USO DE TESTE DE INVASÃO PARA AVALIAR A SEGURANÇA DO SISTEMA SHAVI

Otávio Hideki Gonçalves Kochi (PIBIC/CNPq/FA/UEM), Luciana Andréia Fondazzi Martimiano (Orientadora), Heloise Manica Paris Teixeira (Co-orientadora), e-mail: ra107635@uem.br, lafmartimiano@uem.br, hmpteixeira@uem.br.
Universidade Estadual de Maringá / Centro de tecnologia /Maringá, PR.

Ciências Exatas e da Terra. Ciência da Computação.

Palavras-chave: Segurança da informação, Teste de Invasão, Lousa Eletrônica.

Resumo:
O sistema Web denominado SHAVI (*Shared View*) foi desenvolvido para a unidade de emergência do Hospital Universitário de Maringá e tem como principais funcionalidades o gerenciamento de informações e a apresentação em lousa eletrônica de alertas sobre atividades realizadas pelos profissionais da saúde. Como o sistema gerencia dados críticos, é importante que este gerencie os dados com segurança. Para analisar a segurança do sistema foi utilizado o método de Teste de Invasão, que busca encontrar possíveis vulnerabilidades que um atacante mal intencionado poderia descobrir e explorar. Com a utilização de ferramentas apropriadas para auxiliar a realização de testes, foi possível identificar vulnerabilidades no SHAVI.

Introdução

Com a dimensão que a Internet tomou nos últimos anos e com todas as possibilidades de uso que ela nos proporciona, é comum pessoas questionarem se suas informações estarão seguras e se serão confidenciais. Caso alguns dados sejam obtidos por pessoas mal intencionadas, há uma grande possibilidade de que criminosos possam realizar golpes trazendo prejuízos pessoais e financeiros (ou não) para os clientes e para as empresas. Em sistemas desenvolvidos na área da saúde, onde dados críticos e confidenciais são gerenciados, cresce a relevância de estudos relacionados a segurança de informação.

O Sistema SHAVI, desenvolvido com base nas atividades da equipe de enfermagem da unidade de emergência do Hospital Universitário de Maringá (HUM), dentre suas principais funções, gerencia informações sobre atendimentos realizados em pacientes e apresenta em uma lousa eletrônica (monitores) avisos e alertas importantes aos profissionais de saúde. Neste contexto, os dados gerenciados devem ser confiáveis e confidenciais, fazendo-se necessária uma avaliação de segurança sobre o sistema.

O teste de invasão (ou *pentest*) são testes autorizados e utilizados por um profissional, denominado de *pentester*, que avalia a segurança de um

determinado sistema de software, simulando métodos que um possível atacante (*cracker*) poderia realizar (Weidman, 2014). Tais testes recaem sobre os pilares da segurança da informação, que são a confidencialidade, integridade e disponibilidade (Tanenbaum; Wetherall, 2011). Confidencialidade requer que as informações sejam acessadas apenas por entidades autorizadas. A integridade requer que as informações só sejam alteradas por entidades e tarefas autorizadas, não podendo ser alteradas maliciosamente. Já a disponibilidade requer que o sistema forneça as informações aos usuários autorizados sempre que forem requisitadas.

Materiais e métodos

É possível classificar o teste de invasão em três tipos diferentes: *Black Box, White box* e *Gray box.* No caso do *Black Box*, o *pentester* não possui informações prévias sobre o alvo que atacará, simulando assim um ambiente mais real. Já o *White Box* é o contrário, o profissional tem conhecimento total sobre o sistema que irá invadir, possui o código-fonte, as tecnologias utilizadas e assim por diante. Por fim, o *Gray Box* é uma mistura entre os dois outros tipos de teste, no qual o *pentester* possui um conhecimento a respeito do sistema, porém, limitado. No presente trabalho utilizou-se o teste *Gray Box*.

Independentemente do tipo escolhido em uma determinada situação, o teste é constituído por oito fases (Weidman, 2014): *Pre-Engagement, Reconnaissance, Enumeration, Scanning, Exploitation, Maintaining Access, Covering Tracks* e *Report.* Na etapa ***Pre-Engagement*** o *pentester* determinará o escopo da invasão, juntamente com os responsáveis pelo sistema, delimitando quais testes poderão ser realizados e em quais horários poderá ser efetuado. Já a etapa ***Reconnaissance*** é de reconhecimento, na qual o atacante coleta informações sobre o alvo para planejar seu ataque. A etapa ***Enumeration*** consiste em o profissional identificar possíveis vetores de ataque para realizar a exploração, como por exemplo, identificar redes compartilhadas, nomes de usuários e computadores ligados ao sistema. A fase ***Scanning*** é quando o *pentester* procura encontrar vulnerabilidades no sistema, e algumas ferramentas podem ser utilizadas nessa etapa para agilizar o processo. ***Exploitation*** é a utilização de códigos maliciosos para explorar as vulnerabilidades encontradas nas etapas anteriores a fim de ganhar alguma vantagem sobre o sistema, podendo obter até mesmo privilégio de administrador. ***Maintaining Access*** é a etapa em que o atacante procura meios para persistir no alvo já infectado até que realize tudo que deseja. Na etapa de ***Covering Tracks*** o profissional tenta remover todos os possíveis rastros que indicam que o sistema foi atacado. Por fim, na etapa ***Report*** o pentester faz um relatório mostrando as vulnerabilidades encontradas no sistema, quais riscos elas podem trazer e maneiras de corrigi-las.

Neste trabalho foram utilizadas as seguintes etapas: *Pre-Engagement, Reconnaissance, Enumeration, Scanning, Report.* Também foi utilizada a distribuição Kali Linux (https://www.kali.org), uma vez que é a distribuição

mais conhecida para a realização de teste de invasão e conta com várias ferramentas pré instaladas no sistema.

Dentre as ferramentas disponíveis para aplicações Web, as seguintes foram utilizadas: Burp Suite (https://portswigger.net/burp), SQLMap (http://sqlmap.org/), OWASP Zap (https://owasp.org/www-project-zap/) e Vega (https://subgraph.com/vega/). Estas ferramentas foram selecionadas por serem gratuitas e amplamente utilizadas para realização de teste de invasão em sistemas Web. A principal funcionalidade da ferramenta Burp Suite é a interceptação do *proxy*, podendo assim ser feita a análise das requisições que são enviadas para a aplicação, é possível também alterá-las e encaminhá-las para o destino. Já as ferramentas OWASP Zap e Vega têm como principal função o *scan* automatizado. A ferramenta SQLMap procura encontrar vulnerabilidades relacionadas a *Injection*, como por exemplo SQL *Injection*, podendo também recuperar nome das linhas da tabela de um banco de dados.

Para avaliar as vulnerabilidades que foram encontradas no sistema SHAVI, foi necessário realizar um estudo do projeto OWASP (*Open Web Application Security Project*) (https://www.owasp.org/). O OWASP é um projeto sem fins lucrativos que visa à melhoria da segurança de software. Dentre os inúmeros projetos, está o OWASP Top Ten (https://owasp.org/www-project-top-ten/), um documento para desenvolvedores que diz respeito sobre as vulnerabilidades mais críticas de segurança para aplicações web. O documento apresenta, conforme segue, as 10 vulnerabilidades Web mais críticas: *Injection; Broken Authentication; Sensitive Data Exposure; XML External Entities; Broken Access Control; Security Misconfiguration; Cross-Site Scripting; Insecure Deserialization; Using Components with Known Vulnerabilities; Insufficient Logging & Monitoring*.

Resultados e Discussão

Após a realização das etapas do teste de invasão no sistema SHAVI, foi encontrada uma vulnerabilidade no sistema que é bastante difundida e muito crítica segundo o projeto OWASP Top Ten.

Também foi possível identificar outras vulnerabilidades que não estão presente no projeto OWASP Top Ten, as quais foram: *Cookie Without Secure Flag*, *HTTP Trace Support Detected*, *X-Frame-Options Header Not Set*,, *Cookie Without Secure Flag, Absence of Anti-CSRF Tokens, Cross-Domain JavaScript Source File inclusion, Incomplete or No Cache-control and Pragma HTTP Header Set* e *X-Content-Type-Options Header Missing*.

A Tabela 1 apresenta um resumo das principais vulnerabilidades encontradas no SHAVI. Caso algum atacante consiga explorar tais falhas, é possível que o sistema possa sofrer algum tipo de prejuízo dentre os já descritos acima.

Tabela 1. Principais vulnerabilidades encontradas

Cross-site Scripting	Vulnerabilidade que permite a injeção de scripts maliciosos devido a não filtragem

	correta por parte do sistema, permitindo redirecionar o usuário para outros sites, sequestrar a sessão do usuário ou roubar seus cookies.
Cross-Domain JavaScript Source File inclusion	Ocorre quando um *script* de domínio externo é utilizado na aplicação. Com isso, não se tem controle sobre o código inserido, sendo assim, caso o *script* seja alterado a aplicação pode se tornar vulnerável
Cookie Without Secure Flag	O propósito do *Secure flag* é prevenir que o *cookie* seja visto em modo texto. Quando se utiliza do *Secure Flag*, os *cookies* só serão enviados por requisições HTTPS, que utilizam TLS e, com isso, não pode ser visto por alguém, pois utiliza canais seguros/criptografados.

Conclusões

Esta pesquisa buscou estudar segurança de aplicações Web, suas possíveis vulnerabilidades e aplicar os conhecimentos sobre teste de invasão em um estudo de caso no sistema SHAVI. Como resultado, identificou-se falhas de segurança, que poderão ser corrigidas antes que o sistema seja implantado no Hospital, prevenindo assim, ataques maliciosos no sistema.

O trabalho teve como objetivo principal identificar possíveis vulnerabilidades, mas não as explorou. Um trabalho futuro é aprofundar o estudo das falhas identificadas, a fim de minimizar riscos de ataques maliciosos.

Agradecimentos

Agradecemos à Universidade Estadual de Maringá pela Bolsa de Iniciação Científica e aos financiadores do projeto de pesquisa do Sistema SHAVI (Edital PPSUS Edição 2015 - Fundação Araucária-PR/SESA-PR/CNPq/MS-Decit).

Referências

OWASP. **OWASP Top Ten.** Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 7 ago. 2020.

TANENBAUM, Andrew S.; WETHERALL, David. Redes de Computadores. 5. ed. Pearson Universidades, 2011.

WEIDMAN, Georgia; Testes de Invasão: Uma introdução prática ao hacking. Editora Novatec. 2014.