ENTRE O ENSINO E A INVASÃO: ANÁLISE DE REQUISITOS DE SEGURANÇA CIBERNÉTICA EM PLATAFORMAS DIGITAIS EDUCACIONAIS

Caio César Jordão Paccagnella (PIC/UEM), Luciana Andréia Fondazzi Martimiano (Coorientadora), Alisson Renan Svaigen (Orientador). E-mail: ra138549@uem.br.

Universidade Estadual de Maringá, Centro de Tecnologia, Maringá, PR.

Área e subárea do conhecimento: Ciência da Computação/Sistemas de Computação

Palavras-chave: Segurança. Teste de Invasão. Educação.

RESUMO

A adoção de plataformas digitais acadêmicas exige avaliação dos requisitos de segurança cibernética para resguardar aspectos de confidencialidade, integridade e disponibilidade dos dados institucionais. Neste sentido, este trabalho apresenta uma análise de segurança cibernética baseado em teste de invasão (*penetration testing*) no ambiente virtualizado de homologação de uma plataforma de gestão acadêmica, conduzida sob o modelo *black-box* mediante autorização e termo de confidencialidade. A partir de uma revisão sistemática da literatura, foi estabelecida uma metodologia baseada no ciclo Backtrack, englobando etapas de *footprinting, fingerprinting*, varredura automatizada e validação manual dos achados. Os resultados revelaram aspectos críticos, como o uso de tecnologias obsoletas e vulnerabilidades de *cross-site scripting*, mas também indicaram boas práticas de segurança cibernética, principalmente voltadas à mitigação de ataques de injeção. Conclui-se que os mecanismos de segurança da plataforma, podem ser substancialmente elevados, garantindo a proteção da comunidade acadêmica.

INTRODUÇÃO

Diversas instituições de ensino estão adotando plataformas digitais educacionais para gerenciar as atividades pertencentes ao próprio escopo. Nessa direção, é esperado que esses sistemas possuam o mínimo de vulnerabilidades possíveis. Uma forma de garantir esse requisito é por meio de testes de penetração (*penetration testing*), que consistem na simulação controlada de técnicas de ataque com o objetivo de identificar vulnerabilidades e avaliar os riscos associados a elas (MORENO, 2015). Seguindo uma metodologia chamada *backtrack*, é possível simular um ataque real ao sistema, de forma que seja possível prevenir futuros ataques reais mitigando vulnerabilidades.

Diante disso, o objetivo deste trabalho é avaliar a segurança cibernética de um sistema educacional[1] por meio da aplicação de testes de penetração. Para a realização dos testes, a entidade responsável pelo sistema autorizou os testes em ambiente controlado sob um termo de confidencialidade por ambas as partes.

MATERIAIS E MÉTODOS

Inicialmente, foi realizada uma revisão sistemática[2] a fim de adquirir o arcabouço teórico relacionado aos testes de penetração. Baseado no arcabouço, o modelo de teste *black-box* (caixa-preta) foi definido, no qual os auditores operam sem conhecimento prévio da infraestrutura interna, das tecnologias empregadas ou do código-fonte do sistema. Além disso, para a realização dos testes de penetração, foi utilizada uma máquina virtual com a distribuição Kali Linux (configurada com 8 GB de RAM e 4 vCPUs), sustentada por um computador hospedeiro com processador Intel Core i5 de 13ª geração e 32 GB de memória RAM.

Os procedimentos operacionais foram estruturados com base nas fases da metodologia *Backtrack*, que consiste em: Planejamento do projeto, *Footprinting*, *Fingerprinting*, Enumeração, Mapeamento e Análise de vulnerabilidades, Exploração do alvo, Engenharia social (opcional), Escalonamento de privilégios, Manutenção do acesso, DoS – Negação de serviço, Documentação técnica e Redes sem fio (opcional) (MORENO, 2015). Devido ao tempo despendido para o projeto, foram realizadas as primeiras cinco etapas descritas, restando a execução das demais em trabalhos futuros. O planejamento envolveu a autorização formal da instituição mediante termo de confidencialidade, estabelecendo a abordagem black-box para a condução das análises em ambiente de homologação. Na fase de *footprinting* (reconhecimento), realizou-se o levantamento de diferentes características do sistemas, como os registros DNS, os subdomínios, o mapeamento de blocos de IP e a análise de rotas. Na sequência, as fases de *fingerprinting* e de enumeração identificaram os serviços e a pilha tecnológica ativa. A estrutura interna de diretórios e arquivos foi mapeada via força bruta e *fuzzing* (MIESSLER, 2021).

Por fim, a fase de mapeamento e análise de vulnerabilidades iniciou-se com uma varredura automatizada efetuada pelo scanner OWASP ZAP (versão 2.17.0) configurado em modo *Safe*. Com o objetivo de eliminar falsos positivos e atestar a real explorabilidade das falhas, todos os alertas de alto risco foram submetidos à validação manual e classificados conforme os padrões do OWASP Top 10 (OWASP, 2021).

RESULTADOS E DISCUSSÃO

[1] Devido ao termo de confidencialidade, a IES, o sistema educacional e a entidade responsável estão anonimizadas neste trabalho.
[2] Sumarização da revisão disponível pelo link: https://tinyurl.com/paccagnellaetal2026

A fase de reconhecimento e *fingerprinting* revelou o endereço IP em que a aplicação está hospedada, apresentando apenas as portas 80 e 443 acessíveis. A análise de tráfego realizada pelo utilitário Wafw00f não identificou a presença de um *Web Application Firewall* (WAF), expondo o servidor diretamente a requisições sem filtragem intermediária na camada de aplicação. Isso não indica necessariamente que não há controle de tráfego, mas dá indícios de que ele não acontece nesse nível. Em relação às tecnologias utilizadas na aplicação, mapeou-se a utilização do servidor web Apache 2.4.62 (AlmaLinux) com módulo conector mod_jk 1.2.50, integrado a um servidor de aplicação JBoss 5.0 (JBossWeb-2.1) e Servlet Java 2.5. Destaca-se que a versão do JBoss identificada é obsoleta, representando um fator crítico de risco. Adicionalmente, verificou-se que o *cookie* de sessão JSESSIONID não possui a *flag* HttpOnly configurada, permitindo que seu valor seja lido via *scripts* no lado do cliente caso existam falhas de injeção (OWASP, 2021). O mapeamento estrutural efetuado pelas ferramentas Gobuster e FFUF revelou diretórios restritos.

A varredura automatizada efetuada pelo scanner OWASP ZAP (OWASP ZAP, 2024) sinalizou inicialmente 125 alertas classificados como de alto risco. Contudo, a etapa de validação manual mostrou a relevância do teste prático para a eliminação de falsos positivos. As 115 instâncias de *SQL Injection* apontadas pelo *scanner*, foram separadas em grupos baseados no parâmetro, método HTTP e rota indicada. Cada grupo foi testado separadamente via utilitário SQLMap. Os ensaios automatizados de injeção baseada em tempo e consultas UNION indicaram que nenhum dos parâmetros era injetável, reclassificando todos esses alertas como não exploráveis.

Em contrapartida, as vulnerabilidades de *Cross-Site Scripting* (XSS) foram validadas com sucesso. A falha do tipo *DOM-Based XSS* foi confirmada, com a injeção de *payload* no endereço indicado pelo ZAP, e resultou na execução de *script* diretamente no navegador, sem necessidade de reflexão pelo servidor. O *Reflected XSS* foi validado, destacando-se a submissão de um *payload* personalizado desencadeou a execução da caixa de diálogo. A presença dessas falhas, combinada à ausência da flag HttpOnly no *cookie* de sessão, viabiliza cenários graves de sequestro de sessão e redirecionamento malicioso.

Por fim, a avaliação das bibliotecas do sistema atestou o uso das versões desatualizadas 1.5.0 e 1.6.0.3 da Prototype.js. O teste da vulnerabilidade ReDoS por meio de *script* customizado registrou tempos de execução significativos, mostrando que a falha é explorável na prática no ambiente de execução testado. Os ensaios referentes à CVE-2008-7220 indicaram a ocorrência de comunicação AJAX assíncrona para origem externa, a qual foi bloqueada pelo mecanismo de Compartilhamento de Recursos entre Origens (CORS) dos navegadores modernos, mas permanece como vetor de risco para ambientes legados.

CONCLUSÕES

A auditoria de segurança no ambiente de homologação da plataforma permitiu mapear a superfície de ataque da plataforma sob a perspectiva *black-box*. A etapa de validação manual provou-se indispensável ao descartar 115 alertas de *SQL Injection* apontados pela varredura automatizada, confirmando-os como indicações imprecisas. Em contrapartida, foram confirmadas vulnerabilidades efetivas de XSS que, somadas às bibliotecas JavaScript desatualizadas, evidenciaram a necessidade de readequação de componentes legados. Conclui-se que a segurança da plataforma pode ser melhorada pela implementação de *Output Encoding* no servidor e sanitização do DOM via métodos seguros como textContent para contenção do XSS (OWASP, 2021), aliada à ativação obrigatória da *flag* HttpOnly nos *cookies* de sessão para impedir o roubo de credenciais. Recomenda-se, ainda, a atualização imediata da infraestrutura legada. A implementação dessas mitigações é fundamental para assegurar a confidencialidade, a integridade e a disponibilidade dos dados da comunidade acadêmica. Para trabalhos futuros, serão executadas as demais fases do *Backtrack*, que são: Exploração do alvo, Escalonamento de privilégios, Manutenção do acesso, DoS – Negação de serviço, e Documentação técnica.

AGRADECIMENTOS

À Fundação Araucária, pelo financiamento inicial do projeto, à entidade universitária que forneceu acesso ao sistema, permitindo a análise da plataforma utilizada.

REFERÊNCIAS

MORENO, Daniel. **Introdução ao pentest.** São Paulo: Novatec Editora, 2015.

OWASP FOUNDATION. A06:2021 – **Vulnerable and Outdated Components.** OWASP Top 10, 2021. Disponível em: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.

OWASP FOUNDATION. **Cross Site Scripting Prevention Cheat Sheet.** OWASP Cheat Sheet Series, 2021. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.

OWASP FOUNDATION. **DOM based XSS Prevention Cheat Sheet**. OWASP Cheat Sheet Series, 2021. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html.

MIESSLER, Daniel. **SecLists: SecLists is the security tester's companion**. GitHub, 2021. Disponível em: https://github.com/danielmiessler/SecLists.

OWASP ZAP: Zed Attack Proxy. Versão 2.14.0. [S. l.]: OWASP Foundation, 2024. Disponível em: https://www.zaproxy.org/.