Option #2: Securing the Environment

Liam Bergerson

CSU - Global

ITS411-1

D. Morrill

5/1/22

Option #2: Securing the Environment

SQL injection attacks have the potential to damage an organization on a large scale and they usually are not taken seriously by organizations. In 2020, 736 million web attacks were recorded against financial institutions and around 33% of them were SQL injections attacks (Pitchkites, 2022). SQL injection attempts to gain access to a database by injecting malicious SQL code into a web form (Sheposh, 2020). Organizations need to be aware of SQL injection so they can take the proper security measures to ensure their databases do not fall prey to a data breach caused by SQL injection.

Testing for SQL Injection Vulnerability

In order to test sites to see if they are vulnerable to SQL injection, I would start off by using SQLMAP. SQLMAP is a tool widely used by penetration testers to test to see if a website is vulnerable to SQL injection. More specifically, SQLMAP tests the GET parameter that requests resources from a database for vulnerabilities. If an asterisk is inputted for the GET parameter and it results in a syntax error, then that website is vulnerable to SQL injection. An asterisk requests all data from the database instead of specific data. To further test for SQL injection vulnerability, SQLMAP can be used in a variety of steps. The first step is to list the information about existing databases associated with the web URL. The next step is to list information about any tables present in a specific database. The next step is to list information about any columns in tables. Finally, the last step is to try to dump the information in the columns of the database.

Data dumped from vulnerable websites allow for unauthorized users to explore the database and look for any valuable information to steal.

Inferential Testing

Inferential testing is a version of blind SQL injection which asks the database true or false questions and determines the answer based on the application's response. Inferential testing can be used when a web application is configured to show error messages but has not mitigated the code that is vulnerable to SQL injection (OWASP, n.d). When the database is configured to not output data to the web page when a typical SQL injection attempt is made, inferential testing will be used to try and extract data from a database by asking the database a variety of true and false questions.

Identifying Dangerous Source Code

Any faults in the source code of a website can affect the integrity of the website in a dangerous way; it could leave the website vulnerable to a SQL injection attack. A plan to analyze the source code to check for those faults must be required before the website is released or updated. Source code analysis is the strategy that will be used in order to check for any faults in the source code. Source code analysis is basically automated code debugging and the goal is to find any bugs or faults in the source code that can be missed by the developer (Techopedia, 2014). Examples of possible subtle faults in the source code can include buffer overflows, unorganized pointers, and garbage collection functions. All of these faults could be exploited by a hacker in a SQL injection attack

(Techopedia, 2014). Code analyzers need to be set up properly in order to detect all possible faults in source code. If an analyzer is not precise enough, it will output too many false positives which will flood the user with useless warnings. If an analyzer is too precise, it will take too long in order to detect all possible faults. For all future code changes, no new code will be added to the progression environment without going through an automated code debugging regression test. Once the regression test passes, then the code can be deployed.

Methods for Preventing SQL Injection

I will offer several suggestions in order to fortify websites and prevent them from being exploited by SQL injection. Input validation is the process that verifies if the input that the user submitted is a data type that is allowed. It makes sure that the input meets type, length, and format requirements (PT, 2020). If they do not meet the requirements, the user input is not accepted. Depending on how long an input can be, SQL scripts inputted will not be accepted because they may be too long.

It is important to always use character-escaping functions for user-supplied input. Doing that will mitigate SQL injection by ensuring that the Database Management System does not confuse any SQL code inputted by the user with the SQL code written by the developer (PT, 2020). SQL injection scripts work to trick the database by executing the code provided by the malicious user, not the original script. By using character-escaping functions, the database will continue

to use the original script provided by the developer and will not run the malicious script provided by the user.

It is crucial to avoid connecting your website to a database with an account that has root access. If this is done, the attacker will have access to the entire system if they gain access to the account through SQL injection. It is important to implement the principle of least privilege to ensure that any account that is compromised will have limited privileges. Instead of focusing on which rights to take away from an account, focus on figuring out which privileges a user needs in order to do their job (PT, 2020).

A web application firewall (WAF) can be used to prevent multiple security threats such as SQL injection. A WAF is a firewall that sits in the front of the web servers and monitors traffic coming in and out of the website (PT, 2020). A WAF can be configured with security rules that can prevent a SQL injection attack. For example, a configured WAF will monitor the GET and POST requests and block any malicious traffic. New rules and policies can be added to the WAF with ease and will act immediately upon confirmation. Not only can a WAF prevent SQL injection attacks, but it can prevent cross-site scripting, session hijacking, distributed denial of service attacks, and so on (PT, 2020).

Conclusion

A majority of data breaches are the result of SQL injection attacks. Most organizations do not even take precautions to defend against SQL injection. Ragan (2014) states that around 52% of respondents to a study do not take

precautions to defend against SQL injection. Since SQL injection attacks are quite common, organizations need to improve upon security measures to prevent data breaches from SQL injection.

References

OWASP. (n.d.). *Blind SQL Injection | OWASP Foundation*. https://owasp.org/www-community/attacks/Blind_SQL_Injection

Pitchkites, M. (2022, March 22). *26 Cyber Security Statistics, Facts & Trends in 2022*. Cloudwards. https://www.cloudwards.net/cyber-security-statistics/

PT. (2020, December 4). *How to prevent SQL injection attacks*. Positive Technologies. https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-prevent-sql-injection-attacks/

Ragan, S. (2014, April 16). *Organizations suffer SQL Injection attacks, but do little to prevent them*. CSO Online. https://www.csoonline.com/article/2144261/organizations-suffer-sql-injection-attacks-but-do-little-to-prevent-them.html

Sheposh, R. (2020). *SQL Injection*. Salem Press Encyclopedia of Science. https://eds-s-ebscohost-com.csuglobal.idm.oclc.org/eds/detail/detail?vid=0&sid=a7a12427-70de-485c-a2f6-32817674856c%40redis&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=125600140&db=ers

Techopedia. (2014, February 26). *Source Code Analysis*. Techopedia.Com. https://www.techopedia.com/definition/29997/source-code-analysis