# GreenSQL Security

**Jaydeep K. Dabhi[1] Ass. Prof. Tarun K. Sureja[2]**
[1]M.E. CE (I.T. Systems & Network Security) [2]M. Tech (CSE)
[1]AVPTI, Rajkot, Gujarat Technological University [2]NIT, Rourkela

*Abstract*— In today's modern world, security is a necessary fact of life. GreenSQL Security helps small to large organizations protect their sensitive information against internal and external threats. The rule-based engine offers database firewall, intrusion detection and prevention (IDS/IPS). GreenSQL Security Engine applies exception detection to prevent hacker attacks, end-user intrusion and unauthorized access by privileged insiders. The system provides a web based intuitive and flexible policy framework that enables users to create and edit their security rules quickly and easily. GreenSQL interfaces between your database and any source requiring a connection to it. This approach shields your database application and database operating system from direct, remote access.

GreenSQL Database Security

1) Stops SQL Injection attacks on your web application
2) Blocks unauthorized database access and alerts you in real time about unwanted access
3) Separates your application database access privileges from administrator access
4) Gives you a complete event log for investigating database traffic and access
5) Ensures you achieve successful implementation with 24/7 support

*Keywords*: SQL injection, web architecture, SQL tautology, IDS (intrusion detection), IPS (intrusion prevention )

## I. SQL INJECTION

SQL injection is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.

The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.

It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another.

SQL injection is a basic attack used to either gain unauthorized access to database or to retrieve information directly from the database.

SQL injection is a technique used to take advantage of no validated input vulnerabilities to pass SQL commands through a web application for execution by a backend database. SQL commands are injected from the web form into the databse of an application to change the databse content or dump the database information like creditcard or password to attacker.

Most web applications include a back-end database, either running on a separate database server, or installed on the same machine as the web server itself. SQL Injection is one of the most common application layer attack techniques used today[1].

### *A. Web Application Architecture*

Web application commonly has three tiers: presentation, logic, and storage.

The presentation tier is the topmost level of the application. It displays information related to such services as browsing merchandise, purchasing, and shopping cart contents, and it communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network. [2]

The logic tier is pulled out from the presentation tier, and as its own layer, it controls an application's functionality by performing detailed processing.

The data tier consists of database servers. Here, information is stored and retrieved. This tier keeps data independent from application servers or business logic.

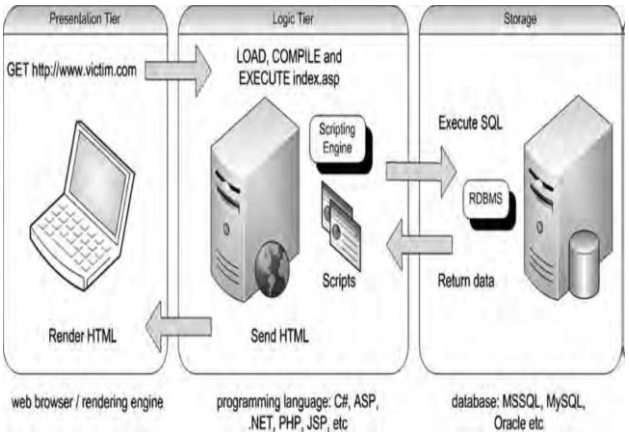


Fig.1: Web Application Architecture

Giving data its own tier also improves scalability and performance. In this Figure, the Web browser (presentation) sends requests to the middle tier (logic), which services

 

them by making queries and updates against the database (storage).

A fundamental rule in three-tier architecture is that the presentation tier never communicates directly with the data tier; in a three-tier model, all communication must pass through the middleware tier.

In this Figure, the user fires up his Web browser and connect to http://www.victim.com. The Web server that resides in the logic tier loads the script from the file system and passes it through its scripting engine, where it is parsed and executed.

The script opens a connection to the storage tier using a database connector and executes an SQL statement against the database.

The database returns the data to the database connector, which is passed to the scripting engine within the logic tier.

The logic tier then implements any application or business logic rules before returning a Web page in HTML format to the user's Web browser within the presentation tier.

The user's Web browser renders the HTML and presents the user with a graphical representation of the code. [4]

### B. GreenSQL

GreenSQL database firewall used to protect databases from SQL injection attacks.

GreenSQL works as a proxy for SQL commands and has built in support for MySQL & PostgreSQL.

GreenSQL secure database from unauthorized access by monitoring every SQL command sent to the databases. GreenSQL helps business secure their information assets and demonstrate regulatory compliances.

The logic is based on evaluation of SQL commands using a risk scoring matrix as well as blocking known db Administrative commands (DROP, CREATE, etch). GreenSQL is distributed under the GPL license.

### C. Calculating a query's risk

GreenSQL calculates each query's risk. Essentially, this is an anomaly detection subsystem. After the risk is calculated, GreenSQL can block the query or just create a warning message (this depends on the application mode). There are a number of heuristics GreenSQL uses when calculating risk [6] For example, query risk is increased by:

1) Fingerprinting of Database (users, accounts, credit information)
2) Stack Based Query (Comments inside SQL commands)
3) An SQL expression that always returns true (SQL tautology)[10]

## II. SQL TAUTOLOGY

This type of attack injects SQL tokens to the conditional query statement to be evaluated always true. This type of attack used to bypass authentication control and access to data by exploiting vulnerable input field which use WHERE clause.

"SELECT * FROM employee WHERE userid = '112' and password ='aaa' OR '1 '='1 III

As the tautology statement (1=1) has been added to the query statement so it is always true.[11]

## III. MODES OF GREENSQL

The GreenSQL db firewall can be used in a number of ways:[7][8][9]

1) IDS
2) IPS
3) Learning mode
4) Database firewall

### A. IDS Mode

During Simulation Mode basically nothing is blocked. GreenSQL works as a database IDS system (IDS stands for Intrusion Detection System).During this mode, our risk scoring matrix engine identifies suspicious queries and notifies the database administrator using the GreenSQL Management Console.

### B. IPS Mode

When the system is configured to Block Suspicious Commands, GreenSQL uses its heuristics engine to find "illegal" queries and block them automatically. In this mode, GreenSQL is basically a database IPS system (IPS is Intrusion Prevention System). If a query is considered illegal, a whitelist is checked. If it is found in the whitelist, it will be redirected to the genuine MySQL server. If it is found to be "illegal", GreenSQL will return an empty result set to THE application. During this mode, GreenSQL can sometimes generate false positive and false negative errors.

As a result, some legal queries may be blocked or the GreenSQL system may pass through an "illegal" query undetected. These are the pros and cons of IPS systems. GreenSQL is constantly improving its heuristics engine but it is still not perfect.

### C. Learning Mode

The above methods we recommend enabling Learning Mode and then, after the learning period is over, switching to the Active protection from unknown queries.

During the learning mode, all queries are automatically added to the whitelist. When the learning mode is over, GreenSQL automatically enables active protection.

### D. Database Firewall Mode

When Active protection from unknown queries mode is enabled, all unknown commands are blocked. This is database firewall mode. When an unknown SQL command is detected, it is automatically blocked.

 

## IV. GREENSQL ARCHITECTURE

GreenSQL works as a reverse proxy for MySQL connections. This means, that instead of connecting TO THE MySQL server, your applications will connect to THE GreenSQL server. [9]

GreenSQL will analyze SQL queries and then, if they're safe, will forward them to the back-end MySQL server.

The following picture describes the whole process. [12][7]

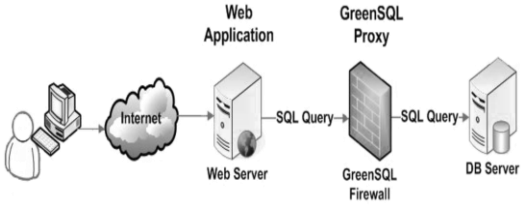


Fig 2: GreenSQL Architecture

As you can see, GreenSQL calls the real database server to execute SQL commands and the web application connects to the GreenSQL server as if it were a real database server.

GreenSQL can be installed together with the database server on the same computer or it can use a distinct server. By default GreenSQL listens on local port 127.0.0.1:3305 redirecting SQL requests to 127.0.0.1:3306 (the default MySQL setting). These settings can be altered using the GreenSQL Console.

## V. CONCLUSION

1) Provides great security regarding different sql attacks.
2) Can be used as a penetration tool!!!

## REFERENCES

[1] Justin Clarke, SQL Injection Attacks and Defense, Second Edition, Syngress Publication, July 2, 2012,ISBN-13: 978-1597494243
[2] GreenSQL Available: http://www.greensql.net/
[3] GreenSQL Available: http://www.howtoforge.com/preventing-mysql-injection-attacks-with-greensql-on-debian-etch
[4] GreenSQL Available: http://www.greensql.com/docs
[5] GreenSQL Available: http://www.greensql.com/support/knowledge
[6] GreenSQL Available: http://www.greensql.com/why-greensql-db-security/top-10-reasons
[7] GreenSQL Available: http://community.spiceworks.com/topic/212598-database-firewall-greensql
[8] GreenSQL Available: http://www.techrepublic.com/blog/opensource/firewall-mysql-with-greensql/317
[9] GreenSQL Available: http://opensourcedba.wordpress.com/2012/03/26/database-firewalls-from-oracle-and-greensql/

 