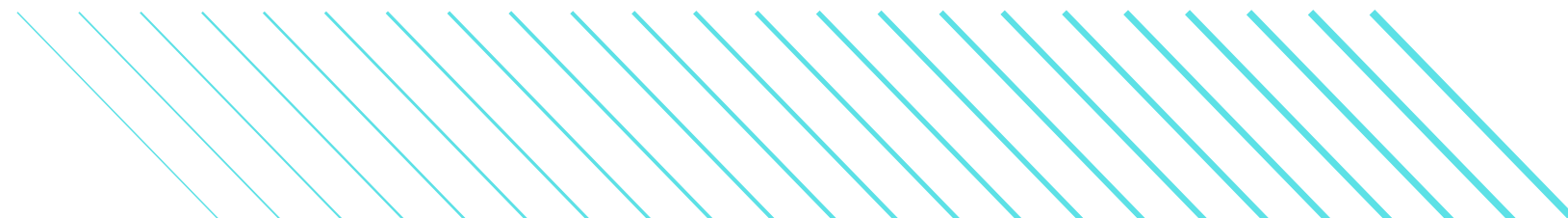
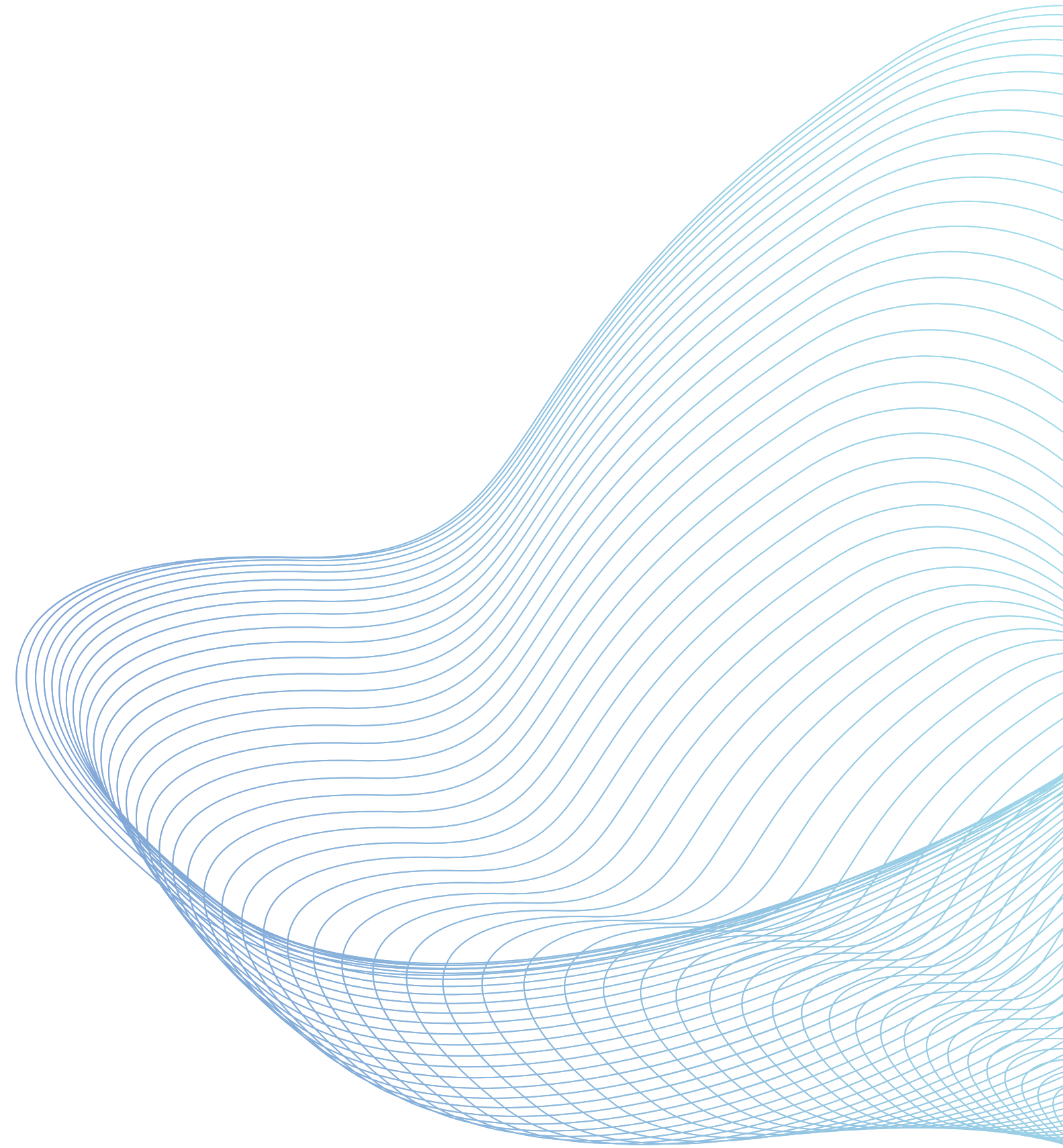
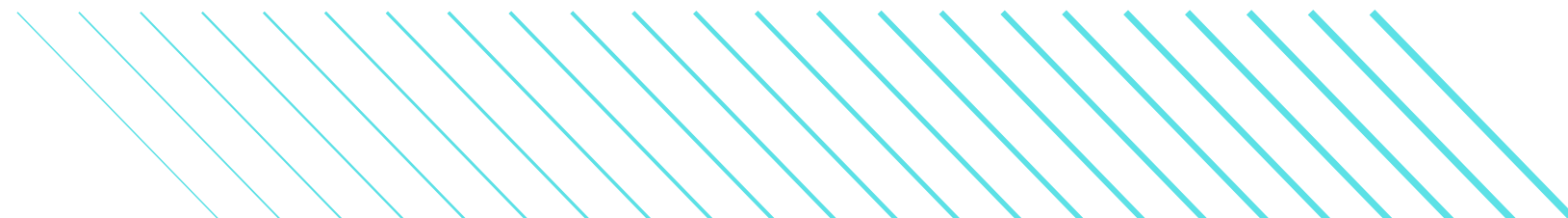
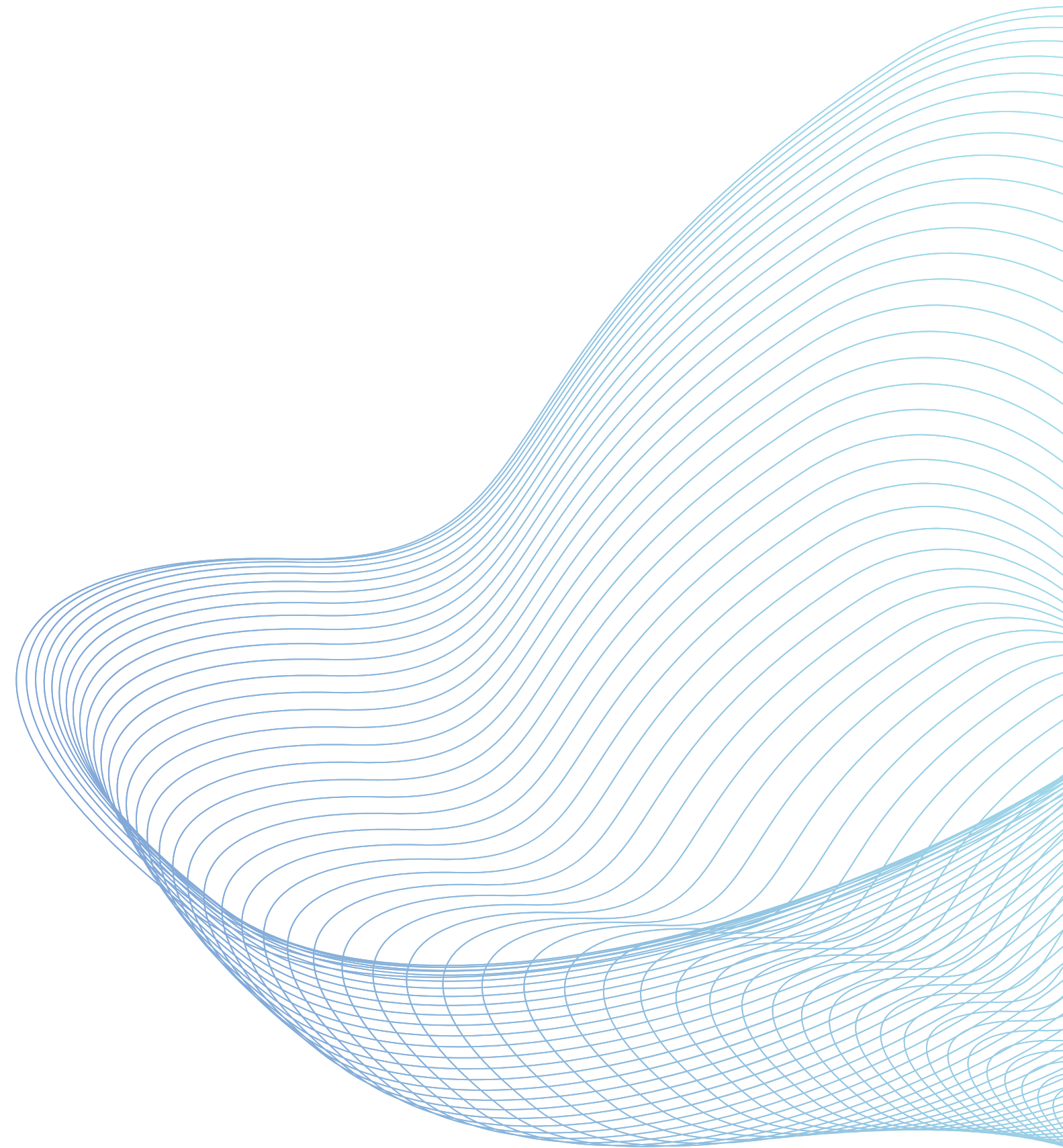


DEMO#1

Security : Mariam Osama



SPRINT 1



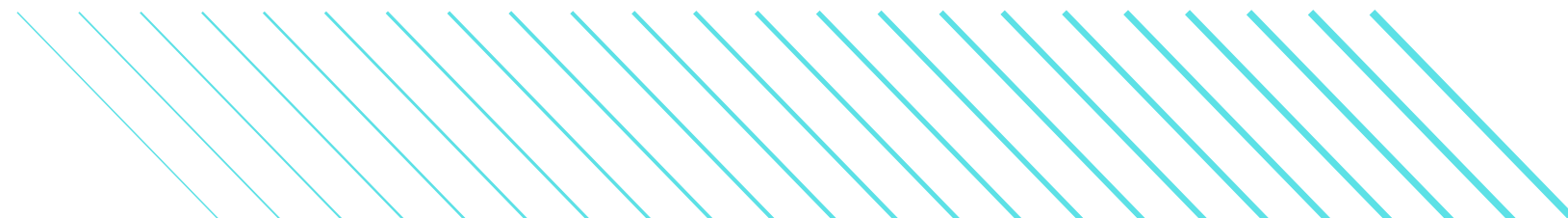
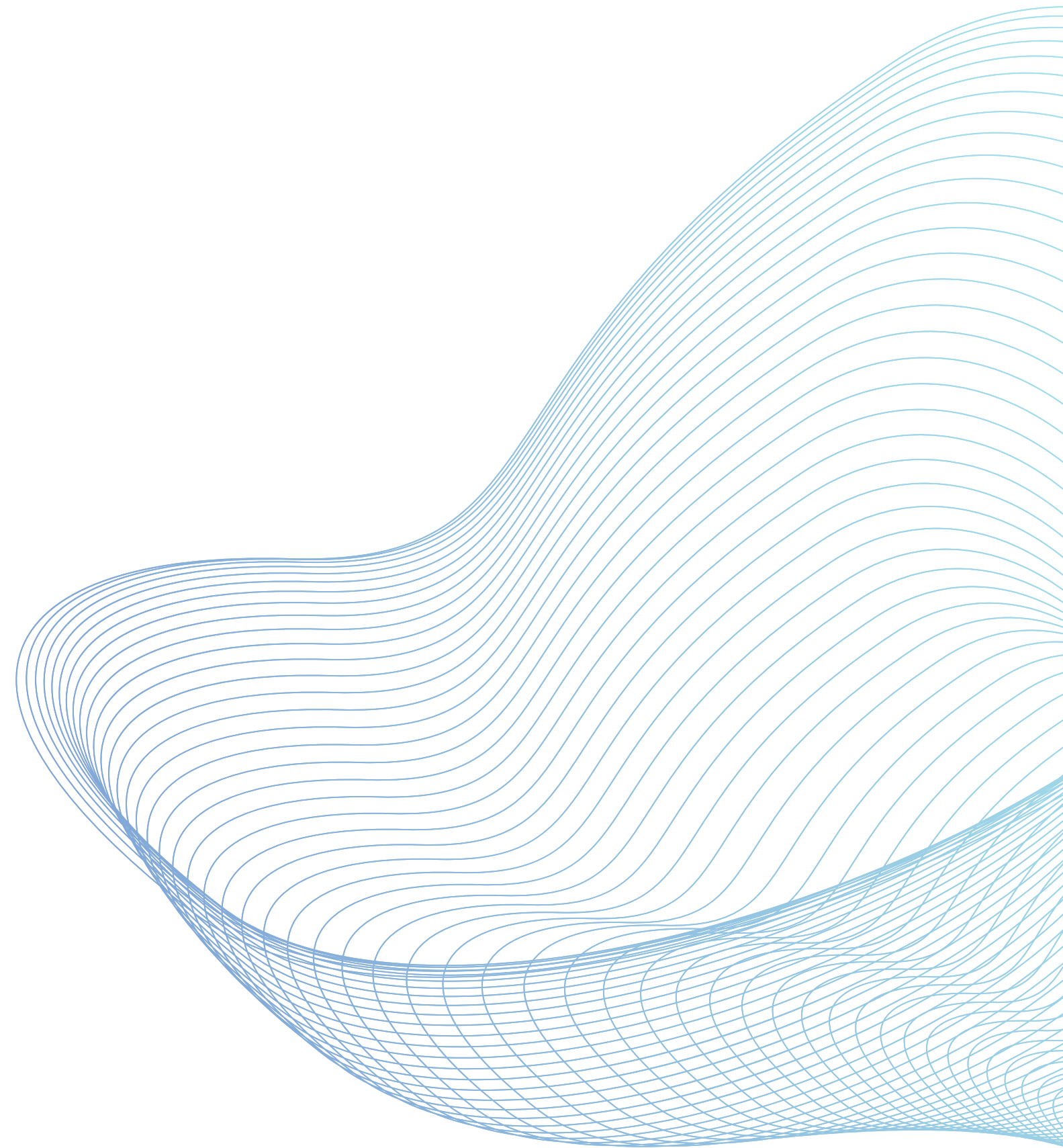
Learnig Phase

Covered the following:

- Basics of K8s
- What is Helm
- Completed the CKS course



SPRINT2



Searching and Testing

The Tasks:

- K8s on-premises hardening
- Implementing a built-in Vulnerability Scanning Tool



K8s on-premises hardening what has been done so far:



Wrote a best practice document that will help me in the hardening task.

Table of Contents

Contents

Introduction

Cluster Setup and Configuration

Node Security

Network Policies

API Server Security

Image Security

Pod Security

Secrets Management

Logging and Monitoring

Backup and Disaster Recovery

Hardening Tools and Scripts

Security Updates and Maintenance

User and Role Management

Physical Security and Network Segmentation

Education and Training

Conclusion

Cluster Setup and Configuration

Creating a secure foundation is paramount to ensuring the overall security of your Kubernetes cluster. The initial setup and configuration choices [lay](#) the groundwork for subsequent security measures. This section outlines key considerations and best practices to follow during the setup phase.

Minimal Installation

Start with a minimal installation of Kubernetes to reduce the attack surface. Include only the necessary components required for your applications. Avoid installing unnecessary features that could potentially introduce security vulnerabilities.

Strong Authentication and Authorization

Implement strong authentication and authorization mechanisms to control access to your cluster. Utilize Kubernetes' Role-Based Access Control (RBAC) to define fine-grained permissions based on user roles and responsibilities. Assign the principle of least privilege, ensuring that users and components have only the permissions they need to perform their tasks.

Disable Default or Unnecessary Services

Disable or remove default services or features that you don't intend to use. Kubernetes often comes with optional services enabled by default, such as the Kubernetes Dashboard or insecure APIs. Disable these services if they are not necessary for your use case to minimize potential attack vectors.

K8s on-premises hardening what has been done so far:

Penetration testing involves identifying vulnerabilities and weaknesses in your cluster's security. Below are some example commands and scripts that you can use as a starting point for conducting penetration tests on your Kubernetes cluster. Keep in mind that penetration testing should only be performed in controlled environments and with proper authorization to avoid disrupting production systems.

1. Unauthorized Access Attempts

Attempt to access the Kubernetes API server with invalid or no credentials:

```
```bash
Without credentials
curl -k https://<api-server-url>

With invalid credentials
curl -k -u invaliduser:invalidpass https://<api-server-url>
```
```

2. Role-Based Access Control (RBAC) Testing

Test RBAC by attempting to access resources and perform actions that are not allowed by the assigned roles:

```
```bash
List pods in a restricted namespace
kubectl get pods -n restricted-namespace
```
```

Also collected a couple of scenarios for testing the cluster guided by the CVEs in this repository(<https://github.com/magnologan/awesome-k8s-security#slides>)

...

4. Image Vulnerability Testing

Deploy a pod with a container image containing known vulnerabilities:

```
```bash
Deploy a vulnerable pod
kubectl apply -f vulnerable-pod.yaml

Wait for the pod to start and check logs for vulnerabilities
kubectl logs <pod-name>
```
```

5. Secrets Exposure

Implementing a built-in Vulnerability Scanning Tool what has been done so far:

```
(YAML)
apiVersion: v1
kind: Pod
metadata:
  name: vulnerability-scanner
  labels:
    app: vulnerability-scanner
spec:
  containers:
    - name: vulnerability-scanner-container
      image: your-vulnerability-scanner-image
      command: ["/bin/sh"]
      args:
        - -c
        - |
          # Install and run vulnerability scanning tools
          apt-get update && apt-get install -y tool1 tool2 tool3
          tool1 scan > tool1_report.txt
          tool2 scan > tool2_report.txt
          tool3 scan > tool3_report.txt
          # Collect scan reports into one log
          cat tool1_report.txt tool2_report.txt tool3_report.txt > scan_report.log
          # Encrypt and securely store the log (adjust as per your setup)
          gpg --recipient your_key_id --output encrypted_report.gpg --encrypt scan_report.log
  volumeMounts:
    - name: report-storage
      mountPath: /reports
  volumes:
    - name: report-storage
      emptyDir: {}
```

2. InitContainers:
InitContainers are executed before the main containers in a pod start. You can use them to run vulnerability scanning tools before your application containers start. This approach ensures that the scanning is performed before any services become active.

3. CronJobs:

mountPath: /reports

4. Docker Image:

This image can be deployed as a separate container within your Kubernetes cluster, and you can run it manually or schedule it to perform vulnerability scans as needed.

1) Create a Docker Image:


Use a base image with the required tools pre-installed
FROM ubuntu:latest

Install necessary packages
RUN apt-get update && apt-get install -y tool1 tool2 tool3 gnupg

Copy your vulnerability scanning scripts to the image
COPY scan.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/scan.sh

Set the entry point to your scanning script
ENTRYPOINT ["/usr/local/bin/scan.sh"]

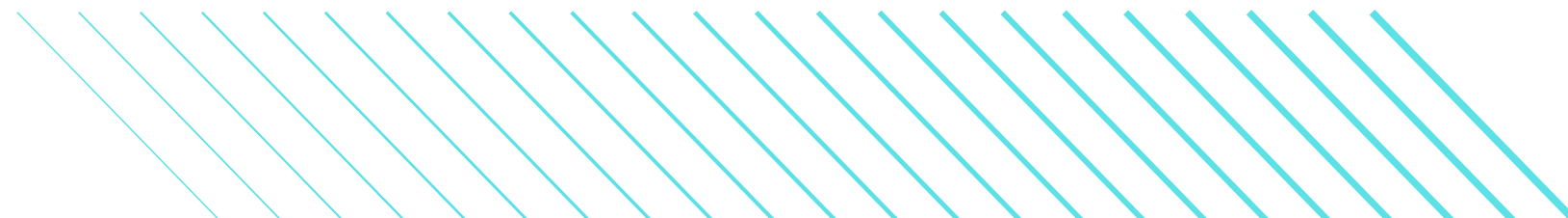
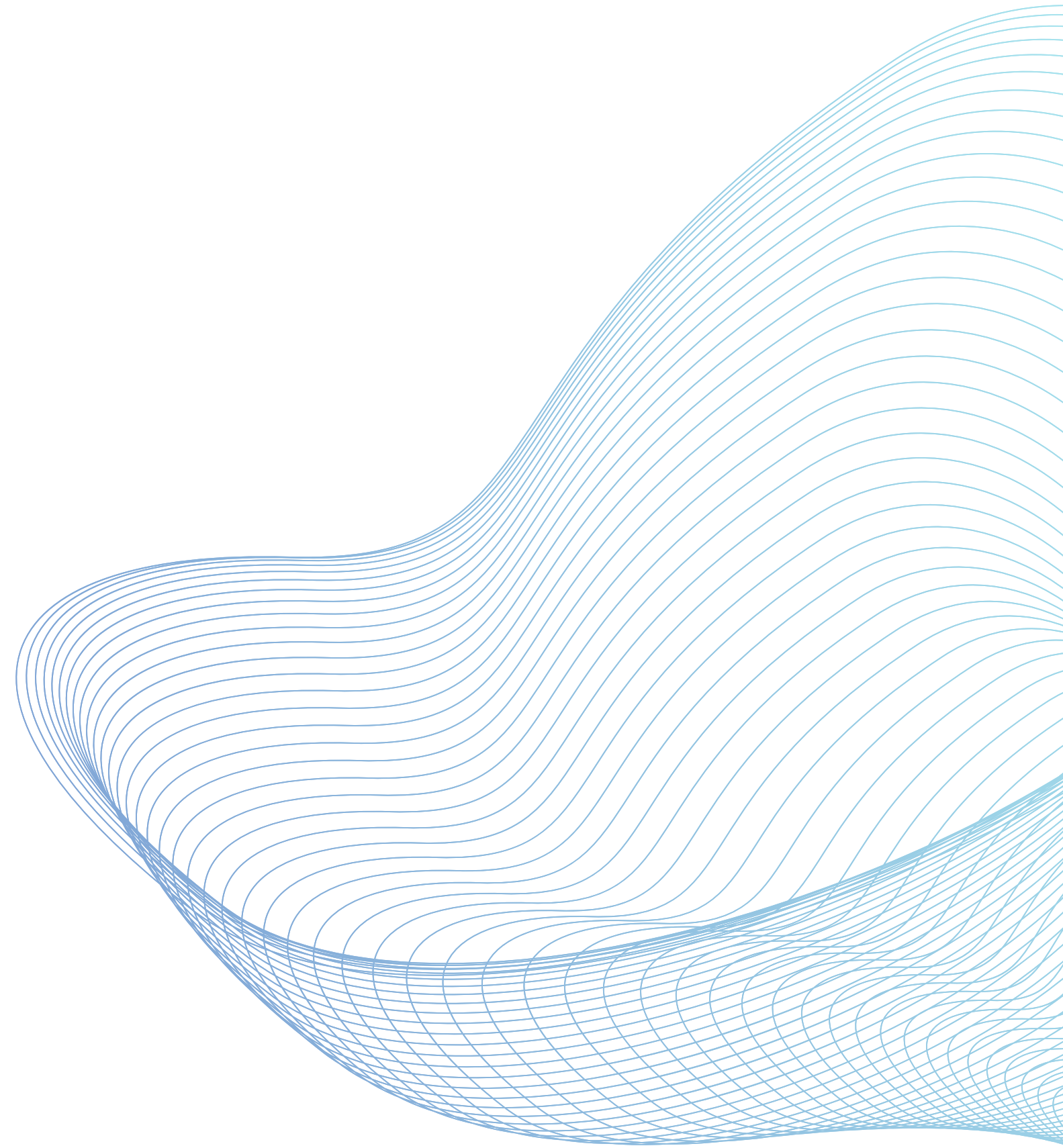
Researched the different options to
implement the task




Regarding this sprint I will work on two things :

1. Create a K8s playground to test the hardening best practices and the penetration scenarios
2. Will continue the research on the second task(e.g. advantages and disadvantages of every method)

SPRINT 3 & SPRINT 4 OBJECTIVES





By that time the cluster should be up and running and then comes the role of applying the results of sprint2 :

1. Hardening the
Clusters/ Penetration Testing
2. Integrating a Built-in
Vulnerability Scanning Tool.

**THANK
YOU**

