**Prototype Requirements**

**<P-06>:<Anomalous Login Detection System via ELK Stack>**

**<team member names & ids>**

| **Student ID** | **Name** |
| --- | --- |
| **26100286** | **Mohammad Mustafa** |
| **26100015** | **Muhammad Aaffan Khan Niazi** |
| **26100399** | **Mustafa Hussain** |
| **25100022** | **Shehroz Faryad** |

**Table of Contents**

[1.](#_upkybcjpfoej) Introduction 3

[2.](#_m691o4pz6pq2) Instructions 4

[3.](#_olcuivhrcovn) List of Requirements for Prototype 5

[4.](#_j7cj0fra8krk) Review checklist 6

# Introduction

<Give an overview of the project here. The overview must highlight the overall objectives of the project and its potential users. Do not exceed one page for description of this section.>

The project is concerned with the detection of unusual or suspicious login activity in real-time using the ELK stack (Elasticsearch, Logstash, Kibana) along with Wazuh. The primary objective of the system is to collect authentication logs and search patterns that can possibly indicate a potential security problem. Such patterns are things like multiple attempts of logging in over a short span of time or abrupt alterations in the location where the user is logging in. The system is able to detect these behaviors and minimize the chances of unauthorized access.

System administrators and security engineers are the key individuals that will utilize this system. To them, the project offers a user-friendly dashboard that lets them easily view the activity of the login at a glance, and also offers instant notification whenever something out of the ordinary occurs. This implies that there is less time wasted in searching through raw logs and that action is taken much faster when an issue arises.

In general, the project will enhance the speed at which organizations can recognize and react to abnormal login activity. Having clear visibility, real time alerts and effective monitoring, system administrators and security engineers can ensure that they protect systems better and contain potential threats.

# Instructions

* The prototype will be developed in two phases. Select a subset of system requirements and implement them. The result of each phase of prototype development must be a working system with the selected set of requirements implemented completely. Mere mock-up screens will NOT be accepted.
* While you may choose to implement Login/Logout functionality for prototype, you must also implement some core/business use cases of the system.
* Select the set of requirements keeping in mind that you have a total of 4 weeks for prototype development. You may be asked to add more requirements if more can be achieved in the given duration.
* The prototype must be built using the tools and technologies that you have selected for your system development.
* Follow standard coding practices.
* By the end of the prototype development phase,
  1. You should have learnt development tools and technologies.
  2. You should have a clear idea of detailed technical architecture of your system. After the prototype phase, you will be required to define detailed technical architecture of your system.
* **Prototype Submission**
* **Prototype Phase—1**
  1. Deploy the properly tested **working prototype** on an online hosting platform.
  2. Upload the **Code (zip file)** with proper comments in “prototype” folder of your project’s Github repository.
  3. Fill the provided template for **Readme-Prototype.txt** file and upload in “prototype” folder of your project’s Github repository.
  4. Fill the provided **Code Review Checklist** and upload in “prototype” folder of your project’s Github repository.
* **Prototype Phase—2**
  1. Deploy the properly tested **working prototype** on an online hosting platform.
  2. Upload the **Code (zip file)** with proper comments in “prototype” folder of your project’s Github repository.
  3. Prepare a **3-4 minutes video** that explains the functionality of your system prototype—to be uploaded in “prototype” folder of your project’s Github repository.
  4. Fill the provided template for **Readme-Prototype.txt** file and upload in “prototype” folder of your project’s Github repository.
  5. Fill the provided **Code Review Checklist** and upload in “prototype” folder of your project’s Github repository.

# List of Requirements for Prototype

<List down the requirements selected for prototype development separately for each part.>

* **Prototype Use Cases: Phase—1**

| **Requirements** | |
| --- | --- |
| **Sr#** | **Use Case Name** |
| 1 | Log ingestion setup - Configure Wazuh agents to collect logs from Windows and Linux hosts and forward them to Logstash. |
| 2 | Data visualization - Build Kibana dashboards to visualize login events (successful vs failed logins). |
| 3 | Anomaly Detection (basic) - Implement detection rules for brute-force or failed login anomalies using Wazuh Manager. |
| 4 | Basic log parsing - Configure Logstash pipelines to parse and normalize incoming authentication logs into ECS format. |
| 5 | Alert notification - Configure Wazuh alert module to generate and display notifications in Kibana upon anomaly detection. |
| 6 | Data Filtering - Filter out irrelevant log entries to focus on authentication-related events only. |
| 7 | Timestamp normalization - Normalize log timestamps into a consistent UTC format for uniform visualization and correlation. |
| 8 | Visualization of login trends - Develop Kibana visualizations (bar/line charts) to show login success/failure trends over time. |

* **Prototype Use Cases: Phase—2**

| **Requirements** | |
| --- | --- |
| **Sr#** | **Use Case Name** |
| 1 | Advanced Anomaly Detection - Implement rules for complex patterns (geo-impossible logins, repeated failed attempts). |
| 2 | Alert Categorization - Add severity levels (Info, Warning, Critical) based on rule logic for better triage in Wazuh. |
| 3 | User Access Control - Define multiple user roles in Kibana (viewer, analyst, administrator) to ensure only authorized users. |
| 4 | Real-Time Dashboard Update - Enable auto-refresh and real-time data stream updates every 10 seconds in Kibana dashboards. |
| 5 | Integration with Email - Extend alerting via SMTP (Azure Mail) for SOC notifications. |
| 6 | Azure Resource Monitoring Integration - Integrate Azure Monitor with Kibana to visualize system-level metrics such as CPU, memory etc. |
| 7 | Automated Incident Summary Reports - Generate daily operational summaries that compile triggered alerts, affected user accounts, and IP locations, automatically shared with administrators via email for review |
| 8 | Performance and Load Evaluation - Conduct load testing (≥10,000 logs/sec) and monitor resource utilization to validate system stability. |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| 1,2,3 | **Mohammad Mustafa** |
| 1,2,3 | **Muhammad Aaffan Khan Niazi** |
| 1,2,3 | **Mustafa Hussain** |
| 1,2,3 | **Shehroz Faryad** |