Estudo comparativo entre técnicas de detecção e prevenção de ataques de injeção SQL

Gustavo P. Lages[1], **Rafael T. Pereira**[2]

[1]Colégio Técnico Industrial de Santa Maria (CTISM)

[2]Curso de tecnologia em redes de computadores
Universidade Federal de Santa Maria (UFSM) – 97105-900 – Santa Maria, RS – Brazil

{gustavolages, rafatp}@redes.ufsm.br

Abstract. *Currently, data security becomes an aspect that must be taken into account during the development of web applications. One of the most common attacks, which requires great attention during the process of creating systems, is the SQL Injection attack, which allows intruders to insert malicious code to the database in order to access, modify or even delete confidential information. With that in mind, the present work has the objective of presenting a comparative study between intrusion prevention and detection systems and Web Application Firewalls based on automated SWL injection tests, to define effective options to protect different applications of this type of attack.*

Resumo. *Atualmente, a segurança de dados se torna um aspecto que deve ser levado em consideração durante o desenvolvimento de aplicações Web. Um dos ataques mais comuns, e que requer grande consideração no processo de criação de sistemas, é o ataque de injeção SQL, que permite a adversários inserir códigos maliciosos ao banco de dados com o intuito de acessar, modificar ou até mesmo apagar informações confidenciais. Pensando nisso, o presente trabalho tem como objetivo apresentar um estudo comparativo entre sistemas de prevenção e detecção de intrusões e Web Application Firewalls baseado em testes automatizados de injeção de SQL, para definir opções efetivas para proteger diferentes aplicações desse tipo de ataque.*

1. Introdução

Há anos que a Internet deixou de ser algo utilizado somente para entretenimento, e passou a ser um meio de trabalho para de milhares de pessoas, e um dos métodos mais comuns de atacar sistemas *web* é conhecido como ataque de injeção SQL [Kaur and Kaur 2016]. Este tipo de ataque dá ao atacante a possibilidade de injetar códigos maliciosos à base de dados, com o objetivo de modificar e recuperar dados de outros usuários e até mesmo conseguir credenciais para obter acesso a áreas restritas da aplicação.

Para prevenir e mitigar esse tipo de ataque existem diversas ferramentas como *Web Application Firewalls* e sistemas de prevenção e detecção de intrusões. O presente trabalho tem como objetivo apresentar uma comparação entre algumas ferramentas utilizadas para mitigar ataques de injeção SQL e demonstrar qual opção é a mais eficaz em cenários onde este tipo de ataque pode ser executado.

2. Ataques de injeção SQL

[Marashdeh et al. 2021] afirma que o termo injeção SQL compreende um tipo de ataque que permite que usuários obtenham acesso não autorizado a bases de dados através da injeção de código em conjunto de uma *query* SQL. Esse tipo de ataque pode ser executado facilmente em qualquer formulário de *login*, cadastro, ou campo de busca.

Podemos usar, como exemplo, o campo de busca de uma loja online. Para efetuar a busca dos produtos, no banco de dados é utilizado a cláusula `SELECT` como é mostrado no código abaixo.

```
$categoria = $POST['categoria'];

$query = "SELECT *
          FROM produtos
          WHERE produto_categoria = '".$categoria."';
```

O código acima representa a obtenção de dados de um formulário utilizando o SQL em conjunto da linguagem PHP para retornar todos os produtos de uma determinada categoria. Essa informação é inserida pelo usuário na hora da busca, e é armazenada na variável `$categoria`. O método utilizado para montar a *query* é bem simples e não oferece nenhum tipo de tratamento dos dados inseridos pelo usuário, portando, nada impede que no lugar da pesquisa seja inserido um pedaço de código SQL malicioso para que a consulta retorne mais resultados.

Um usuário mal intencionado poderia digitar a *string* `' OR 1=1`, e isso permitiria retornar todos os dados de todos os produtos cadastrados no banco de dados. Isso é possível em função do texto digitado ser concatenado à `query`, mostrada na trecho de código abaixo, fazendo com que a condição adicionada pelo usuário seja sempre verdadeira.

```
SELECT * FROM produtos WHERE produto_categoria = '' OR 1=1
```

3. Métodos de mitigação

Conforme aplicações *web* se expandem e agregam cada vez mais dados de diversos usuários, é importante que seja utilizada uma gama de recursos para proteger a integridade e disponibilidade dos demais sistemas.

3.1. *Web Application Firewall*

Web Application Firewall (WAF) é o mecanismo mais comum a nível de aplicação utilizado para mitigar e prevenir tipos específicos de ataques à sistemas *web*. WAFs possuem a função de examinar o tráfego HTTP e filtrar as requisições, aceitando ou não a sua entrada na aplicação protegida. Os filtros do *firewall* são definidos por um conjunto de regras que pode ser algo como um conjunto de expressões regulares, utilizadas como parâmetro para detectar tentativas de ataques [Appelt et al. 2017]. Para o desenvolvimento do trabalho foram escolhidos WAFs populares e grátis para uso com um bom suporte de suas comunidades.

3.2. Sistemas de detecção e prevenção de intrusões

[Scarfone et al. 2007] define detecção de intrusão como o processo de monitoramento e análise de eventos que ocorrem em uma rede ou sistema para detectar possíveis incidentes, como violações de segurança, implantação de *malwares*, acesso não autorizado de atacantes a serviços e sistemas, etc. Um IPS serve para um propósito mais geral, como monitorar protocolos como: DNS, SMTP e FTP. Para o desenvolvimento do trabalho foram escolhidos IPSs e IDSs populares e grátis para uso com um bom suporte de suas comunidades. WAFs e IPSs são tecnologias com propósitos diferentes que juntas fornecem um nível maior de segurança.

4. Metodologia

Em um primeiro momento será instanciada uma máquina virtual Ubuntu em sua versão 20.0, onde será inicializado um servidor *Apache* contendo o DVWA (*Damn Vulnerable Web Application*), que consiste de uma aplicação *web* desenvolvida em PHP que possui vulnerabilidades propositais para permitir que profissionais de segurança testem suas habilidades e ferramentas em um ambiente controlado [Makino and Klyuev 2015].

Assim que o ambiente com o DVWA estiver configurado, será feita a instalação dos WAF's e IPS, com suas respectivas regras. Para cada teste que será feito, um WAF e um IPS será escolhido. A lista de WAFs pode ser visualizada na Tabela 1.

Tabela 1. WAFs e artifícios utilizados para detecção dos ataques

WAFs	Artificio utilizado para detecção dos ataques
Modsecurity	*Conjunto de regras Core Rules Set*
NAXSI	Filtros básicos do WAF
Shadow Daemon	Blacklists e whitelists por meio da interface gráfica

A lista de IPSs pode ser visualizada na Tabela 2.

Tabela 2. IPSs e artifícios utilizados para detecção dos ataques

IPSs	Artificio utilizado para detecção dos ataques
Snort	*Community SQL Injection Rules*
OSSEC	*Web Rules*
Suricata	Criação de regras com whitelists e blacklists

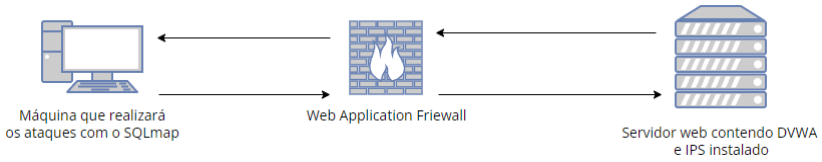
Em seguida em uma máquina atacante será aplicado o *sqlmap*, que consiste de uma ferramenta utilizada para automatizar o processo de detecção e *exploit* de pontos vulneráveis a injeção SQL presentes na aplicação[Ojagbule et al. 2018]. Serão efetuados diferentes tipos de ataques com o auxilio do *sqlmap*.

Para que sejam coletados os resultados do presente estudo, deverá ser criado um ambiente isolado onde os ataques possam ser efetuados de maneira controlada. O desenvolvimento dos testes se resume à um sistema *web* vulnerável (DVWA) reforçado com artifícios para mitigar e prevenir intrusões (WAFs e IPSs), que em um segundo momento

sofrerá ataques automatizados com o *sqlmap*, e por fim serão coletados e comparados os resultados, documentando qual combinação de WAF e IPS foi capaz de prevenir os ataques.

A Figura 1 apresenta como será a comunicação entre a máquina atacante e o servidor contendo o DVWA.

Figura 1. Diagrama demonstrando a disposição do ambiente de testes



Fonte: do autor.

5. Conclusão

É de suma importância que medidas de segurança sejam implementadas à aplicações *web*, tanto por questões de manter a disponibilidade dos serviços, quanto para proteger os dados de seus usuários, pois os sistemas encontrados na *Internet* são suscetíveis a ataques de diversas formas, sendo uma das mais perigosas e comuns a injeção de código SQL malicioso em bases de dados desprotegida, possibilitando o retorno de dados confidenciais e até mesmo credenciais para acesso a áreas restritas, e para prevenir que isso aconteça, várias tecnologias podem ser implementadas.

O presente trabalho visa suprir uma carência na literatura de estudos comparativos entre algumas tecnologias *open-source* utilizadas para prevenir e mitigar ataques de injeção SQL, através de testes automatizados utilizando a ferramenta *sqlmap* para ao final apresentar os resultados, mostrando qual a opção mais eficiente na prevenção dos ataques.

Referências

Appelt, D., Panichella, A., and Briand, L. (2017). Automatically repairing web application firewalls based on successful sql injection attacks. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 339–350. IEEE.

Kaur, D. and Kaur, P. (2016). Empirical analysis of web attacks. *Procedia Computer Science*, 78:298–306.

Makino, Y. and Klyuev, V. (2015). Evaluation of web vulnerability scanners. In *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 399–402.

Marashdeh, Z., Suwais, K., and Alia, M. (2021). A survey on sql injection attack: Detection and challenges. In *2021 International Conference on Information Technology (ICIT)*, pages 957–962.

Ojagbule, O., Wimmer, H., and Haddad, R. J. (2018). Vulnerability analysis of content management systems to sql injection using sqlmap. In *SoutheastCon 2018*, pages 1–7.

Scarfone, K., Mell, P., et al. (2007). Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94.