

CHAPTER 5 - CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

A comparison between manual testing and GSTF shows clear advantages of using an automated framework for security testing. GSTF proved to be highly efficient, achieving a 99% improvement in execution time compared to manual testing. While manual testing uncovers vulnerabilities such as SQL Injection and XPath Injection, it is time-consuming, inconsistent, and relies heavily on human expertise, making it less suitable for large-scale or complex testing scenarios. In contrast, GSTF consistently identified vulnerabilities across multiple endpoints, including more complex issues such as improper handling of edge cases, with significantly reduced test time and fewer errors. However, the framework showed a higher error rate during execution, primarily due to invalid input that required conversion to an RPC status code, which was successfully mapped for validation. Furthermore, although GSTF is relatively fast, it still has the potential for false positives, which has been proven when performing manual verification with the Proof-Based method.

5.2. Recommendations

While GSTF offers advantages over manual testing, the following recommendations are provided to further improve testing efficiency:

1. **Use GSTF for Routine Testing:** Organizations should use GSTF as the primary tool for gRPC security testing to improve efficiency, ensure consistency, and reduce testing time and human error.
2. **Improve Error Handling in GSTF:** Address high error rates by improving input validation mechanisms to minimize unnecessary conversions and increase success rates.

3. Hybrid Approach for Complex Vulnerabilities: Combine GSTF with manual testing for vulnerabilities that require human validation, such as business logic weaknesses or competitive conditions, to ensure comprehensive coverage. Include manual verification to measure false positives in GSTF execution results.
4. Integrate GSTF with CI/CD Pipelines: Incorporate GSTF into DevSecOps workflows to automate security testing as part of the software development lifecycle, further increasing its utility and efficiency.

