

CHAPTER 1 - INTRODUCTION

1.1. Background

The development of APIs has expanded significantly, encompassing various types such as SOAP, REST, and gRPC. Based on Paloalto's 2023 report on the consequences of API security incidents, which can be seen in Figure 1.1, it was noted that team members suffered financial losses after the incident. This caused substantial financial setbacks, especially for the affected companies. Currently, API security has become a very important requirement, and security testing needs to be carried out to mitigate risks. The aspiration is that this research can assist other communities in enhancing the security of their owned APIs, particularly focusing on the gRPC API.

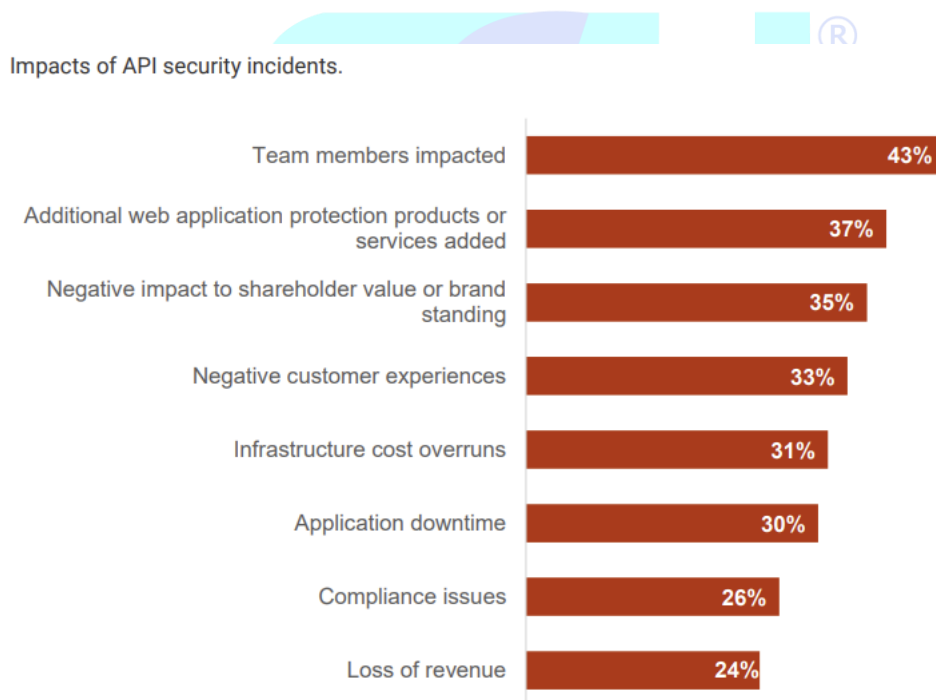


Figure 1.1 API Security Incident 2023 (Melinda Marks, 2023)

Currently, I work as a security tester for applications, particularly focusing on APIs. Despite searching for a few years, I found that API security testing tools are generally built for REST APIs and do not support gRPC natively, making testing more

difficult. (Alexandra Charikova, 2024). Consequently, the testing process is conducted manually, resulting in a time-consuming and repetitive experience. This research aims to introduce a new framework for security evaluation in gRPC, then creating automated methods and combine with DevSecOps to automate them as well, to streamline and enhance the efficiency of the testing process. DevSecOps is a combination of development, security and operations, which makes the process faster and secure because the process is carried out automatically (Zaheeruddin Ahmed and Shoba. C. Francis, 2019). With automated process and the proposed new framework, it can be used to carry out security assessments by carrying out security testing when the application has been developed to reduce the risk of vulnerabilities after the initial process and can reduce testing time significantly, especially in this project which focuses on developing API gRPC. Google Remote Procedural Call (gRPC) is an open-source RPC protocol developed by Google. gRPC has a very high processing speed and runs on the HTTP/2 protocol with messages converted into binary message format (Anthony Giretti, 2022). gRPC also uses clients to generate functions using proto files provided by the server, which are called stub (client generated in a particular language) or it can be called a gRPC client. Then, this function will later be used in making requests and getting responses. This makes the security testing of gRPC different from APIs in general, because the client will send a request using a function that has previously been created and contains a message which has been converted into a protocol buffer message and later the server will send back a response in the form of a protocol buffer that will be converted using the function response on the client.

Conducting a thorough examination of how gRPC works compared to another APIs is essential. The results of this comparative analysis will produce a set of benchmark components used in gRPC testing and help in developing a new framework as well as being an important benchmark in determining which components will be tested. In addition, security weaknesses and vulnerabilities commonly identified in another APIs will be identified and adapted to facilitate analogous testing in the context of gRPC. This project aims to develop a new security testing framework tailored to the gRPC process.

This research is intended to serve as a reference for security professionals in testing gRPC APIs by understanding the proposed framework in this thesis and to be able to solve the problem of long testing time efficiency that I get when doing gRPC testing manually. The approach involves comparing existing frameworks, identifying their weaknesses in relation to gRPC, and redesigning a new framework to align with gRPC's specific functionality.



1.2. Research Problem

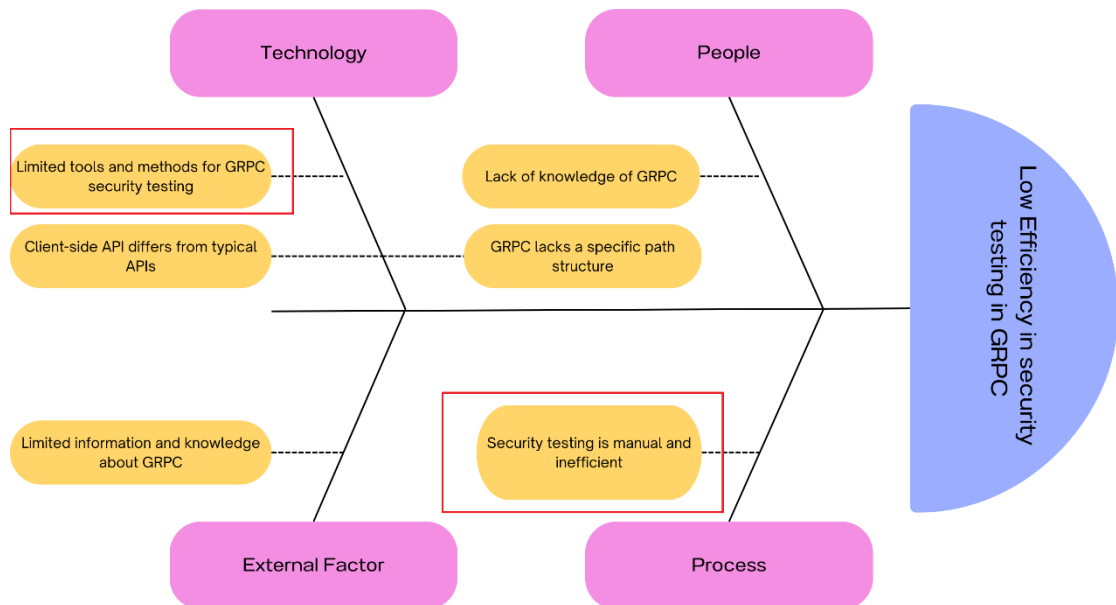


Figure 1.2 Fishbone Diagram

Based on the book "Modern API Design with gRPC" (Hitesh Pattanayak, 2024), gRPC encompasses various aspects, each of which plays a crucial role in its functionality. I have attempted to connect these aspects with the research problems from Figure 1.2 discussed in Table 1.1. The table highlights the main issues in this research, namely "Limited tools and methods for gRPC security testing" and "Security testing is manual and inefficient," which ultimately result in inefficient testing. These problems, as shown in the matrix, are related to error handling and security—key aspects that will be addressed in the current research.

Table 1.1 gRPC aspect and research problem matrix (Hitesh Pattanayak, 2024)

gRPC Aspect	Lack of Knowledge of gRPC (People)	Manual & Inefficient Security Testing (Process)	Lack of Tools & Methods (Technology)	Different Client-side API (Technology)	Lack of External Knowledge (External Factor)
Communication Protocol	✓				
Streaming	✓				
Code Generation				✓	
Data Serialization			✓		
Service Contracts		✓	✓	✓	
Error Handling		✓	✓		

Security		✓	✓		
Load Balancing			✓		
Resilience		✓	✓		

In Figure 1.2, the issue highlighted in this thesis that result in low efficiency of security testing in gRPC, based on experiences during testing and the lack of supporting security tools or methods. This is attributed to several factors:

- **Technology:** One of the problems with technology is the lack of tools and methods available for testing, which often leads to manual testing and results in inefficiencies during the process. This issue is evident in the global challenge faced by gRPC, where traditional scanning tools and methods are not applicable for gRPC testing. (Alexandra Charikova, 2024).
- **Process:** The current testing process is done manually, making it extremely inefficient and time-consuming to run multiple test cases for each service. In the current method, requests are made one by one, starting with a stub that needs to be created from a proto file each time a test is run. Payloads also need to be created individually to meet the specific requirements of the stub. Typically, a proto file contains more than five services, meaning testing time can take up to an hour or more, including understanding the process for each service. The repetitive nature of test cases, such as injection testing, further adds to the time required.

These issues make gRPC testing very inefficient, increase test time, and sometimes accidentally skip repeated tests because they are done manually.

1.3. Research Objective

1. Identify key gRPC components to evaluate their security by developing customized test cases that improve the accuracy and coverage of security testing.

2. Highlight gaps in existing security testing frameworks for gRPC and demonstrate the benefits of the proposed framework in addressing these gaps.
3. Define a new framework for gRPC security testing that enables automated test case generation by integrating stub objects with payloads, ensuring comprehensive and efficient test execution tailored to the unique workflow of gRPC.
4. Validate the applicability and effectiveness of the framework by implementing it in a tool capable of automating security testing and demonstrating increased efficiency, reduced test time, and better visibility of test results compared to manual methods.



1.4. Research Question

- Q1. Which specific components of gRPC are most vulnerable to security risks, and how can test cases be tailored for each component to mitigate these risks?
- Q2. What caused the previous method to be ineffective and what makes the proposed method suitable for increasing efficiency?
- Q3. What framework can be developed to effectively conduct security testing on gRPC, and how does it differ from existing security testing frameworks?
- Q4. What impact will the proposed framework have? Will it help improve testing efficiency?

1.5. Hypothesis

Hypothesis 1: gRPC components, such as the protocol buffer that stores messages, are the most manipulable components in testing. This can be proven by conducting a research study by mapping the components owned by gRPC and comparing them with components in other APIs and ensuring that the protocol buffer component is the right component.

Hypothesis 2: The current testing method, where stubs are manually generated and payloads are input one by one according to their data types, results in a lengthy process, especially for repetitive payloads or test cases like input injection. Therefore, the suggested method involves creating objects for both the stub and payload in advance, which will then be combined based on their data types to form a test case and hope it will be more efficient.

Hypothesis 3: A dedicated security testing framework developed for gRPC, focusing on gRPC protocols and components, will be more effective in identifying security vulnerabilities compared to generic security testing frameworks.

Hypothesis 4: The proposed framework, can be implemented in a tool using a programming language that validates each gRPC test case and can also be integrated with DevSecOps for automation that will significantly improve the efficiency of security testing by reducing manual efforts, reducing testing time by 60% until 80% or even more depending on the context of the features in the proto file, and improving the overall structure of the testing process. If the time required for testing from start to finish with the framework increases efficiency by at least 60%, then the framework is considered successfully adopted. Otherwise, the framework is considered unsuccessful. By reducing all test cases with test cases that can be covered by the proposed framework along with their time.

1.6. Research Scope and Limitation

This discussion will focus on:

1. Component: focus on specific gRPC components, specifically protocol buffers, to determine its susceptibility to manipulation during security testing.
2. Framework development: design a security testing framework that targets specific gRPC API and components, with the goal of identifying vulnerabilities unique to the gRPC API. This framework will be designed to be adaptable to new tools, making them easy to use and validate.
3. Automation and testing: process to improve efficiency and consistency in identifying security issues, allowing for early detection and mitigation of vulnerabilities.

This study has the following limitations:

1. Focus on specific gRPC components: This research will primarily focus on protocol buffers, which are hypothesized to be the most vulnerable components.

2. Framework and tool application: the framework and tool will be tested primarily in controlled environments and may not account for all real-world scenarios or edge cases in production systems.
3. Scope of testing: this research will focus on automating security testing for common gRPC API vulnerabilities, but may not address newer threats.
4. Generalizability: the framework will be tailored specifically for gRPC.

1.7. Significance of Study

This study plays a significant role in addressing the cybersecurity challenges associated with gRPC APIs, ensuring their security is critical for organizations that rely on them for inter-service communication. This study provides value in the following ways:

- Improving security for gRPC API: by identifying the most vulnerable gRPC components like protocol buffers, this study will help in the development of targeted test cases that can uncover and address specific security vulnerabilities. This proactive approach improves the security of gRPC API, reducing the risk of exploitation and data breaches.
- Development of a custom security testing framework: the new framework developed through this study will be tailored specifically for gRPC, unlike existing generic security testing frameworks. This ensures that vulnerability detection can be used on gRPC API as it adapts to the way they work.

Ultimately, the potential societal impact of this research lies in improving gRPC testing practices, reducing future cybersecurity risks, and ensuring the effective flow of goods and people with proper protection. Not only that, this research will set a precedent for future research and development of tools focused on improving the security of modern communication protocols.

1.8. Thesis Structure

The thesis structure is in summary below:

Chapter 1: Identifying research problems, objectives, questions, and hypotheses in gRPC.

Chapter 2: Reviewing the current state and previous studies on gRPC evaluation and Cybersecurity Risk by conducting a literature review to gather information related to gRPC and other APIs for comparison and using the information for the next stage.

Chapter 3: Presenting the methodology, framework, and analysis of gRPC functionality to develop a relevant framework for use in testing, tools, and identifying vulnerabilities.

Chapter 4: Presenting implementation results derived from analysis and testing.

Chapter 5: Summarizing findings and providing recommendations from cybersecurity experts and top management, along with suggestions for future work.