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 4
2. Team Members 5
3. Requirements 5
   1. Minimum Viable Product 5
   2. Epics and User Stories 5
   3. Epic 1: Epic Title 5
      1. User Story 1 5
      2. User Story 2 5
   4. Epic 2: Epic Title 5
   5. Assumptions 6
4. Design 6
   1. Software Architecture 6
      1. Subsystem 1 6
   2. Subsystem 2 7
   3. Data 7
   4. User Interface 7
5. Execution Plan 7
   1. Milestones 7
   2. Workflow 7
   3. Risk Management 8
6. Evaluation 8
   1. Software Testing 8
      1. Approach 8
      2. Results 8
      3. Discussion 8
   2. User Testing (optional but strongly encouraged) 8
   3. Sponsor Feedback 8
7. Future Work 8
8. Reflection 9
   1. What Processes Worked? 9
   2. What Processes Did Not Work? 9
   3. How Did Processes Change Over Time 9
   4. How Will You Fine Tune Your Process for the Future? 9
9. References Cited 9

# Description

## Problem

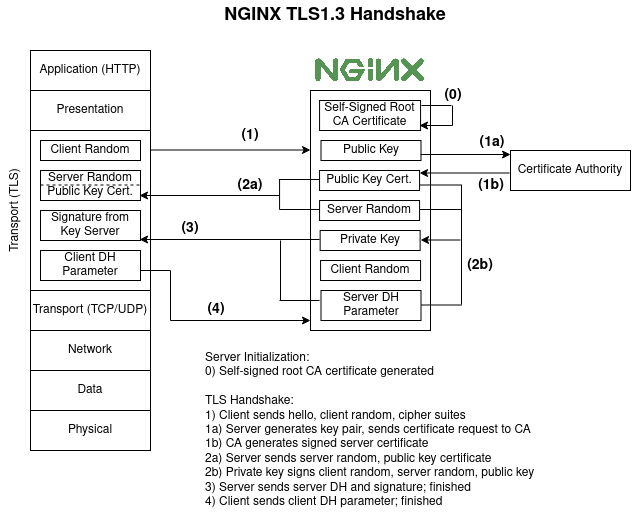
## The software industry of today has ever increasing demands in regards to portability, speed of user access, security, and modularization. As a result, there is a large push towards cloud native applications and infrastructure as a service (IaaS). The remote hosting of both servers and the processes performed on them presents a new host of challenges in terms of other industry needs previously handled on premise, such as authentication, security, and storage.

## 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.

## To counteract this, a framework must be designed and implemented to protect and restrict access to distributed systems. 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. [Begin new section] In this project, we will work to develop a prototype reverse proxy using existing open-source implementations, such as rproxy and NGINX. Then, a SPIFFE universal identity control system will be tested using QuantumSafe to simulate its robustness against quantum algorithms.

In this project, we will work to develop a prototype reverse proxy using existing open-source implementations, such as rproxy and NGINX. Then, a SPIFFE universal identity control system will be tested using QuantumSafe to simulate its robustness against quantum algorithms.



As we can see in Figure 1, a TLS handshake utilizing quantum-resistant encryption is similar to a standard TLS handshake currently used for all web authentication. Rather than changing the overall pattern of communication, the keys themselves are changed. While this may seem a small change at first, the entire stack must be re-configured to support the keys and certificates.

For example, NGINX itself must be re-configured to support specific types of modules that are relevant to the certificate authority (CA) that OpenQuantum Safe uses. The CA certificate must then also be compatible with the various dependencies for OpenSSL, the open-source software library used to actually generate these keys.

However, compatibility issues will likely arise when a NGINX server must communicate to a server using a different type of proxy – e.g. Envoy, Istio, etc. As a result, this stack must be thoroughly tested and integrated against an entire tech stack of authentication and encryption software. For each of these elements, their dependencies must also be modified to work with the new signatures – if they are unable to be modified, they must then be replaced. In the instance where no replacement can be found, one must be created from source.

One example of an element that required this much extra work is nghttp2, a modified open-source implementation of HTTP/2 optimized for various applications. It utilizes software called h2load, F5’s preferred load-tester for TLS communication. This application is completely incompatible with 3rd party signature schemes, requiring a full re-creation of its software.

## Goals and Outcomes

|  |  |
| --- | --- |
| **Goal** | **Desired Outcome** |
| 1. **Gain understanding of cloud and cloud native** | Give seminar presentations on cloud computing and cloud native.  All team members have a solid understanding of cloud computing and its relevant fields |
| 1. **Gain understanding of the relevant tech stacks and programming models** | Explore key products and technologies (NGINX, Envoy, K8s, Docker, containerization, etc); the pros and cons of each; and the relevant security concerns and future outlook |
| 1. **Gain understanding of quantum models of computation** | Give seminar presentation on quantum computation, the underlying mathematics, and its implications on cryptography and cybersecurity |
| 1. **Develop a first prototype with NGINX or ENVOY** | Develop a SPIFFE implementation that utilizes quantum resistant cryptographic algorithms |
| 1. **Benchmarks and results analysis** | Benchmark and analyze performance using automated testing to gain an in-depth analysis of its feasibility and scalability |
| 1. **Develop improved prototypes with multiple key products (NGINX, ENVOY, rproxy)** | Explore the utility of other products in improving current prototypes and developing new ones; potentially present the artifacts in different ways such as open source, article, blog and conference |

## 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 [1]. 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 [2].

# 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 [3]. 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 [4]. 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.

# 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 [5]. 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 [6]. 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 [7].

# 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

*[Add a draft of this section to the “Project Requirements Draft” assignment]*

## 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. NGINX is a reverse proxy that is able to provide security for applications. By the end of the spring semester, we will deliver a quantum resistant NGINX prototype to our sponsor that implements quantum resistant algorithms. The NGINX prototype will also utilize reverse proxies, containers, and the quantum-resistant library SPIRE. We will also deliver quantum resistant prototypes for other cloud networking services, such Envoy.

## Epics and User Stories

In this section, describe the *epics* that compose the minimum viable product (MVP) that your team will produce by the end of the semester. Think of an epic as “a large body of work that can be broken down into a number of user stories.”1 Thus, the scale of an epic is usually *multiple* sprint cycles. Each epic contains multiple user stories. Depending on the length of a sprint cycle, your team may be able to complete a few user stories in each sprint cycle. By the end of the semester, your team should complete one or more epics that, taken together, make up the target MVP delivered to your client.

## Epic 1: Cloud Native & Quantum Computing

Learn the key tools and technology stacks for Cloud Native Development. Additionally, learn how quantum computing will shape cryptography in the future.

### Cloud Computing Background

As a Computer Science student, I want to have a clear understanding of the various forms of Cloud Computing so that I can better utilize NGINX while developing our prototype.

* Thoroughly present topic to teammates
  + Explain various Cloud models and their use cases
* Advisor agrees that the presentation covered the desired topics in enough detail
* Teammates have largely absorbed the presented information
  + Established through quizzes given after the presentation. Teammates should score 90% or higher

**Story Points**: 13

**Priority Level**: 1

### Cloud Computing Tools

As a team member, I want to have a clear understanding of the tools and technologies which we will be utilizing to develop our final product.

* Presentation is given on Cloud Computing tools
* Advisor agrees that the presentation covered the desired topics in enough detail
* Teammates have largely absolrbed the presented information
  + Established through quizzes given after the presentation. Teammates should score 90% or higher

**Story Points**: 13

**Priority Level**: 1

### Background Mathematics

As a Computer Science student, I want to understand the mathematics underlying quantum theory, so I can effectively understand quantum encryption algorithms.

* Presentation is given on complex numbers, Bra-ket notation, vector spaces, Hilbert space, Hadamard gates, and Qubits
* Project mentor signs off on presentation
* Teammates pass the math quizzes given after the presentation with a score of 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 computer science student, 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, SPIFEE, and post-quantum encryption options
* Project mentor signs off on presentation
* Teammates can implement Shor’s algorithm using Qiskit

**Story Points**: 13

**Priority Level**: 1

## Epic 2: Develop an NGINX Prototype

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

### Establish Desired Requirements

As a computer science student, I want to establish requirements for an NGINX prototype so that I can finish the prototype on-schedule.

* Prototype uses SPIRE or similar quantum library
* Project mentor signs off on prototype requirements

**Story Points**: 8

**Priority Level**: 1

### Design the Prototype

As a team member, 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 is modeled in a way agreed upon by the team
* Teammates agree on which design to use
* Advisor signs off on design

**Story Points**: 13

**Priority Level**: 1

### Implement the Prototype

As a user, I want to be able to secure my NGINX prototype from quantum attacks, so that sensitive data stored on my server is not compromised

* Prototype meets requirements established in 3.4.1
* If practical issues a change in design from 3.4.2, the team agrees on a new design before proceeding
* Project mentor signs off on prototype

**Story Points**: 13

**Priority Level**: 1

### Testing, Benchmarking, and Deployment

As a computer science student, I want to write tests and perform benchmarking for my prototype so that it doesn’t crash during production

* Unit and Integration tests have been written for the reverse proxy and all pass
* Deployment script successfully deploys application to production
* NGINX Prototype performs reasonably well on benchmarks

**Story Points**: 13

**Priority Level**: 1

## Assumptions

We assume we will be renting a Platform as a service (PaaS) from a Cloud Provider to develop our NGINX prototype.

We assume NGINX is compatible with SPIRE, 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 SPIRE will be quantum resistant and that we will be building on top of SPIRE, instead of writing our own quantum resistant algorithms from the ground up.

We assume that Envoy is compatible with quantum resistant algorithms.

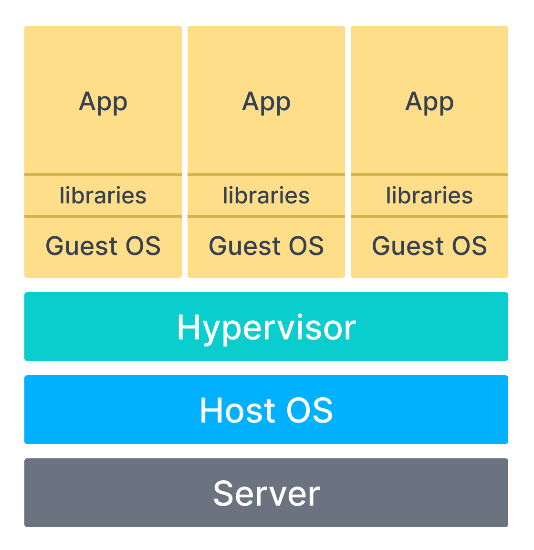
We assume that the quantum-resistant NGINX built from source will improve on the vanilla version of NGINX.

# Design

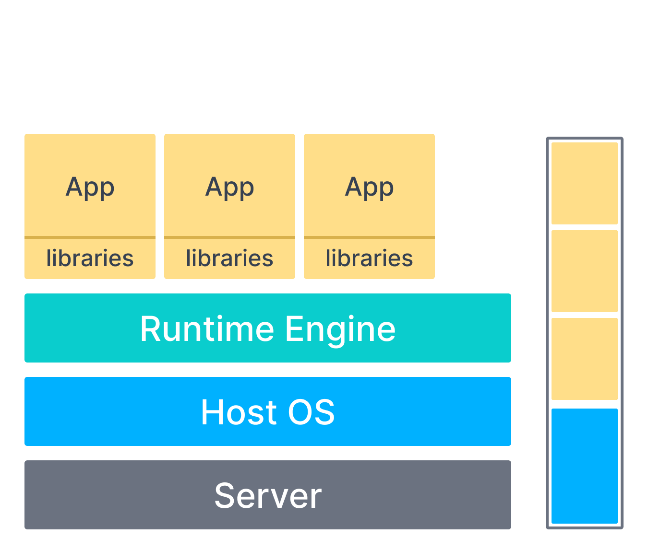
NOTE: The sections architecture, design, and UI design are combined. We have had no discussions with our mentor about the specifics of the design, but we have a clear picture about the problems we are trying to solve or mitigate.

## Software Architecture

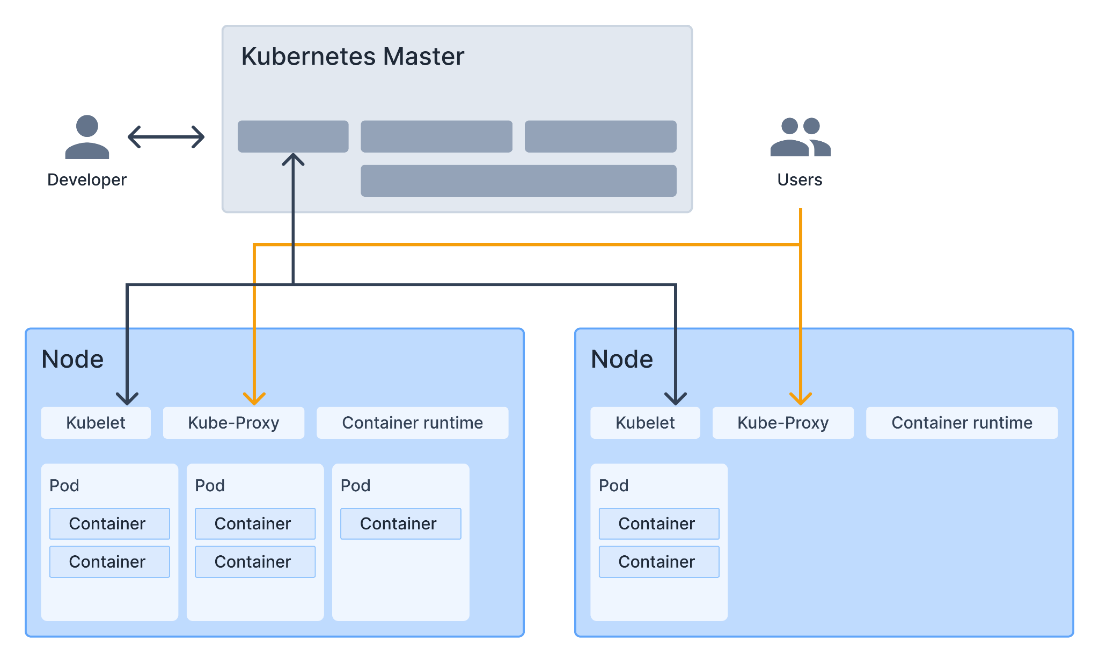
In distributed systems, authentication and controlling access become much more challenging. In a virtual machine (VM) setup, the host OS has its own operating system separate from the guest OS’s; this requires malicious individuals to target all OS’s, rather than just one.



However, distributed systems that use containers condense this into one entity – the host OS managed by the runtime engine. While this is far more resource efficient than VM, the single point of failure makes the issue of security much more important.

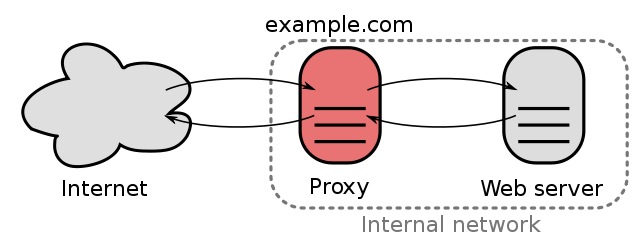


To manage these containers, we will use Kubernetes, a container orchestrator. It automates all the necessary tasks of load balancing, node creation, allocation, and deployment. Essentially, it allows a distributed system to run efficiently on a network of host OSs, or nodes.



However, Kubernetes must still be used in conjunction with a container engines and runtimes, like Docker or containerd. Even still, the security issue remains unaddressed. This is where we will introduce a reverse proxy, NGINX.

While a standard proxy server is intended to protect the client, a reverse proxy protects the servers. It accepts requests from a client, forwards them to one of the servers in the network, then returns the response to the client. As a result, the client only interacts with the reverse proxy server, not the server that actually processed the request. At a high level, the reverse proxy can be thought of similar to a switchboard for phone lines.



Once this framework is implemented, we can then use any algorithm of our choice to encrypt authentication and communication between the proxy and the servers or the clients. We will sandbox this using Open Quantum Safe (OQS), an open-source project specifically designed for developing and prototyping quantum-resistant cryptography.

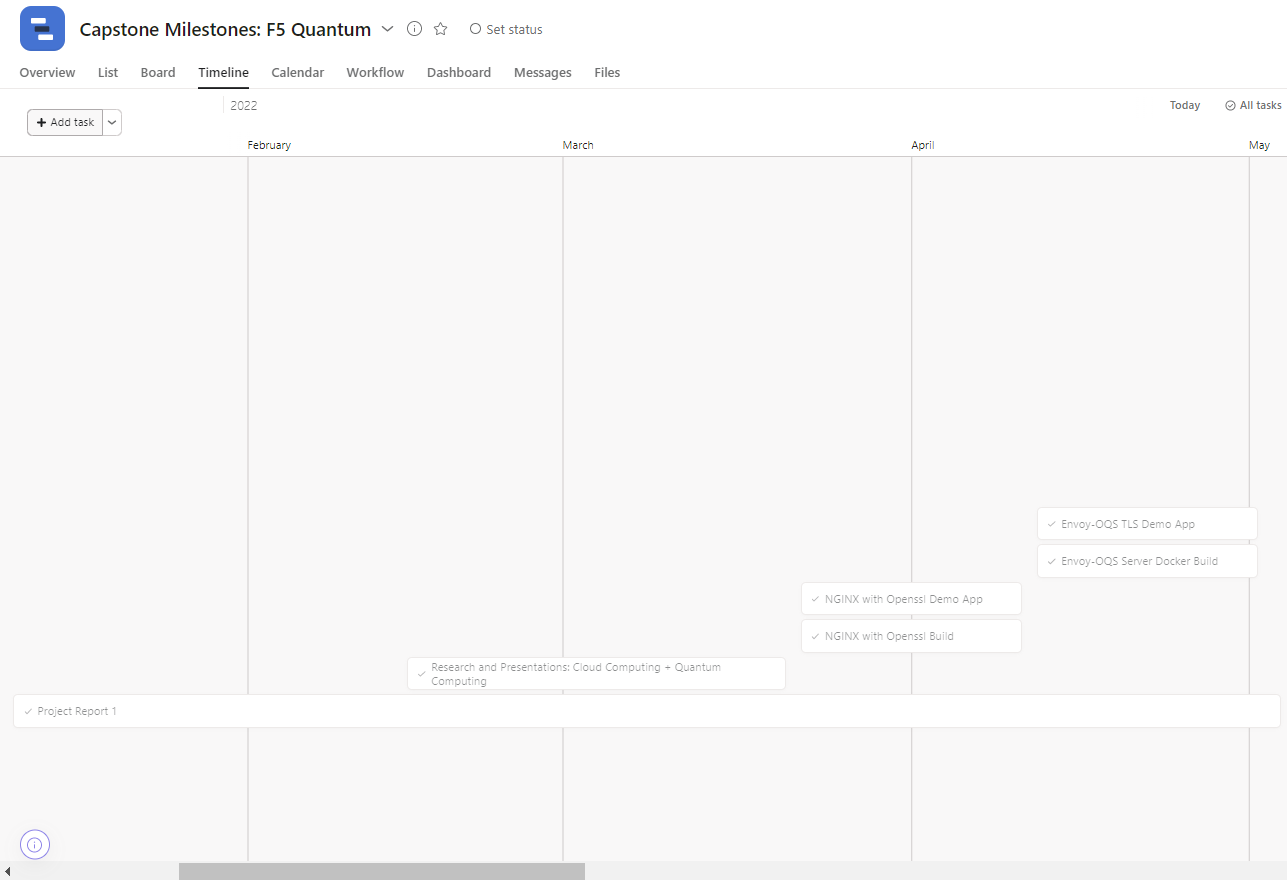
# Execution Plan

# Milestones

Graphical user interface, application

Description automatically generated**Table 1: Milestones**

Graphical user interface, text, application

Description automatically generated**Chart 1: Milestone Timeline View**

**Table 2: Milestone Tasks**

|  |  |
| --- | --- |
| **Milestone** | **Tasks** |
| Project Report 1 | * Write Project Description Draft * Write Project Team Members Draft * Write Project Requirements Draft * Write Project Design Draft * Revise drafts and submit final draft |
| Research and Presentations: Cloud Computing + Quantum Computing | * Research cloud computing * Research quantum computing * Write presentation syllabi * Create and present two 60 minute Cloud Computing presentations * Create and present two 60 minute Quantum Computing presentations * Study the presentations and continue to research missing areas of knowledge |
| NGINX with Openssl Build | * Create Dockerfile to build NGINX with openssl * Test Dockerfile on multiple operating systems and edit as necessary * Write instructions for running the build |
| NGINX with Openssl Demo App | * Create basic application * Run application using the NGINX openssl build |
| Envoy-OQS Server Docker Build | * Create basic application * Run application using the NGINX openssl build |
| Envoy-OQS TLS Demo App | * Create Dockerfile to build Envoy with oqs * Test Dockerfile on multiple operating systems and edit as necessary * Write instructions for running the build |
| Istio Build | * Create Dockerfile to build Istio * Test Dockerfile on multiple operating systems and edit as necessary * Write instructions for running the build |
| Istio Demo | * Create demo that deploys applications using the Istio build |
| NGINX Benchmarking | * Build H2Load on VM * Run benchmark tests * Create graphs and summarize the results |
| Envoy Benchmarking | * Run benchmark tests * Create graphs and summarize the results |
| Create Presentation for Kubecon Conference | * Create presentation slides * Add Istio Demo, Envoy Demo, NGINX demo, NGINX benchmarking demo, and Envoy benchmarking demo to presentation * Write presentation script |
| Rehearse Presentation for Kubecon Conference | * Ensure everything is setup on laptops and backed-up * Memorize presentation * Practice presentations |
| Organize and Document Repository for Open-Source Development | * Re-organize folders in Github * Add and update Readme’s for each folder |
| Build Upon Current Demos and Add New Demos to Showcase Quantum Resistant Proxies | * Improve upon current demos * Add new more advanced demos to showcase quantum resistant proxies |
| Project Report 2 | * Write Project Design Draft * Wrote Project Execution Plan Draft * Wrote Project Evaluation Draft * Wrote Project Future Work Draft * Wrote Project Reflection Draft * Revise drafts and submit final draft |

# Workflow

**Table 3: Workflow on Kanban**

Kanban board can be found here: [F5 - Quantum (github.com)](https://github.com/orgs/wsu-cpts421-sp22/projects/2/views/11)

# Graphical user interface, text, application Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

# Risk Management

|  |  |
| --- | --- |
| **Milestone** | **Risks & Mitigations** |
| Project Report 1 | * 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 |
| Research and Presentations: Cloud Computing + Quantum Computing | * The information delivered in the presentations is not accurate   + Fact check the information with multiple credible sources |
| NGINX with Openssl Build | * Blocked – no way to get build to work   + Create good documentation about what has been tried and why it does not work   + Focus on the other builds and demos and make them strong |
| NGINX with Openssl Demo App | * The chosen demo does not work with the NGINX Openssl build   + If completely blocked – try a different demo |
| Envoy-OQS Server Docker Build | * Blocked – no way to get build to work   + Create good documentation about what has been tried and why it does not work   + Focus on the other builds and demos and make them strong |
| Envoy-OQS TLS Demo App | * The chosen demo does not work with the Envoy OQS build   + If completely blocked – try a different demo |
| Istio Build | * Blocked – no way to get build to work   + Create good documentation about what has been tried and why it does not work * Focus on the other builds and demos and make them strong |
| Istio Demo | * The chosen demo does not work with the Istio build   + If completely blocked – try a different demo |
| NGINX Benchmarking | * Benchmarking does not accurately show NGINX results   + Compare results with vanilla NGINX and make sure that they are similar |
| Envoy Benchmarking | * Benchmarking does not accurately show Envoy results   + Compare results with vanilla Envoy and make sure that they are similar |
| Create Presentation for Kubecon Conference | * Presentation gets deleted/ does not work on computer   + Have presentation setup on backup computer |
| Rehearse Presentation for Kubecon Conference | * The rehearsed presentation is not of appropriate quality for conference   + Start rehearsing early so there is time to make adjustments   + Practice in front of a critical audience   + Have back-up script on computer in case parts of presentation are forgotten |
| Organize and Document Repository for Open-Source Development | * The new structure of the Github Repository is not improved   + Have people who are not apart of the project test out the new repository and make adjustments as necessary |
| Build Upon Current Demos and Add New Demos to Showcase Quantum Resistant Proxies | * No major risks – these are just additional pieces that could strengthen the project |
| Project Report 2 | * 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 |

# 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.

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 |

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 |

### 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] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

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

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

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

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