# SQL Injection Attack Report

SQL injections are considered one of the most serious vulnerabilities. The adversaries inject (legitimate) SQL commands to the input data of a given application for modifying the initial SQL command, and gain access or modify private information. The adversaries exploit the fact that database makes no differentiation between end-users' actual data and SQL commands. We have observed that about 36% of the traffic in our campus network, analyzed through deep content inspection, contains SQL injection keywords.

SQL injections exploit security vulnerabilities in an application's software. In data-driven applications, malicious SQL statements are inserted into an input field for execution (e.g. to dump the database contents to the attacker). SQL injection can be used to attack SQL databases in a variety of ways.

SQL Injection attacks are unfortunately very common, and this is due to two factors: the prevalence of SQL Injection vulnerabilities and the attractiveness of the target (databases containing the interesting/critical data for the application).

The cause of SQL injection is accepting data from untrusted sources (Internet users) by application software that fails to validate and sanitize data. Subsequently, the app uses the data to dynamically construct a SQL query to the database backing the app. SQL injection attacks cause the breach of confidentiality in the data stored in the compromised databases, which results in financial costs for recovery, downtime, regulatory penalties, and negative publicity.

Any data that is passed from the user to the vulnerable web application and then processed by the supporting database represents a potential attack vector for SQL injection. The attack vectors are supplied through HTTP GET, HTTP Post, and HTTP cookie data.

Defensive programming such as employing sanitization techniques on end-users' inputs is the obvious approach to address SQL injection attacks. SQL injection attacks can be prevented by accepting certain characters in input fields, and limiting the length of those fields. So for the username and password, for example, only accept letters for the alphabet and numbers, and limit the field to 15 characters if possible.

There are multiple rules for detecting the attack depending upon the organization's level of sensitivity. If an organization wishes to detect each and every possible SQL Injection attack, then they simply need to watch out for any occurrence of SQL meta-characters such as the single-quote, semi-colon or double-dash.

Penetration testing tool automates the process of detecting SQL injection flaws. It also includes a number of features for further exploitation of vulnerable systems, including database fingerprinting, collecting data from compromised databases, accessing the underlying file system of the server, and executing commands on the operating system via out-of-band connections.

An attacker can inject SQL into input taken from a form, as well as through the fields of a HTTP cookie. The input validation logic should consider each and every type of input that originates from the user – be it form fields or cookie information – as suspect. Also if you discover too many alerts coming in from a signature that looks out for a single-quote or a semi-colon, it just might be that one or more of these characters are valid inputs in cookies created by the Web application. Therefore, each of these signatures needs to be evaluated for the particular Web application

A web application firewall (WAF) is an appliance, server plugin, or filter that applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as cross-site scripting (XSS) and SQL

injection. By customizing the rules to your application, many attacks can be identified and blocked. The effort to perform this customization can be significant and needs to be maintained as the application is modified.