórias de bactérias.

Visto que os portais foram criados por diferentes estudantes e em distintas épocas, serão analisadas as possíveis falhas de segurança que possam ter sido deixadas no decorrer de seu desenvolvimento, resultando em brechas que possam vir a expor informações restritas, permitir acessos indevidos ou até mesmo resultar em quedas nos sites oriundas de ataques ou falhas de segurança. Nesse sentido, o trabalho tem por objetivo verificar possíveis falhas e vulnerabilidades de segurança de informação nos portais do grupo de ZZZ, tendo a seguinte questão pesquisa como norteadora do

[1] Site 01

[2] Site 02

trabalho: "Existem problemas de vulnerabilidade e ameaças de segurança de informação nos portais do grupo de ZZZ da ZZZ?"

Referencial teórico

Segurança da Informação

Pode-se afirmar que a segurança da informação é a área responsável por proteger as informações ou dados de uma determinada instituição contra ataques ou vazamento (CAMPOS, 2006). Para Sêmola (2014), segurança da informação é uma área do conhecimento específica, cujo objetivo é a proteção dos ativos da informação contra acessos não autorizados, alterações indevidas ou sua indisponibilidade. Fontes (2006) e Sêmola (2014) concordam ao afirmarem que ela pode ser determinada como um conjunto de orientações, normas, procedimentos, políticas e demais ações, que tem por objetivo proteger as informações de uma dada organização ou pessoa.

Kim e Solomon (2014, p.12) conceituam a segurança da informação como "*hardware*, sistema operacional e *software*, que trabalham juntos para coletar, processar e armazenar dados para indivíduos e organizações". Nesse contexto, ainda afirmam que a segurança da informação "é a coleção de atividades que protegem os sistemas de informações e os dados armazenados neles". Segundo Engebretson (2014), um sistema de segurança da informação baseia-se em três princípios básicos: confidencialidade, integridade e disponibilidade. Se todos estes princípios forem implantados corretamente será possível satisfazer os requisitos de segurança da informação.

Para Stallings (2016) a confidencialidade é a proteção que os dados transmitidos recebem contra ataques passivos. Pode-se destacar que este pilar é responsável por garantir que a informação transmitida em uma rede receba técnicas de criptografias antes de trafegar de modo que somente o receptor da informação possa decodificá-la e acessar sua informação. O segundo pilar, a integridade, garante que a informação não seja manipulada e que mantenha suas características originais definidas pelo proprietário. Outra função da integridade é prevenir que os dados sejam corrompidos durante o processo de transmissão da mensagem entre origem e destino

(FONTES, 2006). O pilar da disponibilidade é responsável por garantir o acesso aos dados sempre que forem requisitados, devendo permitir a acessibilidade a todo *hardware*, *software* e dados através dos sistemas. A disponibilidade também se relaciona com a redução das vulnerabilidades em sistemas computacionais (CAMPOS, 2006).

Vulnerabilidades, Ameaças e Intrusões

A Vulnerabilidade pode ser classificada como uma falha que, quando não detectada e corrigida, pode comprometer um sistema ou uma determinada informação. Para Kim e Solomon (2014, p. 6), "se existir uma vulnerabilidade em um sistema, logo existirá a possibilidade de uma ameaça. Qualquer ameaça contra uma vulnerabilidade cria um risco de que um evento negativo possa ocorrer". Sêmola (2014, p. 33) define ameaças como sendo:

> agentes ou condições que causam incidentes que comprometem as informações e seus ativos por meio de exploração de vulnerabilidades, provocando perdas de confidencialidade, integridade e disponibilidade, e consequentemente causam impactos aos negócios de uma organização.

Campos (2006) define que ameaça é um agente externo ao ativo de informação, que pode vir a se aproveitar de vulnerabilidades. As ameaças podem ser classificadas de três formas: naturais, involuntárias ou voluntárias. As ameaças naturais acontecem por meio de fenômenos da natureza enquanto as ameaças involuntárias podem ser originadas por acidentes ou pela falta de conhecimento em um determinado sistema. No caso de ameaças voluntárias, pode-se dizer que acontecem de forma proposital, com o intuito de destruir, sequestrar ou visualizar informações (SÊMOLA, 2014).

Ataques ou intrusões ocorrem quando um sistema é invadido ou comprometido por um invasor, tendo acesso a dados que não estariam disponíveis ao público. Campos (2006) alega que intrusão é uma consequência da descoberta e exploração de vulnerabilidades. Kim e Solomon (2014) afirmam que o comprometimento de um sistema ao expor a intrusão

ou ataque pode ser classificado como um conjunto de ações que comprometem os três pilares da segurança de um recurso computacional.

De acordo com Weidman (2014) os ataques podem envolver várias etapas ou fases: *(i)* Coleta de informações: levantamento não intrusivo de informações do sistema-alvo, baseando-se em fontes públicas de informação; *(ii)* Varredura: etapa na qual são realizadas pesquisas acerca do sistema-alvo para descoberta de características que possam auxiliar na intrusão; *(iii)* Penetração: obtenção de acesso elevado não autorizado, por meio de brechas ou vulnerabilidades do sistema; *(iv)* Negação do serviço: comprometimento ou quedas no funcionamento do sistema ou de partes do mesmo; *(v)* Eliminação de vestígios: remoção dos registros que possam identificar o ataque ou quem o executou; *(vi)* Criação de entradas: realização de instalações de programas ocultos *(backdoors)* que permitam uma nova invasão no futuro.

As etapas citadas podem não seguir este padrão para a realização de um plano de ataque, podendo existir todas ou apenas algumas delas. Contudo, vale destacar que as fases de varredura, penetração e negação de serviço ocorrem na maioria dos ataques.

Penetration Test

O *pentest*[3], ou teste de penetração, é uma técnica que utiliza ferramentas e métodos para avaliar a segurança de um sistema através de varreduras e análises em busca de falhas de segurança da informação. As análises obtidas pelos testes são apresentadas junto com uma avaliação do seu impacto, contendo contramedidas para resolução do problema detectado (ASSUNÇÃO, 2014). Um *pentest* baseia-se em planejamento de escopo, no qual é verificado o que será testado e estipulado o tempo do teste necessário. O *hacker* irá reunir informações necessárias para a execução do teste como: escopo do projeto, objetivo, tempo de demanda dos testes, atividades a serem realizadas e custos.

Para auxiliar na realização de um teste de penetração estão disponíveis metodologias para padronização das tarefas a serem realizadas. Elas

[3] http://www.pen-tests.com/

norteiam a organização de um processo de análise de vulnerabilidades em grande escala. Aqui destacam-se três padrões: *(i) Open Source Security Testing Methodology Manual* (OSSTMM) desenvolvido pelo *Institute for Security and Open Methodologies* (ISECOM)[4]; *(ii) Information Systems Security Assessment Framework* (ISSAF) criado pelo *Open Information Systems Security Group* (OISSG)[5] e; *(iii) Open Web Application Security Project* (OWASP)[6].

A OSSTMM é considerada a metodologia mais popular para padronização de teste de penetração. Os testes são padronizados e explicados detalhadamente para que se tenham resultados reduzíveis, com base nos conceitos em segurança humana, física, sem fio, telecomunicações e de redes de dados. Sua documentação não é considerada um ferramental, portanto, as técnicas sugeridas podem ser utilizadas através de qualquer outra ferramenta de análise de vulnerabilidades compatível com os testes propostos.

A ISSAF fornece manuais de padronização desenvolvidos para a realização de *checklists* de auditoria em sistemas, contendo técnicas para análises de vulnerabilidades. Os principais testes da ISSAF estão relacionados com a segurança de rede, *host*, aplicação e banco de dados. Destaca-se que este padrão não exige conhecimento prévio em ferramentas e sistemas operacionais por disponibilizar seu manual com imagens e exemplos de uso do ferramental necessário para a realização dos testes. Sua concepção é estruturada em três grandes áreas de execução: *(i)* planejamento e preparação; *(ii)* avaliação e relatório; *(iii)* limpeza e destruição de artefatos. Nas fases de planejamento e preparação são executados os passos necessários para definir os ambientes de testes, a configuração das ferramentas de testes, os contratos e aspectos legais, a montagem da equipe de trabalho, os prazos, os requisitos e as estruturas para os relatórios finais. Na fase de avaliação são representadas as metodologias e executados os testes de penetração constituídos das atividades descritas no Quadro 1.

[4] http://www.isecom.org/

[5] http://www.oissg.org/

[6] https://www.owasp.org/

Tabela 1 – Atividades realizadas na fase de avaliação

ATIVIDADE	DETALHE
Coleta de Informações	Explorar todas as vias possíveis de ataque sobre o alvo.
Mapeamento da rede	Obtenção de informações sobre a topologia de rede do alvo. Assim, visa identificar todos os *hosts*, sistemas operacionais, *firewalls*, sistemas de detecção de intrusão, servidores e serviços, roteamento e topologia da rede.
Identificação de vulnerabilidades	Busca de falhas de rede, servidores e outros recursos.
Penetração	Exploração das vulnerabilidades identificadas anteriormente.
Acesso e Escalada de Privilégio	Elevação dos níveis de privilégios de acesso.
Enumeração	Execução de ataques a senhas, monitoramento de tráfego de rede, coleta de *cookies*, listagem de endereços de e-mail, mapeamento de redes internas entre outros.
Comprometer usuários remotos	Realização de tentativas para comprometer usuários remotos, sites e servidores de usuários.
Manutenção de acesso	Burlar os *links* de comunicação com a rede alvo para monitoramento de acessos futuros.
Cobrindo rastros	Ocultação de ferramentas ou rastros utilizados no ataque ao alvo.

Fonte: Adaptado de Bertoglio e Zorzo, (2016)

Ao final, as fases de relatório, limpeza e destruição de artefatos são responsáveis pela pós-invasão do teste alvo. O testador cria um relatório contendo todas as informações obtidas durante o ataque e destrói todos os artefatos utilizados durante a fase de avaliação (ALLEN; HERIYANTO; ALI, 2014).

De acordo com os métodos OSTMM e ISSAF existem três testes que podem ser feitos para verificar a vulnerabilidade de um sistema: caixa preta (*Black-box*), caixa branca (*White-box*) e caixa cinza (*Gray-box*). A diferença entre os tipos de testes é a quantidade de detalhes da implementação a ser testada que se possui conhecimento.

Os testes de caixa preta são aplicados quando não se tem conhecimento prévio da infraestrutura ou ambiente a ser testado. O invasor (ou o que simula a invasão) deve obter a maior quantidade de informações possíveis para conseguir descobrir os pontos fortes e fracos de um sistema. A vantagem que se pode destacar neste tipo de teste é a possibilidade de simular em modo real o ambiente de um atacante, que realiza o ataque da parte de "fora" do sistema, tendo a possibilidade de identificar possíveis vazamentos de informação. Em contrapartida, uma desvantagem deste método é a demora na realização do teste, pelo fato de quem o realiza precisar reunir informações para realizar o ataque (RIOS; MOREIRA, 2006).

O teste da caixa branca ocorre quando o testador possui conhecimento da infraestrutura a ser testada, incluindo o diagrama da rede, endereçamento IP ou outras informações complementares. Seus objetivos são mais específicos, ou seja, visam descobrir vulnerabilidades no sistema e não o vazamento de dados (PADUA FILHO, 2008).

Para os testes de caixa cinza, utiliza-se uma mescla de técnicas de caixa-preta e de caixa-branca. Nesse caso, os ataques geralmente são realizados dentro da corporação. Como exemplo, pode-se citar um funcionário de uma determinada empresa, que possui conhecimento prévio da infraestrutura e exerce ataques contra outro setor (GOODRICH; TAMASSIA, 2012).

Projetos e Normas para Segurança da Informação no Ambiente *Web*

Para auxiliar na prevenção ou identificação de riscos na *web* existem normas que contribuem para diminuição e prevenção de falhas e vulnerabilidades. Destaca-se a norma ABNT NBR ISO/IEC ICS ISBN 27032 (2015) que é dedicada somente para Diretrizes de segurança na *web*. Existe também o projeto OWASP que possui projetos e métodos que visam auxiliar programadores no desenvolvimento *web* seguro, compartilhando estudos e materiais referentes a processos de identificação das falhas de segurança.

Norma ISO/IEC 27032

A norma ISO/IEC 27032 "Tecnologia da informação -Técnicas de segurança - Diretriz para segurança cibernética" é uma norma internacional que fornece diretrizes para melhorar o estado de Segurança Cibernética, traçando aspectos típicos desta atividade e suas ramificações em outros

domínios de segurança como: informações de segurança, segurança de rede, segurança na Internet e proteção da infraestrutura de informações críticas (CIIP).De acordo com a norma ISO/IEC 27032 (2015), classifica-se como espaço cibernético um ambiente complexo, resultante da interação de pessoas, softwares e serviços na Internet que são suportados por instrumentos físicos de TIC com redes em todo mundo, ressaltando que:

> No entanto, há questões de segurança não abrangidas pela atual segurança da informação, segurança de Internet, segurança de redes e assim melhores práticas recomendadas de segurança de TIC, assim como as lacunas entre esses domínios, bem como a falha de comunicação entre as organizações e provedores no Espaço Cibernético. Isso ocorre porque os dispositivos e redes conectadas que suportam o espaço cibernético têm vários proprietários, cada um com suas próprias preocupações comerciais, operacionais e regulamentares (ABNT NBR ISO/IEC 27032, 2015, p. IX).

A norma ISO/IEC compartilha informações para coordenação e gestão de incidentes entre as partes interessadas (provedores, serviços, empresas, gestores) e usuários, fornecendo diretrizes e técnicas para tratar riscos de segurança como: ataques de engenharia social; *hacking*; proliferação de software mal-intencionado (*malware*); Software espião (*spyware*) ou outros softwares potencialmente indesejados. Além de lidar com os riscos citados anteriormente, ela contempla controles para: preparar-se contra ataques de malware, meliantes ou de organizações criminosas na Internet; detectar, monitorar e responder a ataques.

Projeto OWASP

OWASP é uma comunidade mundial *on-lin*e, na qual os participantes colaboram para que sejam criados e compartilhados livremente artigos, metodologias, documentações, ferramentas e tecnologias voltadas para a área de desenvolvimento e segurança *web*. É uma organização sem fins lucrativos que não possui intervenção de empresas privadas da área. Ela colabora com corporações, universidades, agências governamentais e outras organizações que buscam utilizar seus métodos na busca de soluções voltadas à segurança da informação (MEUCCI, 2008). Dentre vários projetos disponibilizados pela OWASP, destacam-se:

- *Testing Guide*: é um guia que auxilia em questões como: o que, o porquê, quando e como testar aplicações *web* em busca de vulnerabilidades.
- *Code Review Guide*: é um guia que sugere boas práticas na revisão de códigos com foco em detectar vulnerabilidades.
- *Development Guide*: é um projeto que visa realizar documentação durante o desenvolvimento de aplicações voltadas para *web*. É composto por vários tópicos referentes à segurança e por soluções para falhas encontradas.
- *Top Ten*: Trata-se de um documento que apresenta e explica as dez principais vulnerabilidades de aplicações *web*, demonstrando as consequências ao explorá-las e correções que poderiam protegê-las.

O Quadro 2 apresenta a lista *Top Ten* na qual são listadas as vulnerabilidades mais recorrentes e uma breve descrição de cada uma.

Tabela 2 - Vulnerabilidades Web reportadas pela OWASP Top Ten

VULNERABILIDADE	DESCRIÇÃO
A1: Falhas de Injeção	Falhas de injeção (*SQL Injection*) ocorrem quando os dados são enviados a um interpretador com partes de comandos ou consultas. A informação enviada pelo atacante burla o interpretador de consultas que irá executar o comando malicioso.
A2: Falhas de Autenticação	Funções relacionadas ao gerenciamento de autenticação de sessão, que quando implementadas incorretamente, permitem que o invasor explore falhas que permitam a utilização de identidade de outro usuário.
A3: Exposição de dados sensíveis	Os atacantes podem interceptar dados sigilosos e utilizá-los em crimes. Dados sensíveis exigem métodos de proteção adicionais como, por exemplo, a criptografia para dados em trânsito.
A4: Entidades externas XML (XXE)	Processadores XML obsoletos ou mal configurados podem ser utilizados para varredura de portas internas, execução remota de códigos, ataques de negação de serviço ou compartilhamento de arquivos internos.
A5: Perda de controle de Acesso	Os invasores podem utilizar falhas nas restrições de usuários para acessar funcionalidades e/ou dados não autorizados.

A6: Configurações incorretas de Segurança	A configuração incorreta de segurança se dá ao serem aplicadas configurações padrão, incompletas ou a falta delas. Alguns exemplos são: cabeçalho HTTP configurado incorretamente, mensagens de erro com conteúdo sensível, falta de atualizações, entre outros.
A7: Cross Site Scripting (XSS)	Falhas XSS ocorrem quando uma aplicação recebe dados não confiáveis e os envia ao navegador sem nenhuma validação ou filtro.
A8: Desserialização Insegura	A desserialização insegura geralmente leva à execução remota de códigos ou podem ser usadas para executar ataques de repetição, injeção ou elevar seus privilégios de execução.
A9: Componentes com vulnerabilidades conhecidas	Aplicativos e APIs que utilizam componentes com vulnerabilidades conhecidas podem enfraquecer as defesas de aplicativos e resultar em vários ataques e impactos.
A10: Registro e Monitoramento Insuficiente	Registro e monitoramento insuficientes associados à integração ausente ou ineficaz permitem aos atacantes manter ataques persistentes.

Disponível em https://owasp.org/www-project-top-ten/

A lista *Top Ten* disponibilizada é construída de acordo com a frequência com que as vulnerabilidades são detectadas nas aplicações testadas. Para seu desenvolvimento é numerada cada instância em que uma vulnerabilidade é encontrada; na segunda etapa é feita a contagem de aplicativos em que cada vulnerabilidade é detectada (uma ou mais vezes). Segundo a OWASP *Top Ten*, a lista permite comparar testes realizados por ferramentas com testes realizados por ferramentas acompanhadas por testadores.

Ferramentas de análise de vulnerabilidade Web

Existem diversas ferramentas para análise de vulnerabilidades em ambientes *web* licenciados ou de código aberto. O site SecTools[7] mantém um ranking das ferramentas de segurança e análise de vulnerabilidades segundo especialistas da área (MARTINELO; BELLEZI, 2014). Das ferramentas disponíveis no site, procurpu-se analisar as ferramentas de código aberto (*open source*), considerando as seguintes características: *(i)* Classificam as falhas encontradas e as listam; *(ii)* Mostram como um atacante poderia fazer uso da falha detectada; *(iii)* Apresentam sugestões para a correção das

[7] https://sectools.org/

vulnerabilidades encontradas; *(iv)* Disponibilidade de interface gráfica e *(v)* levantamento total de todo o mapa do site (*Crawling*). O quadro 3 apresenta a análise comparativa de ferramentas selecionadas.

Tabela 3 – Lista de Ferramentas

FERRAMENTA	*POS. SECTOOLS*	NÍVEIS DE VULNERABILIDADES DETECTÁVEIS	INTERFACE GRÁFICA?	*CRAWLING*
OpenVas	19º	A6	SIM	NÃO
Nikto	14º	A1 - A3 - A5 - A7	NÃO	SIM
OWASP Zap	12º	A1 - A2 - A3 - A4 - A5 - A6 - A7 - A8 - A10	SIM	SIM
W3AF	18º	A1 - A2 - A3 - A4 - A5 - A6 - A7 - A8 - A10	AMBAS	SIM
SqlMap	30º	A1 - A7 - A8	NÃO	NÃO
SkipFish	39º	A1 - A7 - A8	NÃO	SIM

Fonte: elaborado pelo autor baseado no site SecTools.org

A ferramenta *Nessus scanner* é um utilitário de código proprietário, a partir do qual o *OpenVas*[8] ramificou-se para permanecer aberto. O *OpenVas* é dividido em dois componentes principais: *scanner* e gerente. O *scanner* pode ser "apontado" para um alvo e se dedicar em descobertas de vulnerabilidades para o gerente. Também gera os dados obtidos através de vários *scanners* e algoritmos, aplicando sua própria inteligência artificial para criar um relatório detalhado. O *OpenVas* pode ser utilizado para rastreio de vulnerabilidade em redes TCP, mas também é utilizado para busca de falhas em ambientes *web*, sendo considerada uma ferramenta estável e confiável.

A ferramenta *Nikto*[9], por suportar protocolos HTTP e HTTPS, é frequentemente utilizada na verificação de sites. De código aberto, ela fornece resultados de modo interativo e detalhado, utilizando uma técnica denominada de mutação, pela qual é criada uma combinação de vários testes HTTP em conjunto para formar um ataque, com base na configuração do servidor *Host* e do código em que o site foi concebido. Desta maneira, ele

[8] http://www.openvas.org/

[9] https://cirt.net/nikto2/

busca brechas críticas, tais como erros de configuração de *upload* de arquivos, manipulação de *cookies* indevidos, erros de *cross-scripting*, entre outros.

A ferramenta *OWASP Zap*[10] foi desenvolvida pela OWASP com a finalidade de detectar vulnerabilidades em aplicações *web*. É mantida pelo projeto OWASP e aplicável em diversos sistemas operacionais. Como características principais, destacam-se a capacidade de realização de *crawler*, possuir *scanner* automático e abranger as principais vulnerabilidades reportadas pelo *Top Ten*.

O *W3AF*[11] é uma ferramenta de código aberto e conta com recursos para busca de injeção de *SQL*, *cross site scripting* (XSS), inclusão de arquivos local e remota, analisador estático de código PHP, entre outras funcionalidades. Adicionalmente oferece a opção para busca a partir de perfis, como no caso da lista *Top Ten* da OWASP.

O *SqlMap*[12] é capaz de explorar falhas de injeção SQL, tanto na aplicação quanto na avaliação do gerenciador do banco de dados. Com suporte a diversos tipos de bancos dados, ele também pode realizar ataques de adivinhação de senhas podendo ser combinado com outras ferramentas para complemento de varreduras em busca de falhas.

A ferramenta *Skipfish*[13] realiza análise de vulnerabilidades em ambientes *web* através do processo de *crawling*. Entre suas características destacam-se: *(i)* ser *open source*; *(ii)* bom desempenho na realização de seus testes otimizando recursos disponíveis; *(iii)* contempla a maioria das vulnerabilidades da lista OWASP Top Ten.

Metodologia

Na etapa inicial foi realizada uma pesquisa referente aos tipos de vulnerabilidades no ambiente *web* e normas de segurança da informação, unindo aspectos técnicos e legais, os quais ajudaram a validar a proposta. Para

[10] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project/

[11] http://w3af.org/

[12] http://sqlmap.org/

[13] https://code.google.com/archive/p/skipfish/

o desenvolvimento do trabalho, o projeto OWASP foi utilizado como guia para auxiliar nos quesitos relacionados aos tipos de vulnerabilidades *web* existentes, ferramentas e métodos empregados para testes, assim como soluções para as falhas detectadas. Um dos guias utilizado foi o documento OWASP *Top Ten*, que serviu como referência para catalogação e indicação do grau de risco ao qual determinada vulnerabilidade detectada nos testes pertencia.

Para a realização dos testes nos sites foi escolhido o Teste de Penetração, que informa os usuários ou profissionais na área de segurança da informação sobre como devem realizar as etapas de um teste. Quanto às ferramentas, utilizou-se como caráter de escolha o fato de serem *open source*, a quantidade de informações disponíveis sobre cada uma delas e a sua respectiva colocação no *ranking* do site *SecTool* (Quadro 3). Por fim, os resultados obtidos foram catalogados e apresentados com uma breve descrição do problema e com possíveis medidas que poderão ser adotadas em trabalhos futuros a fim de corrigir tais vulnerabilidades. O fluxograma apresentado na Figura 1 resume as etapas seguidas para a execução do trabalho.

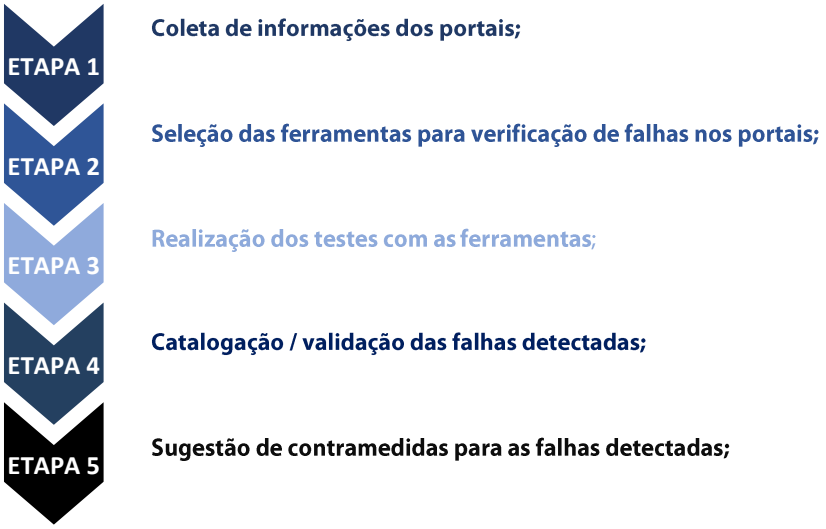

Figura 1 – Etapas realizadas para os testes
Fonte: Autores

Para realizar os testes com as ferramentas utilizaram-se os seguintes recursos de *hardware* e *software*:

- Processador: Intel® Core™ i7 – 920M CPU @2.67 GHz;
- Memória instalada (RAM): 12.00GB;
- Tipo de sistema: Sistema Operacional de 64 bits, processador com base em x64;
- Espaço em disco: 240 Gb SSD;
- Sistema Operacional: Microsoft Windows 10;
- Emulador Virtual Box;
- Kali Linux 2019;

Finalizando as etapas de estudo, seguiu-se para a configuração e validação do ambiente que continha as ferramentas para realizar as análises de vulnerabilidades. Para a construção do ambiente de testes, optou-se por utilizar um gerenciador de máquinas virtuais de distribuição livre, o *VirtualBox*. Nele foi instalada a distribuição *Kali Linux*, que serviu como sistema operacional base para utilização das ferramentas, sendo ela baseada em Debian. Esta versão do *Linux* é voltada principalmente para auditoria e segurança de computadores em geral, sendo mantida pela *Offensive Security LTDA*. A sua escolha baseou-se na quantidade de referências disponíveis sobre ela e por possuir a maioria das ferramentas pré-instaladas. Todas as ferramentas utilizadas nos testes, inclusive a versão do *Linux*, foram devidamente atualizadas e configuradas para suas versões mais recentes.

Resultados e discussão

A seguir apresentam-se os resultados obtidos com cada uma das ferramentas, bem como a catalogação das vulnerabilidades reportadas por elas.

Coleta de Informações do alvo

A primeira tarefa realizada nesta etapa foi coletar informações, como endereço de IP, *DNS*, servidores em que os sites estão hospedados, locais de consultas, área de acesso restrito, entre outras. Para isso, foi necessária a utilização das ferramentas *Maltego*[14] e *Burp Suit*[15]. A ferramenta *Maltego* tem

[14] https://www.maltego.com/

[15] https://portswigger.net/burp

por objetivo principal o levantamento gráfico do site, coletando todas suas referências e ligações com servidores DNS e estruturas de rede. Embora seja uma ferramenta proprietária, alguns recursos básicos são mantidos gratuitamente para utilização. Por sua vez, a utilização do *software BurpSuit* teve por objetivo a análise dos métodos *post* e *get* dos sites, ou seja, verificou-se através dele todo o conteúdo que era transmitido entre usuário e o servidor *web*, a fim de identificar falhas e métodos utilizados na troca de requisições.

Através da etapa de coleta de informações referentes aos portais foi possível identificar que os dois portais estão em um único *host*. Do ponto de vista do usuário, cada site é um portal "diferente", mas para o servidor, os sites estão contidos em apenas um *host*. Isso fez com que os testes fossem realizados apenas no host principal, ou seja, eles sempre foram realizados a partir da URL "http://zzz.zzz.com/". Outro ponto importante detectado nessa etapa é que os portais estão hospedados em um servidor do tipo compartilhado, existindo outros sites compartilhando os mesmos recursos de rede, *software* e *hardware*.

Realização dos testes com as ferramentas

O *penetration test* foi realizado utilizando-se as seguintes ferramentas: *OpenVas*, *Nikto*, *OwaspZap*, *WA3AF*, *SqlMap* e *SkipFish*. Com elas pode-se ampliar a gama de vulnerabilidades analisadas, conforme Quadro 4 que indica, adicionalmente, o tempo de execução de cada uma.

Quadro 4 - Lista de Ferramentas/com tempos

FERRAMENTA	*POS. SECTOOLS*	RELAÇÃO LISTA *TOP TEN*	INTERFACE?	*CRAWLING*	TEMPO/TESTE
OpenVas	19º	A6	SIM	NÃO	5 min/35 seg.
Nikto	14º	A1 - A3 - A5 - A7	NÃO	SIM	45 min
OWASP Zap	12º	A1 - A2 - A3 - A4 - A5 - A6 - A7 - A8 - A10	SIM	SIM	7 h / 6 min
W3AF	18º	A1 - A2 - A3 - A4 - A5 - A6 - A7 - A8 - A10	AMBAS	SIM	8 h / 23 min
SqlMap	30º	A1 - A7 - A8	NÃO	NÃO	2 h / 49 min
SkipFish	39º	A1 - A7 - A8	NÃO	SIM	1 h / 39 min

Fonte: elaborado pelo autor

Vulnerabilidades detectadas

Após a finalização dos testes guiados por ferramentas, seguiu-se para a etapa de realização dos testes manuais das vulnerabilidades reportadas pelos *softwares*, a fim de validar e excluir possíveis falsos positivos. Após os testes, foram agrupadas as falhas identificadas por mais de uma ferramenta, conforme o Quadro 5.

Quadro 5 - Lista de Vulnerabilidades detectadas

ID	NOME VULNERABILIDADE	GRAU DE RISCO	FERRAMENTAS QUE A IDENTIFICOU	PREVENÇÃO
V1	*X-Frame-Options Header Not Set* (Cabeçalho X-Frame-Options não Definido)	Média - (A5)	W3AF Nikto OWASP Zap	Enviar cabeçalhos de resposta referentes à diretiva de ancestrais de estrutura *Content Security Policy* (CSP)
V2	*Application Error Disclosure* (Divulgação de erro de aplicativo)	Média - (A3)	OWASP Zap	Tratar todas as mensagens de erros do site de forma genérica ou personalizada
V3	*X-Content-Type-Options Header Missing* - (Cabeçalho de opções de tipo de conteúdo X ausente)	Média - (A3)	W3AF Nikto OWASP Zap	Definir o cabeçalho X- Content-Type-Options como *'nosniff'*
V4	*Form Cleartext Password* - (Senha exposta)	Média - (A3)	W3AF	Utilizar métodos de criptografia de dados sensíveis em trânsito
V5	*Shared Hosting* (Servidor compartilhado)	Não possui	W3AF	Não utilizar servidor compartilhado

Fonte: elaborado pelo autor

Depois de realizada a validação e catalogação das falhas detectadas, apresenta-se uma breve descrição dos problemas encontrados e possíveis medidas que poderão ser adotadas para sua correção.

Vulnerabilidade V1: X-Frame-Options Header Not Set (Cabeçalho X- Frame-Options não Definido)

Esta vulnerabilidade é mais conhecida como "*Clickjacking*", que consiste em induzir um usuário da *web* a utilizar uma página falsa. Esse tipo de ataque pode ser usado isolado ou em combinação com outros tipos de vulnerabilidades, podendo, por exemplo, enviar comandos não autorizados ou revelar informações restritas enquanto a vítima está interagindo com a página *web* falsa. Este tipo de ataque utiliza de recursos HTML e *Javascript* para forçar a vítima a executar ações indesejadas, como clicar em um botão que executa outra operação ou fornecer suas credenciais para acesso a alguma parte restrita do site.

O invasor cria uma página web aparentemente inofensiva e semelhante a página original, que carrega o aplicativo de destino por meio do uso de "*iframe*". Com isso, o invasor induz a vítima a interagir com a página fictícia por outros meios, como por exemplo, engenharia social. Esta falha acontece do "lado cliente", mas pode ser evitada através de implementações através de códigos de programação.

De acordo com a documentação OWASP, é possível evitar este tipo de falha por meio do envio de cabeçalhos de resposta referentes à diretiva de ancestrais de estrutura *Content Security Policy* (CSP) que instruem o navegador a negar o enquadramento de outros domínios ao site, fazendo com que cabeçalhos HTTP mais antigos sejam substituídos pelo *X-Frame-Options*, garantindo que o conteúdo deles não seja incorporado a outros sites. Também é possível preveni-la adicionando código defensivo na interface do usuário para garantir que o quadro atual do site seja a janela de nível mais alto a ser utilizada.

Vulnerabilidade V2: Application Error Disclosure (Divulgação de erro de aplicativo)

Esta vulnerabilidade consiste no não tratamento de erros resultantes do uso do site, que podem expor informações importantes para um possível atacante, como nome de tabelas ou de campos de um determinado banco de

dados. Com posse destas informações, um *hacker* utilizando ferramentas voltadas à injeção de SQL, pode realizar um ataque mais concentrado ao site.

Para prevenção, sugere-se tratar todas as mensagens de erros do site de forma genérica ou personalizada, a fim de não expor mensagens que possuam informações de nível técnico aos usuários. O tratamento deverá ser realizado de acordo com o tipo da linguagem empregada para o desenvolvimento do site, aplicando o método mais adequado a ela.

Vulnerabilidade V3: X-Content-Type-Options Header Missing (Cabeçalho de opções de tipo de conteúdo X ausente)

Quando o cabeçalho de *Anti-MIME-Sniffing X-Content-Type-Optio*ns não é configurado para *"nosniff"* permite que versões mais antigas dos navegadores executem *sniffing MIME* no corpo da resposta, fazendo com que ele seja interpretado e exibido como um tipo de conteúdo diferente do tipo declarado. Quando o navegador carrega um determinado arquivo ele verifica seu conteúdo para determinar qual tipo de arquivo está sendo enviado. Por exemplo, se o site possuir recurso para *upload* de arquivos do tipo PNG, e este arquivo for modificado e possuir instruções HTML do tipo malicioso ao seu conteúdo, o navegador ou o servidor poderá executá-lo, vindo a comprometer a segurança do site.

Para sua prevenção, sugere-se revisar a configuração do aplicativo ou o servidor *web* para que esteja configurado com cabeçalho *"Content-Type"* apropriado e que defina o cabeçalho *X-Content-Type-Options* como *"nosniff"* para todas as páginas da *web*.

Vulnerabilidade V4: Form Cleartext Password (Senha exposta)

Consiste em armazenar ou transmitir senhas em texto puro, sem nenhum tipo de encapsulamento ou criptografia para proteger os dados no processo de requisição de um acesso. Isso pode fazer com que ao utilizar programas que monitoram o tráfego da rede, capturem e identifiquem informações sensíveis como senhas e demais informações que deveriam permanecer "ocultas" neste processo.

Para solucionar essa vulnerabilidade sugere-se a utilização de métodos de criptografia de dados sensíveis em trânsito, como o TLS, fazendo com que o servidor utilize parâmetros seguros na troca de

informações. Sugere-se também o uso de encriptação utilizando diretivas *HTTP Strict Transport Security* (HSTS).

Vulnerabilidade V5: Shared Hosting - (servidor compartilhado)

Embora não seja considerada uma vulnerabilidade, a ferramenta W3AF a reportou. A hospedagem compartilhada é indicada em sites de pequeno porte ou que possuem poucos acessos, uma vez que seu custo é menor e não demanda alta disponibilidade. No entanto, caso o site venha a receber uma grande quantidade de acessos simultâneos não previstos, alguns servidores poderão se tornar instáveis, fazendo com que o site fique fora do ar por determinados períodos ou até mesmo seja retirado do ar por tempo indeterminado.

Vale ressaltar também que este tipo de hospedagem possui tecnologia inflexível, ou seja, a aplicação deve ser desenvolvida de acordo com os recursos disponibilizados pelo servidor, como por exemplo, tipo de banco de dados ou linguagem de desenvolvimento. Outro problema que pode ocorrer é em relação à segurança do servidor, visto que se algum dos sites for atacado, os demais podem ser comprometidos.

Considerações finais

O objetivo do trabalho foi realizar um levantamento de possíveis vulnerabilidades existentes nos portais ZZZ. A proposta foi realizada através da utilização do projeto OWASP *TopTen* e de ferramentas para realização dos Testes de Penetração.

Em relação à utilização de ferramentas para os testes, pode-se concluir que somente a sua utilização não garante um grau de confiabilidade sobre a existência ou não de vulnerabilidades, visto que elas podem retornar diversos falsos positivos, tendo que assim realizar intervenções manuais para validá-las. Também é importante salientar que a proposta do trabalho se baseou em apenas utilizar ferramentas de código aberto, sem a utilização de ferramentas proprietárias, tendo assim uma possível diferença de resultados, se ambas forem utilizadas e comparadas.

Observou-se também a não utilização de proteções denominadas *captchas* (acrônimo para *Completely Automated Public Turing Test to Tell*

Computers and Humans Apart), em todos os formulários do site e locais de solicitação de *login*, crucial para verificar se quem está utilizando é um computador ou humano, a fim de se evitar o preenchimento de formulários automatizados resultando em possíveis sobrecargas ao site.

Como limitação de estudo, destaca-se que os sites testados apresentaram instabilidade no processo de verificação das vulnerabilidades, inviabilizando a realização de mais testes, já que as ferramentas no seu processo de scanner sobrecarregam o servidor. Em relação às ferramentas listadas na primeira etapa do trabalho, observou-se que algumas delas, mesmo cumprindo com os pré-requisitos, ao utilizá-las não apresentaram um bom desempenho, tendo que serem substituídas por outras. De modo geral, as demais ferramentas cumpriram com seu objetivo, reportando vulnerabilidades que foram validadas através de testes manuais, a fim de comprovar sua existência.

Como ações futuras, recomenda-se a análise dos resultados obtidos neste trabalho, a fim de realizar as implementações de acordo com a linguagem utilizada para o desenvolvimento do site. Depois de efetuadas as correções, recomenda-se uma nova realização dos testes utilizando as mesmas ferramentas, a fim de verificar se as falhas não serão mais detectadas. Outro ponto importante que vale ressaltar é a existência de mais ferramentas *open source* voltadas à verificação de vulnerabilidades que não foram elencadas, mas que podem ser analisadas e utilizadas para testes futuros.

Referências

ABNT (Associação Brasileira de Normas Técnicas). *NBR 27002*: Tecnologia da informação - Técnicas de Segurança - Sistema de gestão da segurança da informação. Rio de Janeiro: ABNT, 2013.

ABNT (Associação Brasileira de Normas Técnicas). *NBR 27032*: Tecnologia da Informação - Técnicas de segurança - Diretriz para segurança cibernética. Rio de Janeiro: ABNT, 2015.

ALLEN, Lee; HERIYANTO, Tedi; ALI, Shakeel. Kali. *Linux*: assuring security by penetration testing. Birmingham: Packt Publishing Ltd, 2014.

ASSUNÇÃO, M. F. *Segredos do Hacker Ético*. 5. ed. Florianópolis: Visual Books, 2014.

BERTOGLIO, D. D.; ZORZO, A. F. Tramonto: uma estratégia de recomendações para testes de penetração. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, 16, 2016, Niterói. *Anais.... Niterói*: Sociedade Brasileira de Computação, 2016.

CAMPOS, A. L. N. *Sistema de segurança da informação*: controlando os riscos. Florianópolis: Visual Books, 2006.

ENGEBRETSON, P. *Introdução ao Hacking e aos Testes de Invasão*. São Paulo: Novatec, 2014.

FERREIRA, F. N. F.; ARAUJO, M. T. *Política de segurança da informação*: guia prático para elaboração e implementação. Rio de Janeiro: Ciência Moderna, 2008.

FONTES, E. *Segurança da informação*: o usuário faz a diferença. São Paulo: Saraiva, 2006.

GOODRICH, Michael T.; TAMASSIA, Roberto. *Introdução à segurança de computadores*. Porto Alegre: Bookman, 2012.

MAMEDE, H. S. *Segurança Informática nas Organizações*. Lisboa: FCA - Editora de Informática, 2006.

MARTINELO, C. A. G.; BELLEZI, M. A. Análise de vulnerabilidades com OpenVAS e Nessus. *T.I.S - Tecnologias, Infraestrutura e Software*. v. 3, n. 1, p. 34-44, 2014.

MEUCCI, M. *Owasp testing guide version 3.0*. OWASP Foundation, 2008. Disponível em: <https://owasp.org/www-pdf-archive/OWASP_Testing_Guide_v3.pdf >. Acesso em: 20 de nov. de 2020.

PADUA FILHO, W. de P. *Engenharia de software*. 3. ed. Rio de Janeiro: LTC, 2008.

RIOS, E.; MOREIRA, T. *Teste de Software*. 3, ed. Rio de Janeiro: Alta Books Editora, 2006.

SÊMOLA, M. *Gestão da Segurança da Informação*: uma visão executiva. 2. ed. Porto Alegre: Elsevier (2014).

STALLINGS, W. *Redes e sistemas de comunicação de dados*. Rio de Janeiro: GEN LTC, 2016.

KIM, D.; SOLOMON, M. G. *Fundamentos de segurança de sistemas de informação*. Rio de Janeiro: LTC, 2014.

WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking*. San Francisco: No Starch Press, 2014.