TOPIC: SQL Injection Prevention Tools

School Of Cybersecurity

Major: Cybersecurity

Name: Olivia Woodward

ID: 01249775

CYSE 250

ABSTRACT

SQL is a widely used programming language for many web developers for its accessibility and ease of use. It is so widely used that companies offer servers premade with SQL databases for developers. However, with SQL comes a price. Without the proper protection, an SQL injection can easily be made, and a website can lose all the valuable information stored in its database in mere seconds. This is an important topic in this technology age because there are so many websites, and people use these sites just as much as developers make them. With technology rapidly progressing, attackers are also finding ways to bypass the SQL injection tools already implemented. This essay explains the history of SQL, its vulnerabilities, commonly used prevention methods, and finally new research that presents alternative methods to preventing injection. The innovative tools of SQL injection prevention are Software-Defined Networking Application Firewalls.

INTRODUCTION

Think of the last time someone has used a website to log in. Most likely, that was only a few milliseconds ago. Websites have become an integral part of humanity, being used everywhere from the classroom to the workplace. Although every site looks different, they most likely have the same blueprint: a website, a programming language, and a database all connected by SQL. SQL stands for Structured Query Language. SQL is commonly used for websites that need to interact with a client. The website sends requests of the client to the programming language, the programming language then makes queries in the database and sends the correct information from the database to the website, giving the client the desired request (Clarke-Salt, 2009). There are many different providers for having a premade SQL server instead of making them from scratch. For example, one of the most prominent SQL server providers is Microsoft's SQL

Server. On Microsoft's SQL Server site, they have many different models of servers based on the clients needs such as security, cloud, or performance ("Try SQL Server on-premises or in the cloud"). Although there are many different types of servers, most web developers ought to always put security into consideration when making an SQL based website because of SQL injections.

SQL INJECTION

A SQL injection happens when a web developer does not validate the input coming from the client side of the site (Alotaibi & Vassilakis, 2023). This can lead to the values messing around with SQL query sent to the database if an attacker puts in an input that can be interpreted as code (Alotaibi & Vassilakis, 2023). This input of code is the SQL "injection" because it is injecting new code into the program. There are multiple different types of SQL injection.

- Injection through user input

  The most commonly used one is injection through user input. This happens when an attacker uses client submission, such as a login page or prompt to enter information, for the web application to read and then execute (Hanford & Viegas, 2006).

- Injection through cookies

  Injections can also use cookies. Cookies are files that are stored on a user's device that is used to restore the client's information when they access the site again (Hanford & Viegas, 2006). Because cookies are client-side, an attacker can act as a client, and then modify the cookies on their device to be SQL queries to attack the site when they visit again (Hanford & Viegas, 2006).

- Injection through server variables

  Injection can also happen through server variables such as: HTTP, network headers, and environmental variables (Hanford & Viegas, 2006). These variables can be used to gauge

statistics on the site such as which pages are visited the most; attackers can modify these variables and SQL inject into them (Hanford & Viegas, 2006).

- Second-order injections

  Finally, there's second order injections, that are more complex and can happen at a later time than the initial injection. The attacker plants the seed in the database or system that will indirectly trigger the injection during usage of the input (Hanford & Viegas, 2006).

The consequences of having a SQL injection happen are very broad because injections can work to attack almost anything in a database. It can even be connected to other types of cyber-attacks. Take identity spoofing for example. An attacker can make it so that a website or email address looks like the genuine address they've injected into. In reality, it is the attacker and due to the client being oblivious to this, can gain massive amounts of sensitive information because they are disguised as a trustable source (Singh et al., 2016). It can also do things to alter a site's information like changing prices, DDoS so that it cannot be accessed by real clients, and halting important transactions (Singh et al., 2016).

This is obviously a big problem because there are so many websites made today, and more so, many that ask for sensitive information. For example, think about a prominent bank's website. What if an attacker was able to send a SQL injection into their site, and ask for the account and routing numbers of each client that has a login? That would be detrimental not only to the bank, but to their loyal customers as well, as their private information is no longer private. Unfortunately, this hypothetical situation has happened in real life before. In 2007, the 7-Eleven database was hacked, causing customer's debit cards to be accessed and drained of money ("SQL Injection", n.d.). This is all the more reason to keep a close eye on how sites are storing, securing, and privatizing their client's sensitive information.

SQL DETECTION

Before one can act on preventing SQL injection, one must determine if their database has been compromised. Although some argue that detection is useless because the attacker has already

compromised the system, it is still important because injections, if done stealthily enough, can go unnoticed by even developers. There are multiple ways to do this:

- Check IIS log

  The IIS log records a user's IP address and access files. The file size and content can be modified if an SQL injection has taken place (Qian et al., 2015).

- Check database

  An attacker can modify the look of SQL database tables in order to look like nothing is wrong. Then, if the table contains sensitive information, can snatch the values from the tables when developers aren't looking. By checking the table's content and structure, you can evaluate whether it is compromised (Qian et al., 2015).

- Check user input

  Developers can limit the length of or validate user input in order to deter against SQL injections. This includes checking the input's format, length, type, and range (Qian et al., 2015).

But detecting injections is not the equivalent to preventing or defending against them, these are just strategies that allow developers to be alerted of an injection.

SQL INJECTION PREVENTION

SQL is tricky because it is not an attack you can actively fight against. Once it happens, it happens. So, the only way to stop them is to prevent them from happening in the first place. Although SQL is dangerous because of injections, it has been around long enough for basic prevention measures. The most common prevention method is to use prepared statements. According to a group of researchers at the 2nd International Conference on Information Science and Systems, a prepared statement is a template repeated for all database statements. During a query, for example, the constant values will be replaced by something else (Castillo et al., 2019). When the values are submitted, the application compiles and optimizes the query and then holds onto the results for a later execution time (Castillo et al., 2019). This hinders SQL injection

because it binds the parameters of, for example, a username and password, to the databases for usernames and passwords. If the injection doesn't match any of the values in there – which is likely because usernames usually are practical information that relates to a client – the code returns an error saying that the input was invalid (Castillo et al., 2019). The prepared statement is easily applicable to SQL code and is one of the reasons why it is so common.

Although there are specific strategies like the prepared statements that are common guides for developers, that is only one of few. Most SQL prevention isn't specific tools because each website is unique and not always malleable to what's out there. SQL injection prevention depends on how meticulous the developer is with protecting their code. According to researchers at the 2015 International Conference on Estimation, Detection, and Information Fusion, common ways to validate SQL code are to: make no assumptions about type, size or content of data received by application; check the content of string variables; accept only expected data; verify size and value type of input; use stored procedures to validate; and do not concatenate user input (Qian et al., 2015). All of these are easily applicable to SQL code because it is just a matter of modifying the code. For example, accepting only expected data might be as simple as putting an "if" statement into the code to see if the data type is the correct one.

In recent years however, researchers have been able to make major breakthroughs. University of York researchers and lecturers Alotaibi and Vassilakis from University of York, UK found that using software-defined networking (SDN) firewalls was considerably strong against SQL injections. Software-defining networking is moving decision making moves form switches to a centralized controller (Alotaibi & Vassilakis, 2023). SDN firewalls are a fairly new venture in cybersecurity, with the researchers admitting that their paper is the first of few to talk about them. The SDN firewall works like any other firewall and uses signature detection to determine whether or not an attacker machine is using an SQL injection tool. A signature is like a pattern that is detected in a database, each one being unique. So, if an attacker is not careful and their signature is not like what the firewall has seen before, the firewall is bound to block them. The SDN firewall also uses regular-expression-based detection, which uses a guide for detecting multiple signatures instead of one at a time (Alotaibi & Vassilakis, 2023). While the SDN firewalls lag behind in reducing CPU usage – with the signature type being the same rate as a

regular company bought firewall and a regular-expression-based SDN firewall being two times that – it makes up for it with impressive latency (Alotaibi & Vassilakis, 2023).

CONCLUSION

SQL is a tricky concept to protect because there are so many ways it can occur. There is no "one method fits all" for website SQL databases because each one is different. Luckily, researchers have been able to come up with multiple prevention strategies because of the time they've had to be able to study SQL. As long as developers are careful to validate and filter the input coming from client computers, most worries of an SQL injection are dissipated. An SDN web application firewall will help even more so, helping block out the attack before it even had a chance to begin.

However, as technology keeps on advancing, so will attackers. There will be a time where the tools discussed in this paper will be futile. With that in mind, it is important to support the researchers currently exploring new ways to protect sites, so it is possible to keep up with their pace while also using sites in peace. But until that happens, the only choice is to keep oneself knowledgeable on topics such as SQL injection so they can secure themselves.

Works Referenced

Alotaibi, F.M., Vassilakis, V.G. (2023). Toward an SDN-Based Web Application Firewall: Defending against SQL Injection Attacks. *Future Internet, 15*(170), 1-15. https://doi.org/10.3390/fi15050170

Castillo, R.E., Caliwag, J.A., Pagaduan, R.A., Nagua, A.C. (2019). Prevention of SQL Injection Attacks to Login Page of a Website Application using Prepared Statement Technique. *ICISS '19, March*(2019). 171-175. https://doi.org/10.1145/3322645.3322704

Clarke-Salt, J. (2009). *SQL Injection Attacks and Defense* [2nd Edition]. Syngress. https://learning.oreilly.com/library/view/sql-injection-attacks/9781597499637/

Hanford, G.J.W., Viegas, J., Orso, A. (2006). A Classification of SQL Injection Attacks and Countermeasures. Retrieved April 13, 2024, from https://sites.cc.gatech.edu/home/orso/papers/halfond.viegas.orso.ISSSE06.pdf

TOPIC: SQL Injection Prevention Tools

Qian, L., Zhenyuan, Z., Hu, J., Shuying, L. (2015). Research of SQL Injection Attack and Prevention Technology. *2015 International Conference on Estimation, Detection, and Information Fusion (ICEDIF 2015).* 303-306. https://doi.org/10.1109/ICEDIF.2015.7280212

Singh, N., Dayal, M., Raw, R.S., Kumar, S. (2016). SQL Injection: Types, Methodology, Attack Queries and Prevention. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom).* 2872-2876. https://ieeexplore.ieee.org/abstract/document/7724789/citations

SQL Injection. (n.d.). Malware Bytes. Retrieved April 12, 2024, from https://www.malwarebytes.com/sql-injection

Try SQL Server on-premises or in the cloud. Microsoft. Retrieved April 12, 2024, from https://www.microsoft.com/en-us/sql-server/sql-server-downloads