Уязвимости в Go сервисе.

SQL INJECTION   
В ходе анализа кода была выявлена уязвимость в логике обрабатывания SQL запроса в строке 38,   
так как переменная ‘query’ формируется используя ‘fmt.Sprintf’ путем добавления ‘searchQuery’ в сам SQL запрос.  
Эксплуатация уязвимости данной функции может привести к тому, что атакующий может манипулировать ‘searchQuery’ вписывая запрос по типу ‘”; DROP TABLE products; --‘, что само по себе является большой угрозой для продакшна и приведет к большому ущербу. Данная уязвимость может привести как к несанкционированному просмотру, так и к изменению или удалению нужной информации в базе данных, что нарушает доступность и целостность данных (что само по себе по триаде кибербезопасности является критическим).  
Хорошим решением будет использовать подготовленные выражения с плейсхолдерами для инпута и входных параметров. В свою очередь, это обеспечит обработку ввода юзера как отдельных данных, а не часть SQL запроса. Пример замены уязвимого кода:

  
решение данной уязвимости

Отсутствие валидации ввода  
Также, необходимо подметить, что отсутствие валидации инпута является уже само по себе большой уязвимостью, как например в строке 32,   
где подобный подход может привести к другим уязвимостям отличающимся от SQL Injection, таких как XSS и т.д.   
Последствия данной уязвимости зависят напрямую от контекста, но сама по себе она является критической, так как открывает ещё большее пространство для эксплуатации.  
Решением будет реализовать грамотную и качественную валидацию ввода. Убедиться, что он будет совпадать по форматам, длине, типу и проверке на использования запрещенных символов перед использованием его в запросах или операциях.

Среди решений наиболее критическим и универсально рекомендуемым является использование подготовленных выражений для запросов к базе данных. Это не только исправит потенциальную уязвимость SQL Injection, но также обеспечивает уровень проверки ввода, гарантируя, что ввод обрабатывается в ожидаемом нам контексте и ожидаемым нам образом. Подготовленные выражения являются отличной практикой для операций с базами данных и предлагают преимущества не только в безопасности, так и в производительности за счёт предварительной компиляции и кэшировании запросов.

Объединив эти две меры, приложение значительно улучшит свою защиту от общих веб-уязвимостей, особенно от SQL Injection, которая является одним из самых серьёзных рисков безопасности.