Uso da ferramenta sqlMap para detecção de vulnerabilidades de SQL Injection

Felipe Santin[1], José Antônio Oliveira de Figueiredo[1], Vanessa Lago Machado[1]

[1]Curso Tecnologia em Sistemas para Internet – Instituto Federal Sul-rio-grandense
Campus Passo Fundo – RS – Brasil

felipesantin123@hotmail.com,
{jose.figueiredo,vanessa.machado}@passofundo.ifsul.edu.br

Abstract. *This work describe, no exhaustively, the SQL Injection problem. A brief study on this vulnerability and its variants . At sequence its explained sqlMap tool, used for automate the detection and explore of this kind of vulnerability. Main risks of SQL Injection are tested and demonstrated with sqlMap. The main contribution is a systematized demonstration of sqlMap use as tool to detect and prevent SQL Injection vulnerability.*

Resumo. *Este artigo apresenta de forma não exaustiva o problema conhecido por SQL Injection, fazendo um breve estudo sobre a vulnerabilidade e apresentando as principais variantes da falha. Na sequencia é apresentada a ferramenta sqlMap, utilizada para automatizar a detecção e exploração deste tipo de vulnerabilidade. São demonstrados os principais riscos e consequências da da falha. A principal contribuição está na demonstração sistematizada do uso da ferramenta sqlMap para a descoberta da vulnerabilidade de forma preventiva.*

1. Introdução

Segundo Clarke (2009) o *SQL Injection* é uma técnica onde um atacante explora uma vulnerabilidade que permite alterar comandos SQL em uma aplicação. Conhecido como uma das vulnerabilidades de maior impacto em um negócio. Isto ocorre porque a falha pode expor todas as informações armazenadas no banco da dados de uma aplicação, incluindo dados sensíveis de usuários como senhas, telefones e endereços [Clarke, 2009, tradução nossa].

Segundo Muscat (2016) desde a primeira discussão pública sobre *SQL Injection*, que ocorreu por volta de 1988, a vulnerabilidade tem se tornado um dos mais antigos e mais predominantes causadores de *bugs* em softwares que ainda são explorados atualmente. Além disto, o *SQL Injection* figura entre as vulnerabilidades mais perigosas, (inclusive na lista de Top 10[1]) conforme o Projeto *Open Web Application Security* (OWASP, 2017).

Para execução de testes ou exploração do *SQL Injection* existem diversas ferramentas que visam a automatização desta tarefa, entre elas Clarke (2009) cita o sqlMap[2] que é uma ferramenta baseada em linha de comando, *Open Source,* desenvolvida sobre a licença GNU GPLv2 por Bernardo Damele A.G. e Miroslav Stampar.

As referências para este trabalho foram buscadas no Portal de Periódicos da Capes/Mec, com a chave de busca "*SQL Injection Detection*". Foram buscados apenas

1 https://www.owasp.org/index.php/Top_10_2017-Top_10
2 http://sqlmap.org/

artigos revisados por pares entre o período de 2015 e 2017. A mesma busca foi feita no Google Acadêmico no período de 2015 a 2017, classificado por relevância e com o termo "*SQL Injection attacks detection and prevention*".

O restante do artigo está organizado como segue: Na seção dois é feita um estudo bibliográfico sobre as características do *SQL Injeciton* e da ferramenta sqlMap. Na seção três é descrito a execução de um estudo de caso aplicado para demonstrar o funcionamento da falha e da ferramenta. Por fim, na seção quatro são apresentadas algumas considerações sobre o trabalho.

2. Vulnerabilidade Sql Injection

Segundo Atoum e Qaralleh (2014) não existem soluções que garantem o controle de todas as vulnerabilidades. Estas vulnerabilidades ocorrem em função de falhas de hardware e de software. Caso esses elementos não sejam atualizados constantemente, acabarão por representar uma maior probabilidade de conter vulnerabilidade que possa ser exploradas em ataques.

Para Atoum e Qaralleh (2014) o *SQL Injection* é uma técnica de ataque, que pode ser bastante efetiva para acessar ou manipular dados em um sistema, de forma não autorizada. Isto é um problema bastante preocupante para desenvolvedores de aplicações web, principalmente as publicadas na Internet.

Charania (2016) explica alguns dos principais tipos de *SQL Injection* que existem:

- **Tautologia**: Neste tipo são usadas consultas condicionais, sempre procurando gerar condições verdadeiras que sejam aceitas como válidas pelo sistema;
- **Consultas com lógica incorreta/ilegal:** Utiliza-se de mensagens de erros que são retornados de consultas;
- **Consultas com "Union":** Consulta-se utilizando o operador "union" assim obtendo formulários de informações do banco de dados;
- **Consultas "Piggy-backed":** São as consultas utilizando ";" onde várias consultas podem ser executadas ao mesmo tempo, sendo que a primeira seria a verdadeira e as demais seriam ilegítimas, mas que retornam informações valiosas;
- **Procedimentos:** Procedimentos são sub consultas pré compiladas no banco, onde existem diversas formas de ataque;
- **Injeção cega:** Ocorre quando os desenvolvedores escondem as mensagens de erro, e assim o ataque será difícil mas pode ser realizado através de consultas com verdadeiro e falso;
- **Ataques por tempo:** Utilizando tempo de demora da execução de uma consulta o atacante pode conseguir informações, e através de uma consulta com vários "if-else" o atacante consegue, pela demora do processo, observar se o resultado é verdadeiro;
- **Codificações Alternativas:** Utilizando codificações com ASCII e Unicode, atacantes podem passar por filtros de caracteres especiais.

Segundo Muscat (2016) um atacante pode pode injetar uma consulta sql em um sistema, de forma que provoque a exclusão de tabelas, levando a negação do serviço oferecido por este sistema. Outro aspecto importante é a possibilidade de o atacante obter

controle total sobre o servidor que contém o banco de dados. O autor comenta ainda que com o surgimento do *SQL Injection* os administradores de servidores Web notaram que mostrar mensagens detalhadas de erros para o público em geral não era uma boa opção, assim começaram a ocultar estas informações. Mas esta ação foi apenas uma solução paliativa, pois as consultas sql continuam sendo executadas mesmo que o atacante não tenha uma visão exata dos resultados obtidos, ou seja, o atacante ainda pode causar danos ao sistema.

Segundo Atoum e Qaralleh (2014) ações de monitoramento, verificação de logs, validações de acesso, detecções de intrusão e entre outras são muito importantes em arquitetura que busca aumentar a segurança de um sistema. Além disso, se o sistema não dispor de um técnica eficiente para validação das entradas poderá ter sérios problemas de segurança, uma vez que os parâmetros de entrada são os primeiros passos para uma atacante conseguir injetar códigos maliciosos.

Diante disto, verificar sistemas web em busca de falhas de *SQL Injection* antes (ou mesmo depois) da publicação, pode ser uma estratégia que contribua para o bom funcionamento do sistema e consequente aumento da disponibilidade do serviço online.

2.1. Automatização de testes com a ferramenta sqlMap

Segundo Clarke (2009) para execução de testes de *SQL Injection*, dependendo da situação, são necessários dezenas, centenas ou até milhares de requisições para conseguir as informações que você pode precisar. A realização deste serviço manualmente seria algo muito trabalhoso. Para resolver este problema adota-se ferramentas que automatizam este processo de teste.

Alguns exemplos de ferramentas citadas por Clarke (2009) são sqlMap, Bobcat, BSQL, Havij, Sqlsus, Pangolin. O sqlMap foi utilizado neste trabalho devido a sua popularidade e disponibilidade em diversas distribuições utilizadas para testes de vulnerabilidade.

O sqlMap é uma ferramenta *Open Source*, utilizada para automatizar a execução de testes para detecção e exploração de *SQL Injection* em aplicações Web. Usando sqlMap o testador pode executar comandos no sistema operacional, analisar detalhes de gerenciadores de bancos, consultar ou apagar dados do banco de dados e até acessar arquivos do servidor [Charania 2016, tradução nossa].

Segundo Clarke (2009) o sqlMap é uma das ferramentas mais utilizadas no momento da escrita de seu artigo, pois tem uma grande gama de recursos, suporta diversos banco de dados como: Microsoft Sql Server, Microsoft Access, Oracle, MySql, Postgresql, SqlLite, Firebird, Sybase e SAP MaxDB. Além disso, tem suporte a seis técnicas de *SQL Injection,* que são: *boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band*. Por fim, a ferramenta também é capaz de reconhecer o formato do *hash* de senhas [Charania 2016 Tradução nossa].

3. Escaneando e explorando vulnerabilidades

Para demonstrar o funcionamento da ferramenta sqlMap, demonstrar a identificação e a exploração de falhas, utilizou-se um site de testes hospedado no endereço http://testphp.vulnweb.com/. O site é disponibilizado pela empresa de segurança Acunetix[3], que fornece soluções de segurança, inclusive outras ferramentas para

3 https://www.acunetix.com/

escaneamento de vulnerabilidades. Este site foi desenvolvido especificamente para demonstração de falhas de *SQL Injection*.

O teste inicia-se verificando se há possibilidade de falha em alguma url com passagem de parâmetros. Para este teste, executa-se o comando :

```
#sqlmap -u testphp.vulnweb.com/artists.php?artist=1
```

Após a execução inicial dos testes, o sqlMap informa que a o parâmetro 'artist' é passível de injeção de sql e que possivelmente o banco alvo seja da família MySQL. Na próxima fase, a ferramenta executa o procedimento de escaneamento do alvo, realizando várias requisições e testando as possíveis formas de conseguir acesso ao banco de dados da aplicação.

Ao finalizar este procedimento o programa apresenta um log de mensagens, conforme a Figura Figura, indicando que foram necessárias 79 requisições para descobrir todas as formas de acesso possíveis através da ferramenta.

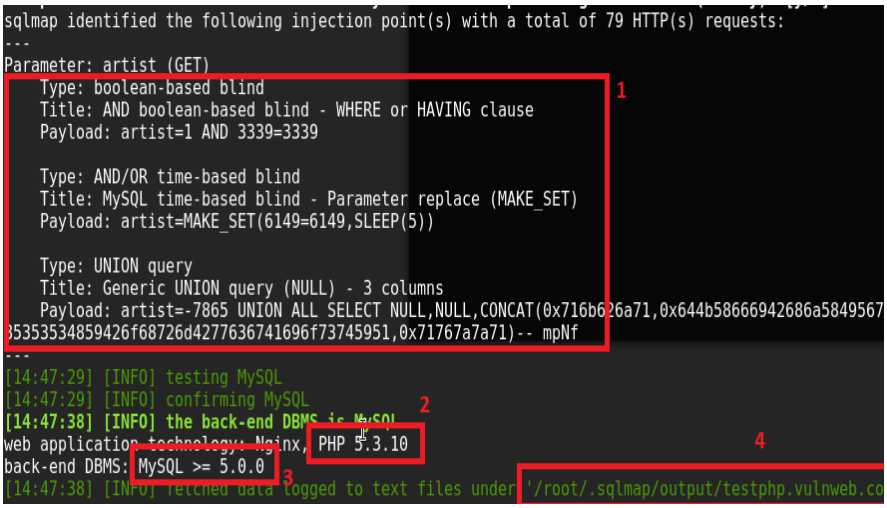

Figura 7: Resultado dos testes de injeção sql sobre o parâmetro 'artist' na url informada.

A Figura Figura está dividida em 4 quadros (marcados em vermelho) com informações, sendo que quadro 1 lista as técnicas de *SQL Injection* identificadas pela ferramenta, para qual o site alvo é vulnerável; o quadro 2 informa a versão e a linguagem de programação utilizada para o desenvolvimento do site, no caso PHP; no quadro 3 é exposto o banco utilizado e a versão deste. Por fim, no quadro 4 está mostrando a localização do log gerado com os resultados dos testes. A pasta do testes recebe como nome o domínio do site testado.

Na pasta de logs são armazenados três arquivos, sendo o primeiro chamado log, que registra todos os comandos executados no site alvo; o segundo arquivo é um banco de dados no formato sqlite, que armazena os dados de sessão dos acessos realizados, facilitando na hora de executar novos comandos e um terceiro arquivo, chamado target.txt, armazenando a url do site.

A partir deste ponto, a ferramenta já obteve acesso ao banco de dados, bastando explorar a ferramenta para obter mais informações. Na seção a seguir, são apresentadas as consultas para identificação do banco de dados (*schemas*) e tabelas disponíveis no

banco.

3.1 Listar databases e tabelas

O sqlMap possui diversas *flags* para execução de tarefas (ou consultas) específicas na url alvo. As *flags* para listagem de *schemas* e tabelas são, respectivamente --schema e --tables. Para listar as tabelas (e também databases) do banco de dados alvo, o comando utilizado foi:

```
#sqlmap -u testphp.vulnweb.com/artists.php?artist=1 --tables
```

Isto gera uma listagem com os banco de dados configurados dentro do SGBD. É mostrado na Figura Figura apenas o recorte das tabelas que estão no banco de dados acuart.

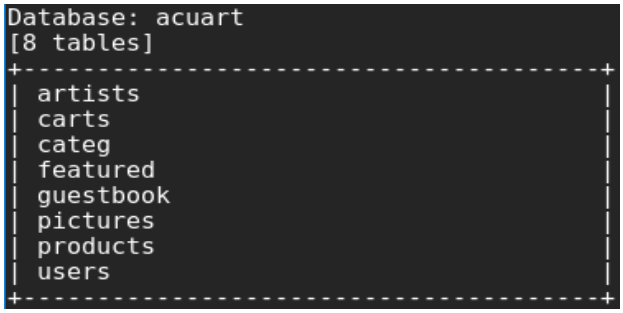

Figura 8: Listagem do schema 'acuart' expondo 8 tabelas do sistema que está sendo explorado.

A consulta expõe tanto o database *information_schema,* de controle do próprio MySQL, quanto o database acuart (alvo). O banco "acuart" contém 8 tabelas, conforme pode ser visto na Figura Figura. Naturalmente, o próximo ataque busca consultar os dados da tabela *users*, que muito provavelmente conterá os dados de login e senha de acesso.

Para esta verificação, utiliza-se um comando para expor as colunas da tabela users. O objetivo do comando é conhecer o nome das colunas e poder chegar a informação de login e senha para acesso completo ao sistema. O sqlMap acessa esta informação com o comando:

```
#sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -T users --columns
```

A *flag* -T é utilizada para especificar a tabela a ser consultada e *--columns* para listar as colunas. O resultado da consluta, mostrado na Figura Figura, expõe o nome das colunas da tabela.

```
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+
```

Figura 9: Listagem dos campos da tabela **users**.

Por fim, a próxima etapa é a execução do *dump* dos dados armazenados na tabela users. Para isto executa-se o comando:

```
#sqlmap -u testphp.vulnweb.com/artists.php?artist=1 --dump -T users
```

O resultado do comando, mostrado na Figura Figura, expõe os dados contidos na tabela, que mostram a existência de um único usuário cadastrado no sistema. A Figura está dividida em 4 partes principais, discutidas a seguir.

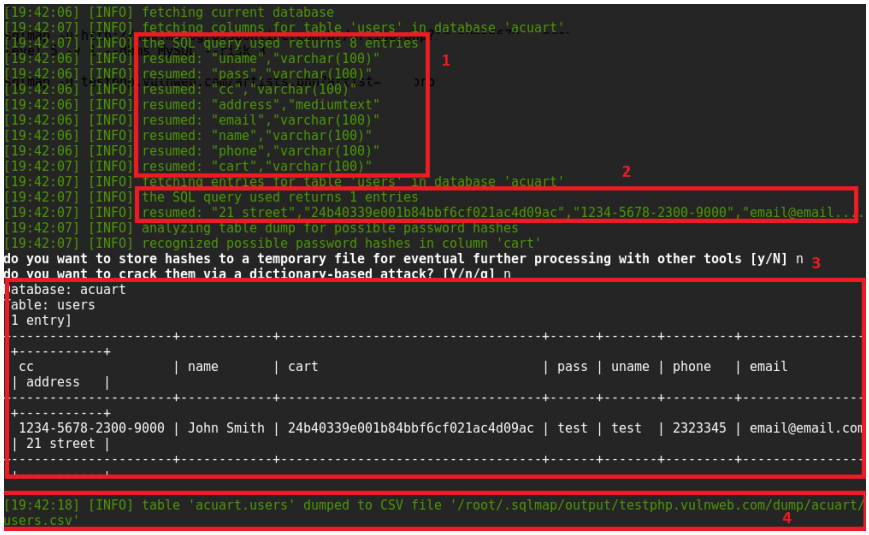

Figura 10: Dump expondo os dados armazenados na tabela **users**.

No quadro 1 são apresentados novamente os nomes de coluna da tabela explorada; o quadro 2 mostra um resumo dos dados inseridos na tabela; o quadro 3 mostra, de forma ordenada, os dados constantes na tabela. Por fim nono quadro 4 é informado a geração de um novo arquivo, de nome 'users.csv', na pasta com os logs do teste, onde ficam registrados os dados desta consulta.

4. Considerações finais

Vulnerabilidades de *SQL Injection* estão entre as falhas mais comuns e que de maior risco para sistemas que estão online. Falhas deste tipo podem expor dados sensíveis do sistema

afetado pela vulnerabilidade, bem como causar prejuízos ao proprietário, seja por perda de dados ou por negação de serviço. A verificação, por meio de testes de penetração (pentest), antes e depois da publicação do sistema são uma forma de auxiliar na prevenção de problemas deste tipo.

O sqlMap é uma das ferramentas mais conhecidas e mais completas para automação de pentest para vulnerabilidades de *SQL Injection*. A ferramenta é gratuita e *Open Source*, possui ótima documentação online e é muito fácil de para se utilizar. Além disto, diversas distribuições destinadas ao pentest trazem a ferramenta já instalada e pronta para uso.

Utilizar uma ferramenta para varredura de vulnerabilidades, antes mesmo da publicação do site ou sistema desenvolvido, contribui para que o desenvolvedor e equipe de TI tenham mais segurança quanto ao funcionamento e disponibilidade do serviço. A varredura por vulnerabilidades de *SQL Injection* é apenas um dos aspectos relacionados proteção de sistemas. A execução não exclui a necessidade de adoção de outras medidas (e políticas) de segurança para manutenção do site.

Os experimentos executados neste artigo demonstram as formas mais comuns para explorar a vulnerabilidade de *SQL Injection* por meio do sqlMap. Em trabalhos futuros pretende-se demonstrar a execução de testes mais avançados com a ferramenta, bem como exploração de falhas em outros tipos de banco de dados, além do conhecido MySQL e também com outras linguagens de programação, além do PHP.

Referências

Atoum, Jalal Omer; Qaralleh, Amer Jibril. A hybrid technique for SQL injection attacks detection and prevention. International Journal of Database Management Systems, v. 6, n. 1, p. 21, 2014.

Clarke, Justin. SQL injection attacks and defense second edition. Elsevier, 2009.

Charania, Swayam; Vyas, Vidhi. SQL Injection Attack: Detection and Prevention. 2016

Muscat, Ian. Web vulnerabilities: identifying patterns and remedies. Network Security, v. 2016, n. 2, p. 5-10, 2016.

OWASP FOUNDATION (Org.). **OWASP Top Ten 2017 Project**. 2017. Disponível em: <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project>. Acesso em: 13 jul. 2017.