Task 2 Report

GAMK

When checking a SQL query for malicious injections we took the approach of going through the most popular attacks and attempting to detect those. This will require the tool to be up to date on the latest SQL attacks, but this will be done through updates. We decided to rank a query on a scale of 0% - 100% (We never say 100% because we can never be certain). For each of the warning keywords that is sees, we add 10% to the total score, and if we see two of them, we add 50%. This is because when two or more of the warning keywords are present it is much more likely to be an attack.

The first injection attack type we defend against is tautologies. We decided to check for the common `1=1` and `--` in the SQL query. The `1=1` allows the query to always return true and the `--` comments out the rest of the query. This could be dangerous because if the query always returns true and the attacker add something to the query it will give the attacker what he wants.

The next attack type we defended against was Illegal or logically incorrect queries. These are queries that will cause the database to return an error and this could give the attacker information on what database we are using, and will give him more tools to attack us with. We decided to check for `convert`, `int`, `sysobjects`, and `xtype`. Now these could be in an actual query and so this is why we decided to use a point system. We cannot be absolutely sure if a certain query is an attack.

We next check for a union query. This is a tough one to deal with because a union might be in the actual SQL query. This is why we add more to the score if we see more than 2 of the warning keywords.

The next attack we checked for was a piggy-backed query. This is typically defined with a `;` and a `--`. This creates a new malicious query inside of the normal query and then comments out the rest of the normal query. This is again checked using the two warning keyword method.

The next attack we check for is a stored procedures attack. We will detect it by looking for `@` and common procedures such as `shutdown` coupled with `;`.

We check for an inference attack next. The ways we checked for this was the two most common ways. An always false equation `1=0` and the next was running a command after a malicious query to see if it worked. We checked for the common keywords of `ASCII`, `WAITFOR`, and `SUBSTRING`.

The last attack we checked for was the alternate encoding attack. We checked for attackers trying to execute code using some sneaky encodings. We checked for `EXEC`, `CHAR` and `0x`. These can send an instruction to the machine while making it unintelligible to humans.