

**Dynamic Vulnerability Assessment**

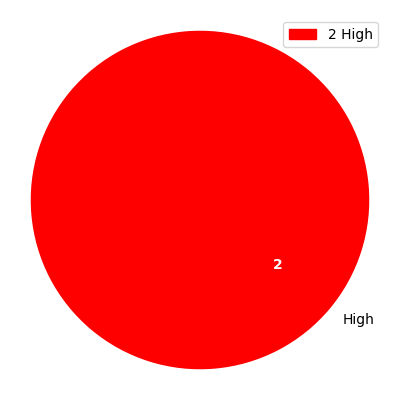
**sdgrg**

**Requested by:** rwgghwerg

## Version Information

|  |  |  |
| --- | --- | --- |
| Date | Application Version | Reviewer |
| 22-May-2025 | Initial Draft | wrgrg |
| 22-May-2025 | Peer Review |  |
| 22-May-2025 | Approved |  |

**Vulnerability Severity Distribution**



## Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Security Observation | Risk Rating | Page No. |
| **High Severity** | | | |
| 1 | Insecure Transport: Weak SSL Ciphers | High | 4 |
| 2 | SQL injection | High | 5 |

## URLs and Scope

URLs: rsfgrgrg

Scope: rgergerg

## Vulnerability Details

### 1. Insecure Transport: Weak SSL Ciphers

**Severity: High**

CVSS Score: 8.3

CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:H

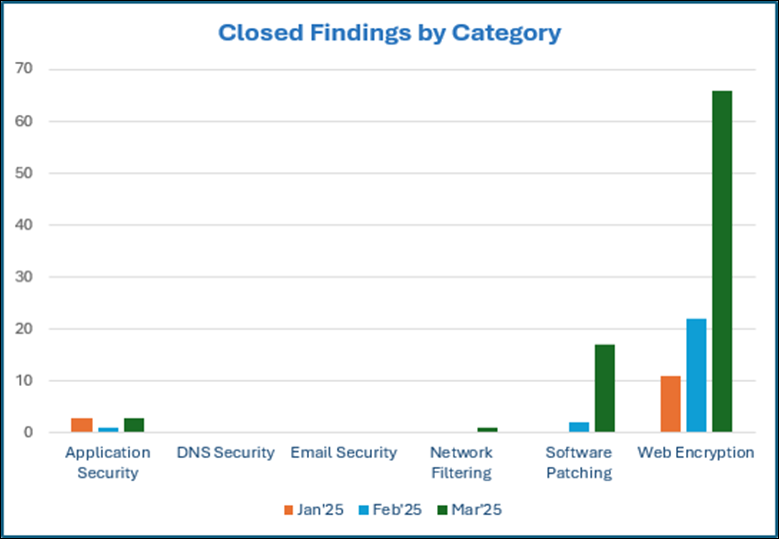
#### Description

Improper The Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols provide a  
mechanism to help protect authenticity, confidentiality and integrity of the data transmitted between a  
client and web server. The strength of this protection mechanism is determined by the authentication,  
encryption and hashing algorithms. These are collectively known as a cipher suite chosen for the  
transmission of sensitive information over the TLS/SSL channel. Most web servers support a range of such  
cipher suites of varying strengths. Using a weak cipher or an encryption key of insufficient length, for  
example, could enable an attacker to defeat the protection mechanism and steal or modify sensitive  
information. If misconfigured, a web server could be manipulated into choosing weak cipher suites. A  
weak encryption scheme can be subjected to brute force attacks that have a reasonable chance of  
succeeding using current methods and resources. An attacker could possibly execute a man in the middle  
attack which would allow them to intercept, monitor and tamper with sensitive data. Each weak cipher  
was enumerated by establishing an SSL connection with the target host and specifying the cipher to test  
in the Client Hello message of the SSL handshake.

#### Evidence

Step 1: erbhergherhgerg





Step 2: edrhgdhdeh



#### Recommendation

It is recommended not to use RC4, CBC,SHA, SHA1, MD5 etc ciphers

#### Reference

https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening/  
http://zero.webappsecurity.com

### 2. SQL injection

**Severity: High**

CVSS Score: 7.6

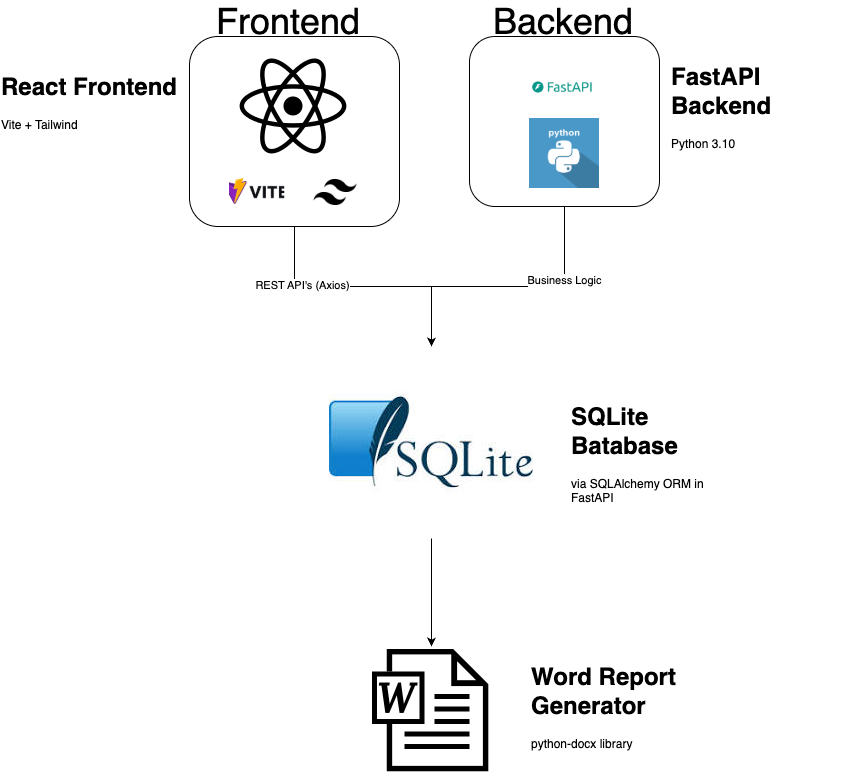
CVSS Vector: CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:C/C:H/I:H/A:H

#### Description

SQL Injection vulnerabilities have been identified in the web application. SQL injection is a method of attack where an attacker can exploit vulnerable code and the type of data an application will accept and can be exploited in any application parameter that influences a database query. Examples include parameters within the url itself, post data, or cookie values. If successful, SQL Injection can give an attacker access to backend database contents, the ability to remotely execute system commands, or in some circumstances the means to take control of the server hosting the database.Recommendations include employing a layered approach to security that includes utilizing parameterized queries when accepting user input, ensuring that only expected data is accepted by an application, and hardening the database server to prevent data from being accessed inappropriately.

#### Evidence

Step 1: sfgjfghjfgj



#### Recommendation

Each method of preventing SQL injection has its own limitations. Therefore, it is wise to employ a layered approach to preventing SQL injection and implement several measures to prevent unauthorized access to your backend database. Use the following recommendations to code web applications that are not susceptible to SQL Injection attacks. Parameterized Queries: SQL Injection arises from an attacker's manipulation of query data to modify query logic. The best method of preventing SQL Injection attacks is thereby to separate the logic of a query from its data. This will prevent commands inserted from user input from being executed. Validate input: The vast majority of SQL Injection checks can be prevented by properly validating user input for both type and format. The best method of doing this is via "whitelisting". This is defined as only accepting specific account numbers or specific account types for those relevant fields, or only accepting integers or letters of the English alphabet for others.

#### Reference

OWASP: https://owasp.org/www-project-top-ten/2017/A3\_2017-Sensitive\_Data\_Exposure.html  OWASP: https://owasp.org/www-community/attacks/SQL\_Injection   
  
Acunetix: https://www.veracode.com/security/sql-injection   
  
Veracode: https://www.veracode.com/security/sql-injection