Anomaly Scoring and SQL Injection

James Sturges III
Department of Computer Science, Furman University, Greenville, SC

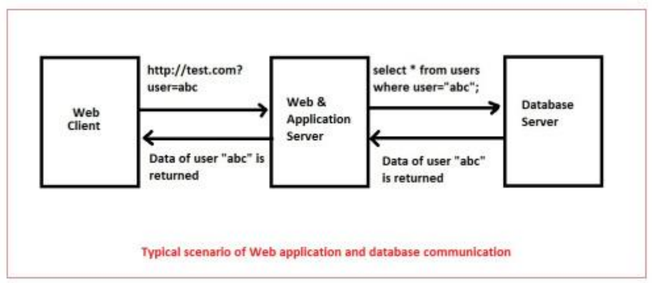
Introduction

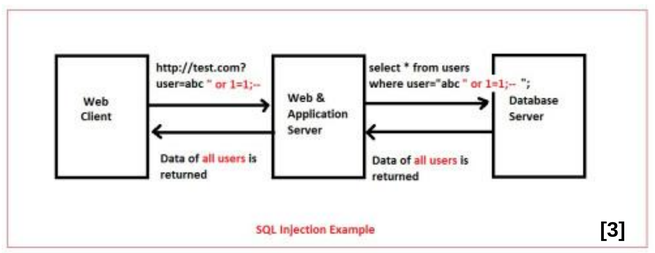
SQL Injection is one of the oldest attacks on the Internet, and is one of the most well-known, but still thousands of websites fall prey to this attack [1].
I wanted to see how difficult it would be to create a website that could effectively protect against SQL Injection Attacks (SQLIAs).

Validation

Filtering and validating user input is a quick and easy way to avoid attacks, and can be done in one statement with PHP's *filter_var*() function. Filtering removes all HTML tags and evaluable tokens, while validating makes sure the user's input is of the correct type. For example, if a site asks for a 4-digit PIN, validation would reject all PINs that are not 4 digits

SQL Injection





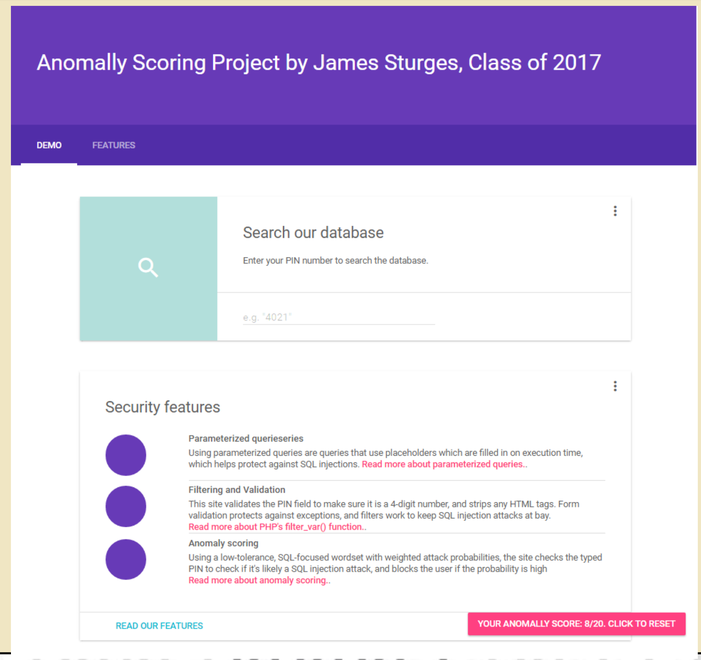
[3]

SQL, or Structured Query Language, is the tool of choice for thousands of web developers worldwide for interacting with databases.
SQL Injection takes advantage of the fact that SQL does not inherently distinguish between user-inputted data and the original intent of the SQL query.
For example, take the following query:

*SELECT * FROM customer WHERE cust_id = $cust_id*

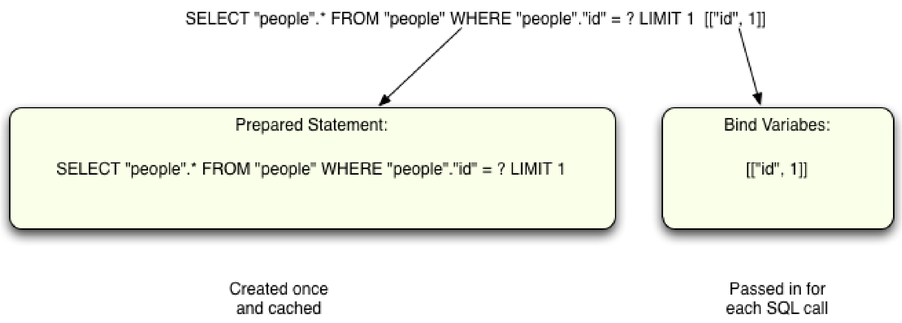
If someone had input 50' OR '1=1, the query would return all customer info. Input must be sanitized and parameterized

Demonstration Website



My website, hosted on cs.furman.edu/~jsturges/securityproject, hosts fictional employee data, accessible with a PIN. The site acts as an interactive front-end that shows how an anomaly score works and provides information on the features the site uses to keep this data safe. If an anomaly score becomes too high, the user will be locked out, and have to click a reset button. The site was coded in HTML, CSS, and PHP.

Prepared Statements



Prepared statements embrace the idea of encapsulation, that is, break code up into discrete functions.
Instead of passing in raw user input as variables to lines of SQL code, we parameterize the query first.
A prepared query is a full SQL query with user-inputted values resolved to variables to pass in later, called binding.
Prepared statements are cleaner, faster, and overall more safe.
Prepared statements are supported in SQLi and PDO, but not in the original SQL. Therefor, using the original SQL is not recommended.

Anomaly Scoring

[2]

Anomaly scoring works by taking an entire user input for a SQL query and analyzing its parts to determine the likelihood of malicious intent. For example, with a last name field, if "1-1" is typed by itself, the score might be incremented by 3, including an "OR" may increment the score by 3, and including "Drop table" may increment the score by 10.
My implementation of anomaly scoring is only meant to protect against SQL injection, but if my site were to display user data, protecting against Cross-Site Scripting would be useful.
Anomaly scoring is recommended by Akamai technologies to identify actual threats [2].

Results

The findings were that implementing secure SQL code handling were not particularly difficult or involved, and once a prepared query was written once, it would be easy to use it elsewhere.

Additionally, using basic anomaly scoring, false positives can be reduced, leading to lower amounts of user frustration. With a more advanced anomaly scoring algorithm, even better attacker prevention could be achieved.

Acknowledgements

Website utilizes Google's Material Design Lite, available at https://www.getmdl.io/
[1] https://blog.sucuri.net/2014/11/most-common-attacks-affecting-todays-websites.html
[2] https://blogs.akamai.com/2013/12/anomaly-scoring-is-a-better-way-to-detect-a-real-attack.html
[3] http://itsecurityconcepts.com/2013/11/04/sql-injection/
[4] http://patshaughnessy.net/assets/2011/10/22/prepared%20statement.png