F5 Quantum: Quantum Resistant Cloud Native Stack

Final Report

Sponsors:

f5, inc (original)



Velotix (new)

Icon

Description automatically generated

Team: f5 Quantum

A picture containing outdoor object, star

Description automatically generated

Team Members

* Emma Dickenson

Contents

1. Description 4
   1. Problem 4
   2. Goals and Outcomes 4
   3. Related Work 7
2. Team Members 7
3. Requirements 10
   1. Minimum Viable Product 10
   2. Epics and User Stories 10
   3. Epic 1: Epic Title **Error! Bookmark not defined.**
      1. User Story 1 **Error! Bookmark not defined.**
      2. User Story 2 **Error! Bookmark not defined.**
   4. Epic 2: Epic Title 11
   5. Assumptions 14
4. Design **Error! Bookmark not defined.**
   1. Software Architecture **Error! Bookmark not defined.**
      1. Subsystem 1 17
   2. Subsystem 2 **Error! Bookmark not defined.**
   3. Data 20
   4. User Interface 21
5. Execution Plan 25
   1. Milestones 25
   2. Workflow 25
   3. Risk Management 27
6. Evaluation 29
   1. Software Testing 29
      1. Approach 29
      2. Results 29
      3. Discussion 31
   2. User Testing (optional but strongly encouraged) 32
   3. Sponsor Feedback 32
7. Future Work 29
8. Reflection 35
   1. What Processes Worked? 35
   2. What Processes Did Not Work? 35
   3. How Did Processes Change Over Time 35
   4. How Will You Fine Tune Your Process for the Future? 36
9. References Cited 37

# Description

## Problem

## The software industry of today has ever increasing demands in regard to portability, speed of user access, security, and modularization. As a result, there is a large push towards cloud native applications. The remote hosting of both servers and the processes performed on them presents a new host of security challenges that were previously handled on premise for non-cloud native applications. As of June 2020, 48% of online businesses around the world use cloud computing to store their most important data and this number is only increasing [1]. Since the amount of important data that is handled by cloud computing is increasing so rapidly, cloud security is essential now more than ever.

## At the same time, the next paradigm in computation and encryption is on the horizon – quantum computing. While still in its infancy, quantum algorithms can theoretically solve problems that would take conventional computers a problematically long time. This poses an immediate and real threat to all forms of encryption and secure data storage.

## Quantum computing utilizes the principles of quantum mechanics as generalized rules of probability and constructs a model of computation around them. The system can then probabilistically generate solutions to specific problem types. Modern encryption relies on the security of large integer factorization, a known NP-complete problem. By solving an analogous problem suspected to be NP-complete, the hidden subgroup problem for finite Abelian groups, quantum computers can theoretically crack modern encryption. This leverages the Cook-Levin theorem, which postulates that all NP-complete problems can be reduced in polynomial time to the Boolean satisfiability problem.

## Although there are no quantum-computers that exist today, the technology is rapidly advancing and it is predicted that by 2030, there will be 2,000-5,000 quantum computers throughout the world [2]. Since the threat to cloud security from quantum computing is coming soon, it is essential that the service mesh for cloud native computing is upgraded to be secure against quantum threats. If this problem is not resolved, the nearly infinite amount of data stored through cloud computing will be easily accessible to quantum hackers. The stakeholders for this problem include both the companies that host cloud native applications who are responsible to keep their user data secure as well as users of any cloud application who do not want their data stolen. To counteract this problem, a framework must be designed and implemented to protect and restrict access to distributed systems. This framework involves aspects of the cloud service mesh including proxies such as Envoy. This is an incredibly important problem to solve, as the development of scaled quantum computers will jeopardize the security of all standard forms of encryption.

## Goals and Outcomes

*Table 1: Goals and Outcomes*

|  |  |
| --- | --- |
| **Goal** | **Desired Outcome** |
| 1. **Give seminar presentations on cloud computing and cloud native** | Team members pass quiz given by sponsor on cloud computing and cloud native with at least 90% accuracy |
| 1. **Gain understanding of the relevant cloud tech stacks and programming models** | Ability to explain key products and technologies (NGINX, Envoy, K8s, Docker, containerization, etc.) and present to others; the pros and cons of each; and the relevant security concerns and outlook |
| 1. **Gain understanding of quantum models of computation** | Ability to explain and give seminar presentation on quantum computation, the underlying mathematics, and its implications on cryptography and cybersecurity  Team members pass quiz given by sponsor on quantum computing with at least 90% accuracy |
| 1. **Develop a first prototype with NGINX forked with quantum resistant library** | Able to use Dockerfile shell script to install NGINX forked with quantum-resistant algorithms on Mac, Windows and Linux computers – NGINX file exists on computer  Run quantum-resistant NGINX proxy demo to deploy an application – terminal shows that the quantum-resistant algorithm Dilithium is being used and URL of application includes https to show that the proxy is being used |
| 1. **Develop a prototype of Envoy using quantum resistant security algorithms** | Able to use Dockerfile shell script to install Envoy forked with quantum-resistant algorithms on Mac, Windows and Linux computers – Envoy file exists on computer  Run quantum-resistant Envoy proxy demo to deploy an application – terminal shows that the quantum-resistant algorithm Dilithium3 is being used and URL of application includes https to show that the proxy is being used  Istio is able to deploy multiple applications using Envoy successfully – can see that multiple applications are deployed and that Envoy is running with the Dilithium3 algorithm in the terminal |
| 1. **Benchmarks and results analysis** | The throughput (requests/s) for NGINX used with standard RSA algorithms should be about 10-15% higher than the throughput for the NGINX using the quantum-resistant Dilithium3 algorithm  The throughput (requests/s) for Envoy used with standard RSA algorithms should be about 10-15% higher than the throughput for the NGINX using the quantum-resistant Dilithium3 algorithm |
| 1. **Gain interest from other cloud native groups to update other parts of the cloud native stack to be quantum resistant through conference presentations and creating a blueprint for further development** | At least fifty other people within the cloud native community are presented information about our project  At least 5 of the Github issues for other parts of the cloud native stack that need to be upgraded are commented on by people interested in developing them |

## Related Work

# The threat of quantum computing on software in the cloud native stack is heavily based on Shor’s Algorithm. Shor’s algorithm uses quantum computing to find non-trivial factors of composite integers in polynomial time with high probability [3]. This means that, with quantum computing, many current cryptographic protocols could be cracked as they rely on the difficulty of non-quantum computers for finding prime factors of a large number. For quantum computers, this difficulty is decreased exponentially [4].

# Although quantum computation is not yet advanced enough to determine prime numbers larger than 21, many companies and researchers have already developed quantum resistant algorithms. The company IBM has developed a solution to protect cloud native application data against quantum computing, also using a reverse proxy. They use HaProxy, whereas we will be using NGINX; they also use Kubernetes service [5]. Despite the existence of already developed quantum resistant services and algorithms, the technology is still so new that there is still room to find the most optimal solution.

# There are a few quantum-resistant cryptographic algorithms that researchers have determined to be the most powerful so far. May of these are algorithms that F5 is currently focusing on as well.

# One of the algorithms is called NTRUEncyrpt, operating on lattice-encryption protocols. This algorithm is efficient and leaves a low memory footprint, but there is patent protected. Another algorithm is the McEliece Encryption system, involving randomization during the encryption process rather than during the key generation process. It is faster than an RSA key, but consists of keys that are 256 times the size of the average RSA key. A third algorithm being researched the Ring Learning with Errors algorithm, which is used to solve problems in field/set theory. RLE consists of keys that are the same size as RSA keys, making them similarly difficult for classical models of computation to crack. So far, the most favored algorithm is the Supersingular Isogeny Diffie-Hellman Key Exchange algorithm. This algorithm has the smallest keys, supports forward secrecy, and is not patent protected. Although this algorithm is currently the most preferred, there have still been disturbing breaches with it. Researchers have also considered integrating a hybrid algorithm, where half of the key uses classical key exchange and the other half use quantum [6]. Clearly, there is still a lot of research to do in finding the best algorithm for post quantum resistant software, and the field is still in its early stages.

Although there has been development in quantum-resistant encryption, there has not been much focus on quantum-security within the realm of cloud computing; thus this is an area that needs immediate focus. The Cloud Native Computing Foundation (CNCF) is the largest open-source cloud native community. They have a task force for Cloud Native security and have begun to discuss the threats of quantum-computing, but they have not put any policies or recommendations in place yet.

# Our team will use the current body of research on these current algorithms to develop the best strategy for building a post quantum software stack on cloud native. In order to research and develop a prototype for post quantum resistant software in cloud native stack, our team will have to learn and utilize skills in cloud and cloud native computing. We will focus on traffic, compute, and storage within cloud computation. In addition, we will deepen our understanding of quantum computing in order to develop quantum resistant software in the cloud native stack, and we will apply knowledge of encryption to evaluate current algorithms. We will also hone in our analytical skills to evaluate our prototype and the effects on a company. Throughout the duration of the project, we will be working with cloud computing products; specifically, NGINX, Envoy, and Kubernetes (K8s), and containers. The NGINX web server, owned by F5, will be used as a reverse proxy [7]. Envoy and Kubernetes provide microservicing for distributed systems. Envoy, the high-performance C++ proxy, allows the abstraction of different functionalities to a single binary so then they can be used within all services. Envoy is deployed into production with Kubernetes, an open-source software that implements service mesh capabilities [8]. These various tools and technologies will allow us to do cloud native computing. Lastly, containers will be used as abstract executable units that include all necessary items to run an application [9].

# Team Members

*Biographies on following pages*

A person smiling for the camera

Description automatically generated with medium confidence**Emma Dickenson**

Seattle, WA

https://www.linkedin.com/in/emma-dickenson/

**Education**

**Washington State University**

Bachelor of Science, Computer Science Expected Graduation: May 2023

**Technical Skills**

* Languages: Python (5 years), Java (4 years), C# (1 year), JavaScript, HTML, CSS

**Experience**

**Qualtrics**  Seattle, WA

*Software Test Engineer Intern & Part-Time* (Python)

May 2021 - August 2022

* Used requirement analysis to develop test plans
* Wrote test automation scripts for new healthcare product to run weekly and report results to development team
* Developed models and tests for new integration testing framework in Python to test notification function of the product

**Coding with Kids** Seattle, WA

*Regional Team Lead, Manager, & Instructor* (Java)

May 2018 - Mar 2020

* Taught children K-8 Python, Java, and web development
* Communicated daily between managers, instructors, and customers to ensure overall satisfaction and make the Seattle region the fastest growing region of the company in the country

**Interests**

Traveling, swimming, snow sports, cooking

# Requirements

## Minimum Viable Product

Advancements in quantum computing are beginning to threaten the encryption algorithms that are currently in place. If quantum computing advances further, most current encryption algorithms are in danger of being cracked by quantum computers. There are quantum resistant algorithms that currently exist. Once quantum computers develop further, it is vital that these quantum resistant algorithms replace the current encryption algorithms for all parts of the cloud native stack. NGINX and Envoy are both proxies that are able to provide security for applications within cloud computing. We will deliver a Dockerfile that runs a shell script to install a quantum-resistant prototype of NGINX as well as a quantum-resistant prototype of Envoy. These proxies will have a throughput within 20% of the throughput of their regular counterparts. In addition, we will deliver materials to help other people in the cloud computing community make other parts of the cloud native stack quantum resistant. This materials include a conference presentation and a blueprint to show how the cloud native stack needs to be upgraded.

## Epics and User Stories

## Epic 1: Cloud Native & Quantum Computing Research

Learn the key tools and technology stacks for Cloud Native Development. Additionally, learn how quantum computing will be a threat to the security within cloud native development in the future.

### Cloud Computing Background

As a cloud developer, I want to have a clear understanding of what cloud computing is and its importance so that I will be able to learn about the various components within the cloud native stack.

* 75 minute Powerpoint presentation
* Thorough explanation of what cloud computing is, why cloud computing is popular, and how it is affecting businesses all over the world
  + Explain various Cloud models and their use cases
* Advisor agrees that the presentation covered the desired topics in enough detail
* Quiz given after to ensure teammates have largely absorbed the presented information
  + Teammates should score 90% or higher

**Story Points**: 13

**Priority Level**: 1

### Cloud Computing Tools

As a cloud developer, I want to have a clear understanding of the various tools and technologies of the cloud native stack that are needed to host applications, so that I can learn how to make the cloud native stack resistant to quantum attacks.

* 75 minute Powerpoint presentation
* Presentation is given on Cloud Computing tools and each part of the Cloud Native stack and how the parts interact with each other
* Advisor agrees that the presentation covered the desired topics in enough detail

**Story Points**: 13

**Priority Level**: 1

### Quantum Background Mathematics

As a cloud developer, I want to understand the mathematics underlying quantum theory, so I can effectively understand how quantum algorithms are a threat to the current security measures in place for the cloud native stack

* 75 minute Powerpoint presentation
* Presentation is on complex numbers, Bra-ket notation, vector spaces, Hilbert space, Hadamard gates, and Qubits
* Project mentor signs off on presentation
* Quiz given after to ensure teammates have largely absorbed the presented information
  + Teammates should score 90% or higher
* Deutsch Oracle algorithm is demonstrated using Python and Cirq, and open-source quantum framework

**Story Points**: 13

**Priority Level**: 1

### Quantum Encryption and Algorithms

As a cloud developer, I want to understand quantum encryption, so I can implement my own quantum resistant algorithms

* Presentation is given on current state of encryption, quantum decryption, Shor’s algorithm, OpenQuantumSafe, and post-quantum encryption options
* Project mentor signs off on presentation
* Math quiz given after to ensure teammates have largely absorbed the presented information
  + Teammates should score 90% or higher
* Teammates can implement Shor’s algorithm using Qiskit

**Story Points**: 13

**Priority Level**: 1

## Epic 2: Develop Quantum-Resistant Proxies

Apply Cloud Computing and Quantum Cryptography knowledge learned during Epic 1 and build a quantum-resistant NGINX prototype

### Design a Quantum-Resistant NGINX

As a capstone team member working on the NGINX, I want to have a clear design before we start coding so people do not make poor design choices and resort to ad-hoc development.

* Design document created showing the entire design blueprint with diagrams as well as a timeline for the Quantum-Resistant NGINX
* Teammates agree on which design to use
* Advisor signs off on design

**Story Points**: 13

**Priority Level**: 1

### Create Quantum-Resistant NGINX

As the owner of a web application, I want to be able to keep the data coming in and out of my website secure against a quantum attack while using a reverse proxy.

* Dockerfile to run shell script to install NGINX forked with OpenQuantumSafe (library with quantum-resistant algorithms) that can be ran on Mac, Windows, and Linux
* New version of NGINX uses quantum-resistant algorithms for security
* New version of NGINX has same functionality as the regular NGINX as a reverse proxy

**Story Points**: 13

**Priority Level**: 1

### Benchmark Quantum-Resistant NGINX

As the owner of a web application, I want the performance rate of my reverse proxy to be similar to the performance rate of regular NGINX

* Results document showing screenshots of quantum-resistant NGINX installation from part 3.4.2 on Windows, Mac, and Linux operating systems
* Document containing benchmark results for both regular NGINX and the quantum-resistant NGINX with results of the quantum-resistant NGINX being no less than 20% lower throughput than for the regular NGINX
* Dockerfile to run shell script that installs tools for benchmarking both the original and quantum-resistant versions of NGINX
* Test script files to be able to replicate the benchmarking

**Story Points**: 13

**Priority Level**: 2

### Deploy Application with Quantum-Resistant NGINX

As the owner of a web application, I want to be able to deploy my application using an NGINX reverse proxy that uses quantum-resistant security

* Dockerfile script that runs demo of an application being deployed using the new quantum-resistant NGINX
* Terminal shows the NGINX is using a quantum-resistant algorithm for security
* URL of web application shows that the application is being secured securely with https

**Story Points**: 8

**Priority Level**: 1

### Design a Quantum-Resistant Envoy

As a capstone team member working on the NGINX, I want to have a clear design before we start coding so people do not make poor design choices and resort to ad-hoc development.

* Design document created showing the entire design blueprint with diagrams as well as a timeline for the Quantum-Resistant NGINX
* Teammates agree on which design to use
* Advisor signs off on design

**Story Points**: 8

**Priority Level**: 1

### Create Quantum-Resistant Envoy

As the owner of a web application, I want to be able to keep the traffic in and out of my website secure against a quantum attack while using a proxy.

* Dockerfile to run shell script to install Envoy forked with OpenQuantumSafe (library with quantum-resistant algorithms) that can be ran on Mac, Windows, and Linux
* New version of Envoy uses quantum-resistant algorithms for security
* New version of Envoy has same functionality as the regular NGINX as a reverse proxy
* New version of Envoy is compatible with Istio

**Story Points**: 13

**Priority Level**: 1

### Benchmark Quantum-Resistant Envoy

* Results document showing screenshots of quantum-resistant Envoy installation from part 3.4.2 on Windows, Mac, and Linux operating systems
* Document containing benchmark results for both regular Envoy and the quantum-resistant NGINX with results of the quantum-resistant Envoy being no less than 20% lower throughput than for the regular Envoy
* Dockerfile to run shell script that installs tools for benchmarking both the original and quantum-resistant versions of Envoy
* Test script files to be able to replicate the benchmarking

**Story Points**: 13

**Priority Level**: 2

### Deploy Application with Envoy

As the owner of a web application, I want to be able to deploy my application using an Envoy proxy that uses quantum-resistant security

* Dockerfile script that runs demo of an application being deployed using the new quantum-resistant Envoy
* Terminal shows the Envoy is using a quantum-resistant algorithm for security
* URL of web application shows that the application is being secured securely with https

**Story Points**: 8

**Priority Level**: 1

### Deploy Application with Envoy with Istio

As the owner of a cloud web application, I want to be able to deploy multiple microservices with Istio and Envoy

* Kubernetes cluster runs (only possible if Istio and Envoy are working together)
* Terminal shows that Envoy is using a quantum-resistant algorithm

**Story Points**: 13

**Priority Level**: 2

## Epic 3: Help others make Cloud Native Stack Quantum Resistant

Spread the importance of making the cloud native stack resistant to quantum attacks and help guide developers to creating other quantum-resistant parts of the cloud native stack.

### Presentation on Threat of Quantum Computers to Cloud Security

As the leader of a company that hosts a website using cloud computing, I want to know the threats that quantum computing has on the security of my user’s data.

* 30 minute presentation addressing the following topics
  + How to get involved with making the cloud native stack quantum-resistant
  + Work we have done so far to address the problem
  + Threat of quantum-computing to the cloud native community
* Easy-to-reproduce demo of how to benchmark NGINX
* Easy-to-reproduce demo of how to run the quantum-resistant Envoy with Istio
* Able to answer questions about the project

**Story Points**: 13

**Priority Level**: 2

### Blueprint of Cloud Native Stack that needs to be upgraded for Quantum Securityv

As a developer working on the cloud stack, I want to easily see which parts of the cloud native stack need to be updated to be quantum-resistant.

* Document that shows each part of the cloud native stack for a small application that needs to be upgraded to be quantum-resistant, including:
  + Overall diagram of the cloud native stack for a small application
  + Individual diagrams of components of each part of the cloud native stack that needs to be upgraded with explanations of what needs to be upgraded

**Story Points**: 13

**Priority Level**: 2

### Update Repository for Open-Source Use

As a developer working on the cloud stack, I want to easily be able to update the current cloud native stack to be quantum-resistant.

* Github repository is formatted so that each component we have worked on (NGINX, Envoy, and Istio) has its own directory
* All information in the repository is up to date
* All information in the repository is easy to follow

**Story Points**: 3

**Priority Level**: 3

### Github Issues for Further Development

As a developer working on the cloud stack, I want to easily be able to see how to update the current cloud native stack to be quantum-resistant.

* A detailed issue for how to make each component that needs to be upgraded to become quantum resistant exists in the Github repository
* Issues are easy to follow for cloud developers
* Issues include how to test that the component works and is quantum-resistant

**Story Points**: 13

**Priority Level**: 3

## Assumptions

We assume NGINX is compatible with OpenQuantumSafe, a library that secures communications between application services and allows authentication between databases or platforms without passwords or API keys.

We assume the algorithms in OpenQuantumSafe will be quantum resistant and that we will be building on top of OpenQuantumSafe, instead of writing our own quantum resistant algorithms from the ground up.

We assume that Envoy is compatible with OpenQuantumSafe.

We assume that the versions of NGINX and Envoy built from source and forked with OpenQuantumSafe will be resistant to quantum attacks.

We assume that Istio will be compatible with the Envoy that is forked with the OpenQuantumSafe library.

We assume that the quantum-resistant versions of NGINX and Envoy will not have a throughput that is greater than 20% less than their regular counterparts.

We assume that we will be accepted into a conference to present on the topic of quantum security within cloud computing.

# Design

Our project is designed following the design of the typical architecture needed to deploy a small application, meaning we focus on the typical components for a small native stack and how they interact with each other. Our design involves changing the original cloud native stack to be quantum-resistant. We do not change the architecture of the cloud native stack, but rather change individual components to be quantum-resistant.

## Software Architecture

Since our project aims to create a completely quantum-resistant cloud native stack used to deploy small sized applications, our project follows the same architecture as the proposed cloud native stack architecture by the Cloud Native Computing Foundation. This architecture was created from years of research by many of the greatest developers. For our development, we focused on developing just one part of the architecture, the Envoy proxy (next to the red arrow), but we want other people who help develop this soon-to-be open-source project, to develop other parts of the architecture to be quantum resistant.

Although all layers in the architecture are connected, there is low coupling between the different aspects of the cloud native stack as this makes it easier to change one part; in fact each layer is owned by a completely different company, although they all work together cohesively. The low coupling is especially helpful when it comes to making each part quantum-resistant as each part can be taken on as a separate task by different people at different times and it will not affect the rest of the cloud native stack.

*Diagram 1: Cloud Native Architecture*

Diagram

Description automatically generated

Here are some of the main components of the cloud native stack and their use in the overall architecture:

Note: The purpose of cloud native architecture is so that many micro applications can be deployed and communicate with one another to act as a single web application without many of the coupling issues that hosting all the services on a single web service would have. The components within the cloud native architecture communicate with one another to allow microservices to work together to act as a single web application when it reality there are multiple applications acting together.

**Istio:** Manages and configures the envoy proxies to route traffic

**Envoy:** A proxy that mediates and controls traffic between microservices, including ensuring that the traffic data is secure

**Custom Resource Definition (CRD):** Sends Istio configuration YAML files to Istio

**Gateway:** Hardware or software-based appliance that the user has on-premise that works as a bridge between local applications and cloud applications

### 

### Subsystem 1: Quantum-Resistant NGINX Reverse Proxy

**Description**

The purpose of NGINX in a general sense is to manage the data traffic that goes into an out of a web application. The purpose of creating an NGINX in terms of the overall architecture of our project was to test and see if it was possible to create a proxy that implemented quantum-resistant algorithms rather than the standard algorithms that are used for security in the regular versions. As seen in the overall “Cloud Native Architecture” diagram for our project, NGINX is not included in the overall architecture of our diagram; this is because in typical cloud native application deployment, Envoy is used as the proxy rather than NGINX. NGINX is easier to manipulate and change, so it made sense to create and test a prototype of NGINX first that used quantum-resistant algorithms before attempting to create a quantum-resistant version of Envoy.

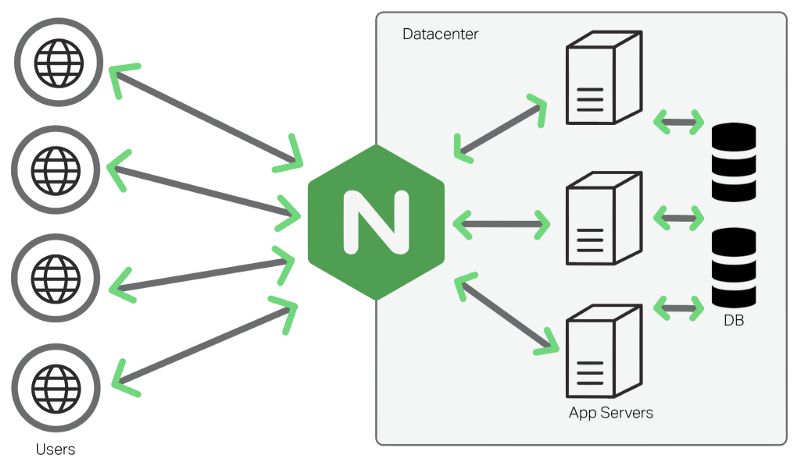
**Design Considerations**

The general installation of NGINX will remain the same so that it has the same functionality as the regular NGINX, but the Certificate Authority will be forked with the OpenQuantumSafe library so that the algorithms used for encryption are quantum-resistant. This means that NGINX will still handle the traffic coming in and out of web applications, but the security provided will be quantum-resistant. The trade off with incorporating the OpenQuantumSafe library is that the algorithms have a slightly lower throughput than the standard RSA algorithms that are used originally, but the slight difference in throughput is insignificant compared to the increase in security that the quantum-resistant algorithms will provide.

**Interface Description**

The diagram below shows how the quantum-resistant NGINX we have created interacts as a buffer between the users of web applications and the application servers themselves.

*Diagram 2: NGINX Proxy with Applications Architecture*

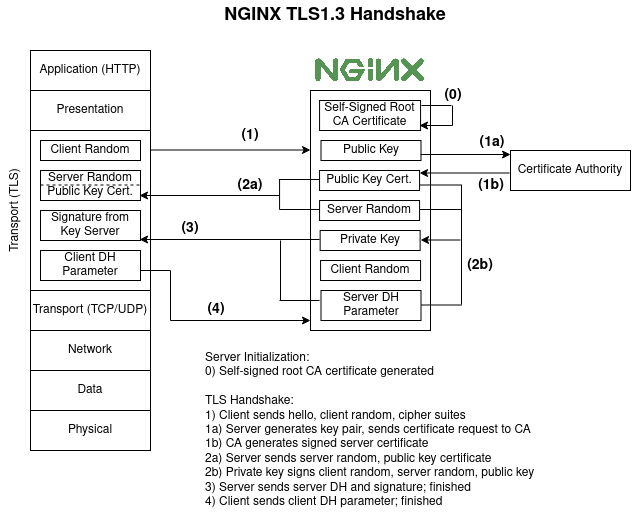


Source: https://www.freecodecamp.org/news/an-introduction-to-nginx-for-developers-62179b6a458f/

The arrows pointing towards and away from NGINX represent the transport security layer in which NGINX controls the traffic and the security of the traffic. The new quantum-resistant NGINX we created can make that security quantum-resistant as the new NGINX contains quantum-resistant security algorithms.

The next diagram shows how we changed the architecture of NGINX itself to be quantum-resistant.

*Diagram 3: NGINX Architecture*



OpenQuantumSafe

We change the Certificate Authority to be forked with the OpenQuantumSafe library so that the certificates use quantum-resistant encryption to authenticate the public private keys. One the keys are authenticated, data is authorized to travel through the transport security layer where it can then reach the users or web applications.

### Subsystem 2: Quantum-Resistant Envoy Proxy

**Description**

The purpose of the quantum-resistant Envoy proxy in terms of the overall cloud native architecture is to allow communication between the various microservices applications within an entire application. Envoy controls the data traffic between the different microservices by communicating to the other Envoys in the other microservices. Istio, a control plane, tells the Envoys how to communicate with each other. By forking Envoy with a library for quantum-resistant algorithms, we change the security of the data traffic traveling from microservice to microservice to be secure against quantum attacks.

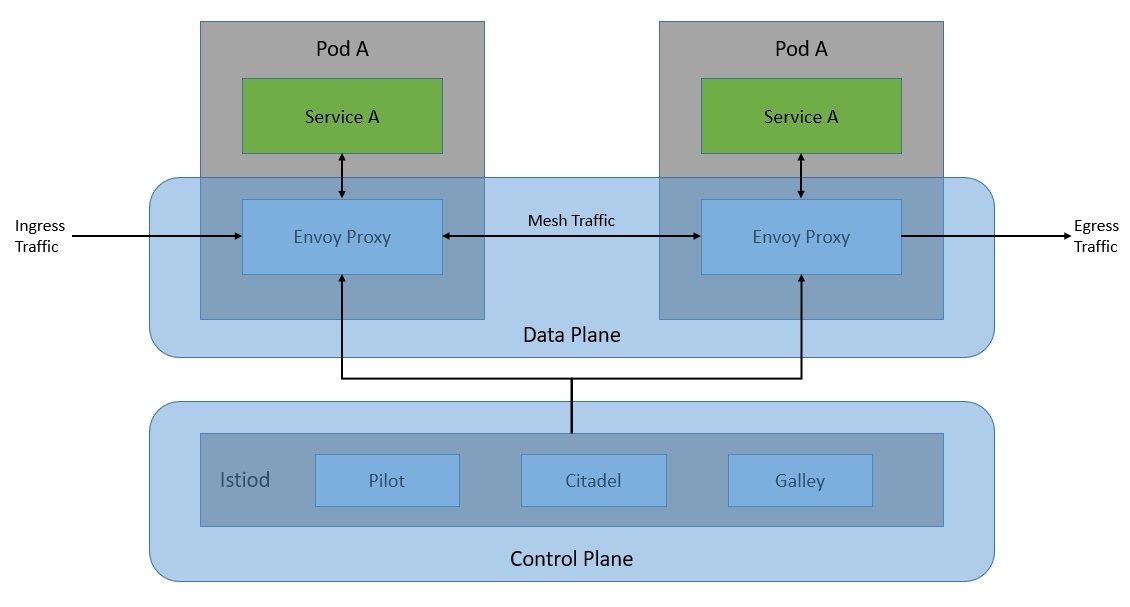
**Design Considerations**

The general installation of Envoy will remain the same so that it has the same functionality as the regular Envoy, but the Envoyy will be forked with the OpenQuantumSafe library so that the algorithms used for encryption are quantum-resistant. This means that Envoy will still handle the traffic between microservices, but the security provided will be quantum-resistant. The trade off with incorporating the OpenQuantumSafe library is that the algorithms have a slightly lower throughput than the standard RSA algorithms that are used originally, but the slight difference in throughput is insignificant compared to the increase in security that the quantum-resistant algorithms will provide.

**Interface Description**

The diagram below shows the relationship between the Envoy with the quantum-resistant algorithms and the rest of the cloud native system for deploying a small application.

*Diagram 4: Envoy with Microservices Architecture*

~~~~

OpenQuantumSafe

OpenQuantumSafe

Source: https://www.baeldung.com/ops/istio-service-mesh

As shown in the diagram above, Envoy uses the OpenQuantumSafe library above to control the mesh traffic from pod (microservice) to pod.

## Data

Our project builds on proxies that support applications and the transfer of data rather than storing data itself. We did not use any major data structures in our project but rather created Dockerfiles to save the configurations of the installations necessary to build NGINX and Envoy from source. In addition, our project will include applications that we build but these applications will not store any major data as the primary purpose of the applications is to show that an application can run on the NGINX and Envoy that we built.

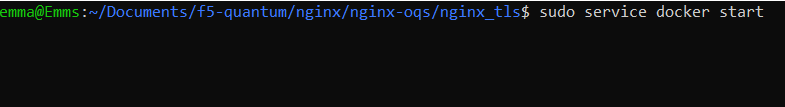
## User Interface

The main technical focus of this proect is on the underlying technologies behind hosting clous applications. This means that the functionalities of the project remain unseen to the end users (the hosts of cloud applications and the users of cloud applications). There is no perceivable difference in accessing web server using pre-quantum and post-quantum encryption from the user’s perspective so a traditional UI demo is sub-optimal for this. Although there is no traditional UI application for project, users must interact with the command line to install our products as well as run our benchmarking and demos. Below there will be step by step instructions of how users will run each main part of our project.

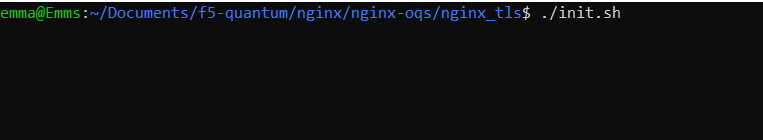
**NGINX Installation**

The NGINX installation much be ran in a cloned version of the NGINX directory within our Github repository.

Step1: Launch Docker Daemon



Step 2: Run init.sh shell script which starts the Docker file that runs through the quantum-resistant NGINX installation



**NGINX Demo**

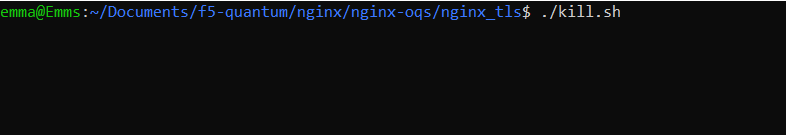
Make sure that you have ran steps 1 and 2 from the NGINX installation in the same terminal session. If NGINX is already installed, step 2 will not reinstall NGINX.

Step 1: Run the query.sh script to open up the transport layer security that is secured with the quantum-resistant NGINX

Text

Description automatically generated

Step 2: Run the kill.sh script when finished to stop NGINX from running



**NGINX Benchmarking**

The NGINX benchmarking should be done in the benchmarking directory within the NGINX repository. For benchmarking the regular NGINX, go into the directory titled “nginx\_vanilla”. For benchmarking the quantum-resistant NGINX, go into the directory titled “nginx\_oqs”.

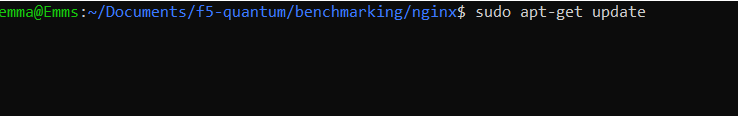
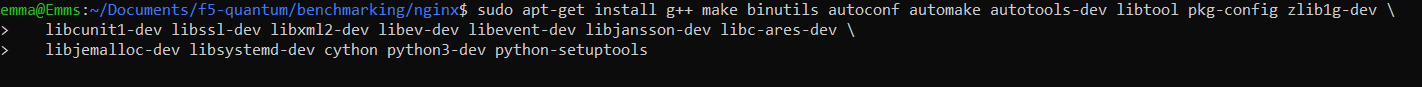


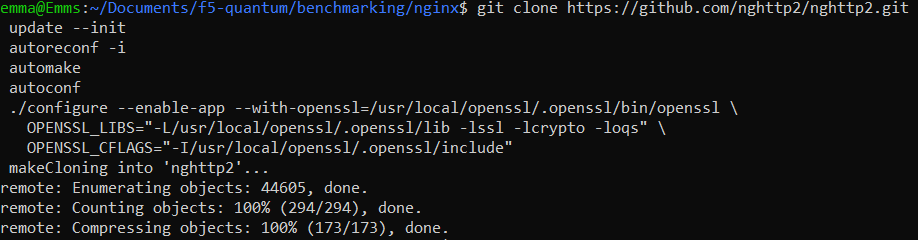
Step 1: Launch Docker Daemon

Text

Description automatically generated

Step 2: Install the h2load tool by running the following commands (if not done so already)





Step 3: Run init.sh script to start up the benchmarking

**Text

Description automatically generated**

Step 4: Run test\_script to run the benchmarking

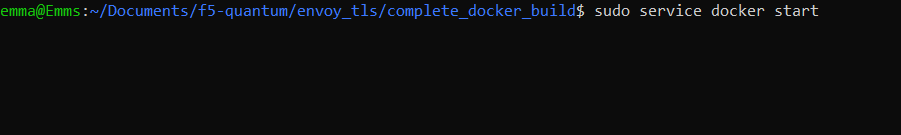
Text

Description automatically generated

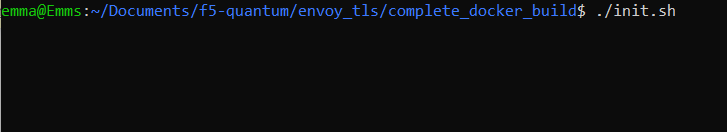
**Envoy Installation**

The Envoy installation much be ran in a cloned version of the NGINX directory within our Github repository.

Step1: Launch Docker Daemon



Step 2: Run init.sh shell script which starts the Docker file that runs through the quantum-resistant Envoy installation



**Envoy Demo**

Make sure that you have ran steps 1 and 2 from the Envoy installation in the same terminal session. If Envoy is already installed, step 2 will not reinstall Envoy.

Step 1: Run the query.sh script to launch an application using Envoy

Text

Description automatically generated

Step 2: Run kill.sh script to stop Envoy when you are finished

Text

Description automatically generated

**Envoy Benchmarking**

Follow the same steps as the NGINX benchmarking but make sure you are in the Envoy directory.

**Istio Demo with Envoy**

Follow the same steps as the Envoy demo but make sure you are in the Istio Directory.

# Execution Plan

# Milestones

*Table 2: Milestones*

**Graphical user interface, table

Description automatically generated**

*Chart 1: Milestone Timeline View*

**Timeline

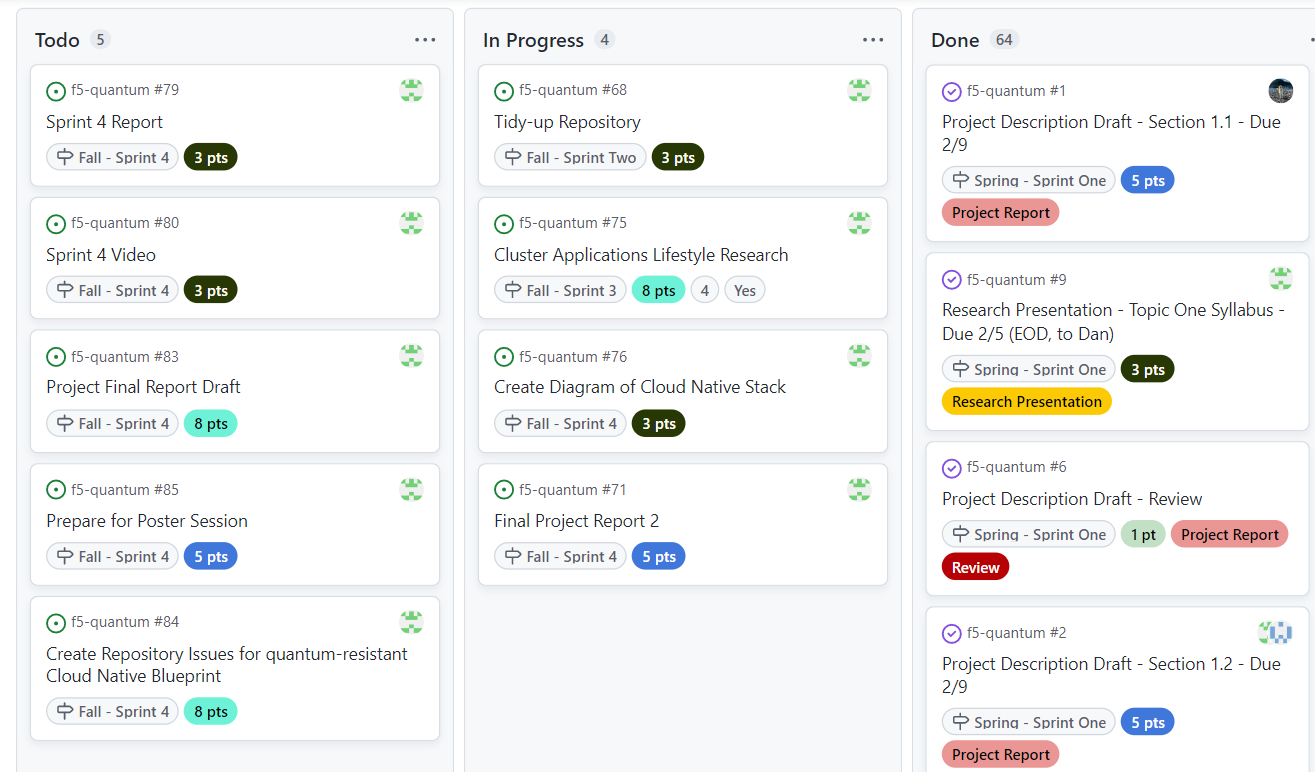
Description automatically generated**Timeline

Description automatically generated

# Workflow

We managed our project using the Kanban board feature it Github. Within that feature, all our issues for our project are automatically placed within the Kanban board. These issues include point hours and who owns the issue. There are four categories within the Kanban board: “No Status”, “To Do”, “In-Progress”, and “Done”. Immediately when a new issue is added, it is placed into the “No Status” category and we drag it into the “To Do” category right away. Once the task has been started, we drag it into the In-Progress category and once its finished it gets dragged into the “Done” category. During the project, we mostly focus on the “In-Progress” section and the “To-do” sections to decide what to work on at a given time.

Kanban board can be found here: https://github.com/orgs/wsu-cpts421-sp22/projects/2/views/1

*Chart 2: Workflow on Kanban*

# Risk Management

*Table 3: Risk Management*

|  |  |
| --- | --- |
| **Project Risks** | **Mitigations** |
| Project report does not reflect the project the sponsor wants | * Weekly check-ins with sponsor to discuss progress on project * Show sponsor drafts of project report before proceeding |
| The information delivered in the Cloud Computing and Quantum presentations is not accurate | * Fact check the information with multiple credible sources |
| NGINX with Openssl Build is not possible | * Create good documentation about what has been tried and why it does not work * Focus on making other components of the cloud native stack quantum-resistant   + Start with Envoy with OQS build |
| Envoy-OQS Server Docker Build is not possible | * Create good documentation about what has been tried and why it does not work * Focus on making other components of the cloud native stack quantum-resistant |
| Istio is not compatible with the quantum-resistant Envoy | * Create good documentation about what has been tried and why it does not work * Research what makes the two not compatible and see if you can make modifications to Istio or Envoy to make them compatible |
| Istio Demo | * The chosen demo does not work with the Istio build   + If completely blocked – try a different demo |
| NGINX and Envoy Benchmarking | * Benchmarking does not accurately show NGINX and Envoy results   + Compare results with vanilla counterparts and make sure that they are similar (there should be around a 10% difference based on literature values of the algorithms) |
| Presentation for Kubecon Conference gets deleted/does not work on presenting computer | * Have presentation setup on backup computer * Bring notes to explain the presentation even if you cannot show actual presentation |

# Evaluation

## Software Testing

### Approach

We did a small amount of unit testing by running the installation instructions outside of the docker container one at a time and checking that the software that we installed existed and was the correct version by running the “–version” tag. This was mostly done during debugging.

In terms of integration testing, we ran the dockerfiles we created on a clean installation of Ubuntu to ensure that the installations work correctly. We verified our results by attempting the demos that only work if the correct versions of NGINX and Envoy that have the quantum resistant algorithms are installed.

For the bulk of our integration testing, did benchmarking to ensure that the NGINX and Envoy with quantum-resistant algorithms that we built have comparable throughput to NGINX and Envoy with standard RSA encryption. To do the benchmarking, we installed a benchmarking tool called H2load. We then had to configure this tool to work with the libraries we used for the NGINX and Envoy we built. Once we had h2load installed, we could run various commands with different parameters. Some parameters we changed were the number of concurrent client connections as well as the number of threads used. We used these parameters to test the throughput for deployments of NGINX and Envoy both with and without the quantum-resistant algorithms and compared results from NGINX and Envoy with the RSA algorithms to the NGINX and Envoy with quantum-resistant algorithms.

### Results

We include a large excel file with all the individual results in the Benchmarking branch of our Github. In this section, I will summarize the results with a few tables and graphs.

The throughput for the NGINX that we built with the Quantum Resistant algorithms should be within a 15% range of the throughput for the regular NGINX for the tests to be considered passing. This is because based on the research we have studied, the Dilithium encryption algorithm’s throughput should be less than 15% lower than the throughput for RSA encryption. The same is true for Envoy.

*Table 4: Benchmarking Results for NGINX using 1 Thread*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Client Connections** | **NGINX using RSA** | **NGINX using Dilithium (Quantum-Resistant)** | **Difference** | **Pass/Fail** |
| 1 | 9925.00 | 8773.00 | 11.6% | Pass |
| 5 | 3341.00 | 2854.00 | 14.6% | Pass |
| 10 | 1675.20 | 1552.80 | 7.3% | Pass |
| 25 | 637.60 | 597.60 | 6.3% | Pass |
| 50 | 291.40 | 271.40 | 6.7% | Pass |

*Graph 1: NGINX Benchmark Results 1 Client Thread*

*Table 5: Benchmarking Results for Envoy using 1 Thread*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Client Connections** | **Envoy using RSA** | **Envoy using Dilithium (Quantum-Resistant)** | **Difference** | **Pass/Fail** |
| 1 | 481.80 | 408.30 | 15.2% | Fail |
| 5 | 456.40 | 350.50 | 23.2% | Fail |
| 10 | 325.60 | 236.90 | 27.2% | Fail |
| 25 | 154.40 | 110.30 | 28.6% | Fail |
| 50 | 83.40 | 56.00 | 32.9% | Fail |

*Graph 2: NGINX Benchmark Results 1 Client Thread*

### Discussion

Benchmarking

At first, our software was not adequately tested as we noticed that we were getting a lot of outliers in our data. We realized that NGINX and Envoy oftentimes are slower when they start up. We made our tests more adequate by creating test scripts that took an average of ten rounds of running the tests to help combat outliers. We replaced our earlier results with these more accurate results.

We compared our benchmarking results to literature values produced on the throughput (number of requests per second) of RSA encryption algorithm security compared to the Dilithium3 encryption algorithm security and our results for the throughput of using these encryption algorithms with NGINX and Envoy match to those literature values. The literature values state that the throughput for Dilithium3 should be 8-13% less than the RSA algorithm; these are the results that we obtained as well.

In terms of further testing, as we add more proxies to the service mesh, we will need to benchmark those proxies as well. There are still more parameters that we can test within just NGINX and Envoy, although these are not a main priority. More importantly, we want to benchmark NGINX and Envoy with other quantum-resistant algorithms such as Spinx.

Integration Testing

There are improvements that need to be made to our integration testing. The installations were only tested once on a clean installation of Ubuntu on my computer. I ran into bugs and confusion in the installation instructions during the process and we fixed the bugs and confusing instructions until the installation worked on my computer, but never did another clean installation after that. Although we were satisfied that the installation did end up working on my computer, we cannot consider the testing adequate until we test with other users until those users run into no problems on a clean installation. To do this, we plan to do user testing in the future.

## User Testing

Currently, no user testing has been completed for our project, but we do have a plan for user testing in the future.

We plan to have a few test subjects read through our Github instructions and run the Dockerfiles to install both the NGINX with the quantum resistant algorithms as well as the Envoy with the Quantum resistant algorithms. After the users have the products installed, we will have them run the demos and then do benchmarking. We will observe the difficulty that it is for the users to follow the instructions. In addition, our product will be deemed successful if the following apply:

* Test subjects consider instructions to be “easy to follow”
* NGINX with OQS-fork and Envoy with OQS-fork are successfully installed
* Correct output for demos
* Benchmarking results within a 10% range of the results we obtained

## Sponsor Feedback

The latest prototype has been shared with our sponsor during a zoom meeting. During this meeting, we practiced presenting our presentation for the Kubecon conference we are attending. During the presentation, we ran all of the demos that show that are product is working and can be deployed. I specifically shared the demo of the benchmarking that I will be presenting at the conference. My sponsor’s feedback was mostly in terms of the presentation aspect of our project.

Overall, my sponsor said that the presentation sounded good and clean and had been improved a lot from the first iteration. In addition, he said that the timing of my part of the presentation was good in terms of the presentations’ time constraints. My sponsor also gave me the following feedback in terms of improvement:

* Find a way to show in the terminal that the NGINX version for the benchmarking demo you run is using the quantum resistant algorithms
* If your results that you obtain during the demo do not line up with the previous results, explain that you need an average of 10 runs
* When you walk through the NGINX demo, when it is getting the NGINX results you start talking fast (because the benchmarking is running faster than you are talking) and it gets confusing – either find a way to make the run take longer or find a different way to talk about what you are doing
* Explain the ciphers that show up in the terminal
* Instead of phrasing the results as “speed” talk about them in terms of “throughput”
* Add in two slides, one with a diagram of the NGINX benchmarking architecture and one with a diagram of the Envoy benchmarking architecture

# Future Work

***There are three directions that the future of this project can take:***

**1. Promote quantum-resistant cloud service mesh project**

There are many more parts of the cloud service mesh that need to be made quantum-resistant and to make a completely quantum-resistant cloud service mesh would take years and lots of resources for one small group to do on their own. The goal of our project was to provide resources for other companies within the cloud service mesh to develop quantum-resistant security for their part of the mesh. In order to help promote the project to other companies, you could start by following similar steps to what our project did:

* + - 1. Research background on the threat of quantum-computing to security and cloud computing. You could start by looking at some of the resources our team has created:
* <https://github.com/wsu-cpts421-sp22/f5-quantum/tree/admin_documents/presentations>
* <https://www.youtube.com/watch?v=YUUAs0QZ1ZU&t=784s>
  + - 1. Record Youtube presentation explaining the importance of quantum-resistant security for cloud services
      2. Choose one part of the cloud service mesh from the issues our team has created and develop it to be quantum resistant

***Issues can be found****:* [*https://github.com/Post-Quantum-Mesh/post-quantum-cloud-service-mesh/issues*](https://github.com/Post-Quantum-Mesh/post-quantum-cloud-service-mesh/issues)

* + - 1. Create demo videos explaining how the quantum-resistant part you created works
      2. Apply to attend conferences where you can network and present project to other companies within the cloud service mesh
      3. Create presentation documents like blueprints, blog posts, and presentations to send to other companies within the cloud service mesh to convince them to work on making their parts quantum-resistant
      4. Provide help to those companies you have reached out to if necessary
      5. Continue to develop more quantum-resistant parts of the service mesh that you can demo

**2. Make completely quantum-secure cloud service mesh**

For our project, our team focused on creating just one aspect of the cloud service mesh: proxies. As mentioned earlier, there are many more parts of the cloud service mesh for a small application that need to be updated to be quantum resistant.

Here is a diagram of those parts:

*\*Will add diagram after Sprint 4 is complete as it is a part of the sprint*

For this route for future development of the project, first choose one issue from the Github issues we created mapping the parts of the cloud service mesh for a small application that need to be updated, found here: [*https://github.com/Post-Quantum-Mesh/post-quantum-cloud-service-mesh/issues*](https://github.com/Post-Quantum-Mesh/post-quantum-cloud-service-mesh/issues).

Start by researching this part of the cloud service mesh and the installation instructions for the regular version of this part of the service mesh. The main part of this project will be to develop Dockerfiles that run shell scripts to install a quantum-resistant version of the part of the cloud service mesh you have chosen. These Dockerfiles should be written in Golang. This will involve researching how that part can be quantum resistant and what libraries are needed to make it so. Here is an example of the Dockerfile we created to make a quantum-resistant version of NGINX: <https://github.com/Post-Quantum-Mesh/nginx-oqs/blob/main/nginx_tls/Dockerfile>.

Once completed with the first issue, begin to choose new issues of different parts of the cloud service mesh to develop to be quantum-resistant. Test and make sure that all the newly developed parts of the cloud service mesh work together effectively.

**3. Develop blueprint for more complex cloud service mesh**

Our team has developed a blueprint to help companies that provide parts of the cloud service mesh make their parts resistant to quantum attacks. We only developed the blueprint for a cloud service mesh that could host a small-scale application. There are many more complex parts of the cloud-service mesh for hosting larger applications.

Expand on the blueprint that we already have to include parts of the service mesh needed to host massive applications. Similar to what we did to create a cloud blueprint for small applications, create diagrams to show how a cloud service mesh would look for larger applications. Example diagrams that we created can be found here: <https://github.com/Post-Quantum-Mesh/post-quantum-cloud-service-mesh/tree/blueprint>. A good start to find which other parts of the cloud service mesh that should be added is to look at the Cloud Native Computing Foundation website: https://www.cncf.io/.

Create Github issues, like the ones we created for the smaller post-quantum cloud service mesh, for people working on the post-quantum service mesh to follow. Create a detailed issue for each additional part of the service mesh that needs to be made quantum-resistant.

Once finished creating the more detailed blueprint, choose some of the issues to develop and make those parts of the cloud service mesh quantum-resistant.

# Reflection

## What Processes Worked?

Most of our processes for this project worked very well. Generally, we planned to meet twice a week with our sponsor but would be flexible and cancel meetings if they did not seem necessary. Our flexible plan to meet twice a week held us accountable to finish our tasks by the meeting dates, and also allowed us to get constant feedback from our sponsor. In addition, the flexibility was helpful because we could adjust the amount of times that we met a week based on necessity. For example, when we were preparing for our presentation, it was vital that we met multiple times a week, but when we were coding, if we did not have any questions that required screensharing, it was not always necessary to meet and our time was spent better working individually. Another process that worked well was our online communication. Rather than waiting for meetings to ask questions to each other and our sponsor, we would message each other on Discord when necessary. Usually when we were blocked, we would work together with our teammates first and only reach out to our sponsor if we were unable to resolve the problem working together. Lastly, our process of splitting up tasks to work on individually worked well as we were able to trust each other to get our individual tasks accomplished.

## What Processes Did Not Work?

We ran into a few processes that did not work. For example, having our twice a week meeting dates on Tuesday and Thursday was problematic because there was a five day gap in between the Thursday and Tuesday meeting but only two days between Tuesday and Thursday. This meant that sometimes barely any work was done before the Thursday meeting whereas sometimes the Thursday to Tuesday gap would be too long if a team member was blocked on a crucial part of the project. Another process that did not work was communicating over Discord when we had a time sensitive manner. Because of the time difference between our sponsor in Isreal, and our group in Pullman, it would often take hours or days for us to respond or get responses from our sponsor. For time sensitive communication, a quick zoom call where we could quickly communicate back and forth worked much better.

## How Did Processes Change Over Time

The way that work was divided among group members changed over time as we learned more about each other’s working styles and gained trust in each other’s ability to accomplish tasks. At the beginning of the project, most of the work that we did included pair programming and multiple people working on the same tasks. While working together, we began to prove to each other that we would accomplish the tasks that we said we would and we would accomplish those tasks to a high standard. Rather than continuing to work on the same tasks together, we divided up tasks based on our individual strengths and would keep each other updated on what we were working on and would ask questions to each other when blocked. In addition, our meetings with our sponsors started to become more and more productive as our project developed over time. At first, our meetings were generally much longer and involved a lot of explaining about the project from both us and our sponsor. Towards the end of our project, all of our meetings were working meetings where we would work on things, such as our presentation, where we needed to be working hands-on with our sponsor. Most other questions, would be communicated through Discord or email.

## How Will You Fine Tune Your Process for the Future?

In the future, we will fine tune our process by changing our twice-a-week meetings to be more spread out. For example, having meetings on Mondays and Thursday would leave more time in between meetings than Tuesday and Thursday. In addition, we will include a weekly message from each member of the team on Mondays (if there is no meeting Monday) to update each other on our progress as well as mention any time conflicts that we may have for the upcoming weeks.

# References Cited

[1] J. Flynn, “25 Amazing Cloud Adoption Statistics,” *Zippia*, 23-May-2022. [Online]. Available: https://www.zippia.com/advice/cloud-adoption-statistics/#:~:text=48%25%20of%20businesses%20choose%20to,store%20their%20more%20crucial%20data. [Accessed: 30-Nov-2022].

[2] D. M. Turner, “When will Quantum Computing Arrive and how will it Impact Cybersecurity?,” Cryptomathic. [Online]. Available: https://www.cryptomathic.com/news-events/blog/when-will-quantum-computing-arrive-and-how-will-it-impact-cybersecurity#:~:text=It%20is%20estimated%20that%20there%20could%20be%202%2C000,the%20numerous%20pieces%20of%20hardware%20and%20software%20required. [Accessed: 21-Jan-2022].

[3] U. Skosana, "Demonstration of Shor’s factoring algorithm for N=21 on IBM quantum

processors," November 2021. [Online]. Available: https://www.nature.com/articles/s41598-

021-95973-w.pdf.

[4] B. Lanyon, T. Weinhold and N. Langford, "Experimental Demonstration of a Compiled

Version of Shor’s Algorithm," December 2007. [Online]. Available:

https://ntserver1.wsulibs.wsu.edu:2981/prl/pdf/10.1103/PhysRevLett.99.250505.

[5] D. N. Ngaratnam, "Protecting Apps on IBM Cloud with Quantum-Safe Cryptography," 3 12

2020. [Online]. Available: https://www.ibm.com/cloud/blog/protecting-apps-on-ibm-

Cloud-with-quantum-safe-cryptography.

[6] D. Holmes, "How Quantum Computing Will Change Browser Encryption," 18 8 2020.

[Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/how-

quantum-computing-will-change-browser-encryption.

[7] NGINX, "What is NGINX?," [Online]. Available: https://www.nginx.com/.

[8] C. Posta, "Microservicing with Envoy, Istio and Kubernetes," 1 5 2018. [Online]. Available:

https://thenewstack.io/microservicing-with-envoy-istio-and-kubernetes/.

[9] Aqua, "Containers in Cloud Computing: Enabling Portability, Agility and Automation,"

[Online]. Available: https://www.aquasec.com/cloud-native-

academy/docker-container/container-cloud-computing/