Departamento de Segurança da Informação e Comunicações – **DSIC** dsic.planalto.gov.br/
Centro de Tratamento de Incidentes de Redes do Governo – **CTIR Gov** www.ctir.gov.br/

Alerta nº 03/2018 – Ataques de *SQL Injection*

1. Descrição do Problema

Data de Publicação: 27/05/2018
Data de Atualização: 27/05/2018

Em virtude de observarmos um crescimento de buscas por sítios do governo com supostas vulnerabilidades a *SQL injection*, o CTIR Gov recomenda a divulgação, em todos os órgãos da APF e entidades vinculadas, do Alerta nº 03/2018 – Ataques de *SQL Injection*.
Ataques de injeção ocorrem quando comandos que deveriam ser processados em tempo de execução, permitem a concatenação de valores fornecidos por usuários, alterando a semântica original do que deveria ser executado pelo interpretador, por meio de injeção de palavras-chave e de caracteres com sentido especial na linguagem empregada, com o objetivo de que sejam executados na camada de dados.
O *SQL Injection* ocorre quando o usuário mal intencionado consegue inserir uma série de instruções SQL dentro de uma consulta através da manipulação das entradas de uma aplicação.
Com um pouco mais de elaboração, é possível aproveitar-se dos mecanismos de interação com o sistema operacional, existentes em bancos de dados, para leitura/escrita de arquivos e execução de comandos arbitrários.
Como a maioria dos fabricantes de software utiliza o padrão SQL-92 ANSI na escrita do código SQL, os problemas e as falhas de segurança aqui apresentadas se aplicam a todo ambiente que faz uso desse padrão para troca de informações.

2. Métodos de Ataques

Devido à falta de validação de entradas de dados das aplicações e, dependendo da estrutura do comando *SQL (Statement*) a ser executado, o atacante pode alterar a semântica do comando *SQL* original, adicionando os comandos maliciosos a serem executados para exploração de vulnerabilidades que podem potencializar o dano do ataque.
Existe uma grande quantidade de técnicas que podem ser utilizadas e variam de acordo com os SGBDs.

A seguir, algumas vulnerabilidades mais comumente exploradas para os ataques de *SQL Injection*:

- Aplicações que não tratam erro adequadamente: Se informações de erros forem exibidas ao usuário, é possível descobrir estrutura de tabelas e de outros objetos.
- Configuração insegura do servidor de Banco de Dados: Contas de acesso com senha padrão, objetos desnecessários instalados, privilégios excessivos sobre objetos, facilitam o vazamento de informação e a escalada de privilégios.

É importante ressaltar que, embora sejam similares, as sintaxes de *SQL* empregadas pelos diversos SGBDs existentes, apresentam vários aspectos específicos que são conhecidos pelos atacantes.
Os atacantes utilizam as tabelas padrões dos SGBDs que contém dados dos esquemas e usam a tautologia na cláusula WHERE para obter informações relevantes.

Ex: *select * from user_data where USERID = ' 'or 1=1- -'*

Nesse caso, a cláusula *where* é sempre verdadeira e a consulta retorna todos os dados da tabela user_data.
Existem diversas outras técnicas que são utilizadas/exploradas por atacantes e que existe vasta literatura sobre as mesmas, como, locais de injeção e regras que devem ser respeitadas, testes básicos utilizando ferramentas automatizadas, extração de dados via cláusula *UNION*, técnicas de identificação do servidor de banco de dados, técnicas de identificação das colunas da consulta, técnicas de escalada de privilégios, descoberta e extração de tabelas, manipulação de arquivos, execução de comandos no sistema operacional, varredura de redes, injeção de *SQL* às cegas, injeção de *SQL* de segunda ordem e evasão de filtros..

3. Sugestões para Mitigação do Problema

Para minimizar que aplicações *WEB* estejam vulneráveis a injeção de *SQL*, sugerimos as seguintes medidas de controle, que devem ser adotadas nos processos de desenvolvimento e de implantação de Sistemas de Informações (Howard et al., 2005; Clarke, 2009; Stuttard e Pinto, 2007):

• Verificar se toda informação fornecida pelo usuário está de acordo com valores reconhecidamente válidos para o campo ou parâmetro e complementarmente, restrinja o tamanho do campo ao máximo permitido.
• Se aspas forem permitidas em campos textuais, sempre duplicar antes de utilizá-las em comandos SQL.
• Não permitir concatenação de comando com valores fornecidos por usuários.
• Utilizar somente "*Prepared Statements*", comandos que são pré-compilados e não permitem que a semântica seja alterada.
• Realizar o acesso à camada de dados por meio de procedimentos definidos no banco de dados, encapsulando assim, a estrutura das tabelas que compõem a aplicação.
• Capturar todos os erros de execução e fornecer apenas mensagens de erro tratadas para os usuários.
• Utilizar na aplicação uma conta de acesso com os mínimos privilégios necessários para utilização do sistema. Nunca usar contas com privilégios *Data Definition Language* (DDL), muito menos contas administrativas.
• Instalar filtro de pacotes no servidor de banco de dados, configurar o mesmo para permitir apenas os tráfegos de entrada e saída válidos.

5. Referências

- https://www.owasp.org/index.php/SQL_Injection
- https://www.devmedia.com.br/evitando-sql-injection-em-aplicacoes-php/27804
- https://secure.php.net/manual/pt_BR/security.database.sql-injection.php
- https://www.devmedia.com.br/prevenindo-sql-injection-asp-net/17545
- https://forum.imasters.com.br/topic/390834-sql-injection/
- https://www.devmedia.com.br/sql-injection/6102 21/
- https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL 6/