AN ENGINEERING PROJECT REPORT

ON

"Threat Detection System"

Submitted By

Sushil Paudel – 200348

Bibek Pandeya – 200311

Bhushan Bartaula – 200310

Prakash Lamichhane – 200323

Submitted To

Department of IT and Computer Engineering

In partial fulfillment of requirement for the degree of Bachelor of engineering in Computer Engineering.

Cosmos College of Management & Technology

(Affiliated with Pokhara University)

Tutepani, Lalitpur, Nepal

Date of Submission: - 2082/01/22

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to all faculty members who supported and encouraged us, without their guidance this project would not have been possible.

We are deeply grateful to the Department of Information and Communications Technology, **Cosmos College of Management and Technology, Tutepani, Lalitpur** for providing us with all the consultation and knowledge to conduct this project. We show our sincere gratitude to our teachers for effective suggestions and strengthening in the completion of this proposal.

We would also like to acknowledge the immense contribution of our supervisor for the kind support and guidance during the project implementation phase as well as his inspiration to move forward for the completion of project on time.

Any consecutive criticism and suggestions for improvements are warmly welcomed.

Sushil Paudel

Bibek Pandeya

Bhushan Bartaula

Prakash Lamichhane

ABSTRACT

Our **“Threat Detection System”** installs small agents on each device to collect information on running processes, network connections, file operations and user sessions. These agents send the data securely to a central server that stores and organizes the logs before passing them to an anomaly detection engine. This engine learns what normal behavior looks like and checks new data for anything unusual, flagging potential security incidents.

The system automates the entire workflow from data collection to analysis. When a threat is detected, an alert appears in real time on a dashboard so administrators can respond quickly. Its modular design makes it easy to add new data sources or detection methods, keeping the system effective as threats evolve in today’s changing landscape.

Table of Contents

LISTS

a) List of Figures:

b) List of Tables: -

c) List of Abbreviations: -

Ui: - user interface

Ux :- user experiences

Apk :- applications

www: - world wide web

jpeg: - joint photographic experts Graphics

png: - portable network graphics

com: - communication

Pdf:- Portable Document Format.

1. INTRODUCTION

1.1 Background

Cybersecurity threats are becoming more complex, making it harder for traditional security systems to keep up. Our **Threat Detection System** addresses this by using machine learning to monitor device activities in real-time. Lightweight agents collect data from devices and send it to a central server for analysis, detecting anomalies and flagging potential threats early. This system helps organizations respond faster to attacks, providing a proactive solution to modern cybersecurity challenges.

1.2 Statements of the problems

With the increasing frequency and sophistication of cyberattacks, traditional security systems often fail to detect advanced threats in time. Many organizations rely on reactive approaches, making it difficult to prevent data breaches, malware infections, and insider threats. There is a need for a proactive, real-time threat detection system that can continuously monitor and analyze device activity, identify anomalies, and respond to emerging threats before they cause significant damage.

1.3 Objectives

- Develop **a real-time threat detection system** that continuously monitors device activity and identifies suspicious behavior using machine learning.
- Provide **proactive alerts** to administrators, enabling fast response to detected threats.
- Create **a scalable solution** that can easily expand to accommodate more devices and evolving security needs.

2. LITERATURE REVIEW

As cyber threats continue to evolve, traditional security systems are becoming less effective. Machine learning (ML) and artificial intelligence (AI) have emerged as powerful tools for proactive threat detection. **Dr. Anil Kumar Yadav** (India, 2020) showed how unsupervised machine learning can detect anomalies in system logs, which aligns with our approach of using ML to analyze telemetry data for threat detection.

Dr. Sushil Bhatta (Nepal, 2019) emphasized the importance of real-time telemetry collection from devices, which our project also incorporates by using lightweight agents to send data to a central server for analysis. **Ravi Kumar** (India, 2021) explored the advantages of centralized log aggregation, which allows for quicker threat identification, similar to our system's design for real-time processing.

Automated response systems are essential for minimizing damage, as highlighted by **Dr. Priya Sharma** (India, 2018), whose work focuses on integrating automated alerts and responses to detected threats. Additionally, **Prof. Rajesh Bhattarai** (Nepal, 2020) noted the importance of scalability in cybersecurity systems, which is addressed by our scalable architecture that can handle increasing data and evolving threats.

In conclusion, combining ML for anomaly detection, real-time telemetry, and centralized processing offers an effective defense against modern cyber threats, forming the basis of our project’s design.

3. REQUIREMENT ANALYSIS

3.1 Technologies used

Component	Technology
Agent (Log Collector)	Python (psutil, socket, requests)
API / Backend Server	Django REST Framework / FastAPI
Database	PostgreSQL or MongoDB (for log storage)
ML Engine	Python (scikit-learn / PyOD / TensorFlow)
Dashboard (Admin Panel)	Django + HTML/CSS + JavaScript
Deployment	Docker, Gunicorn, Nginx (optional)
Version Control	Git & GitHub
Others	PyInstaller (to convert agent to .exe), JWT (for authentication)

3.2 Other Tools used

- Visual Studio Code: Integrated Development Environment (IDE) for development
- Adobe Illustrator: Vector Graphic Designing Software
- Adobe Photoshop: Raster image processing Software
- Adobe XD: Prototyping and UI/UX designing Software
- Brave, Google Chrome: Web Browsers for testing source code

4. METHODOLOGY

4.1 Software Process Model

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the order of activities of the process and the sequence in which they are performed. There are various types of Software Process Model but in this project, we are going to use Iterative and Increment Model.

Iterative Incremental Model:

Since, our project has multiple parts (Agent, API server, ML engine, Dashboard), which can be built and tested in small increments with the help of Iterative Increment Model. It has the following phases:

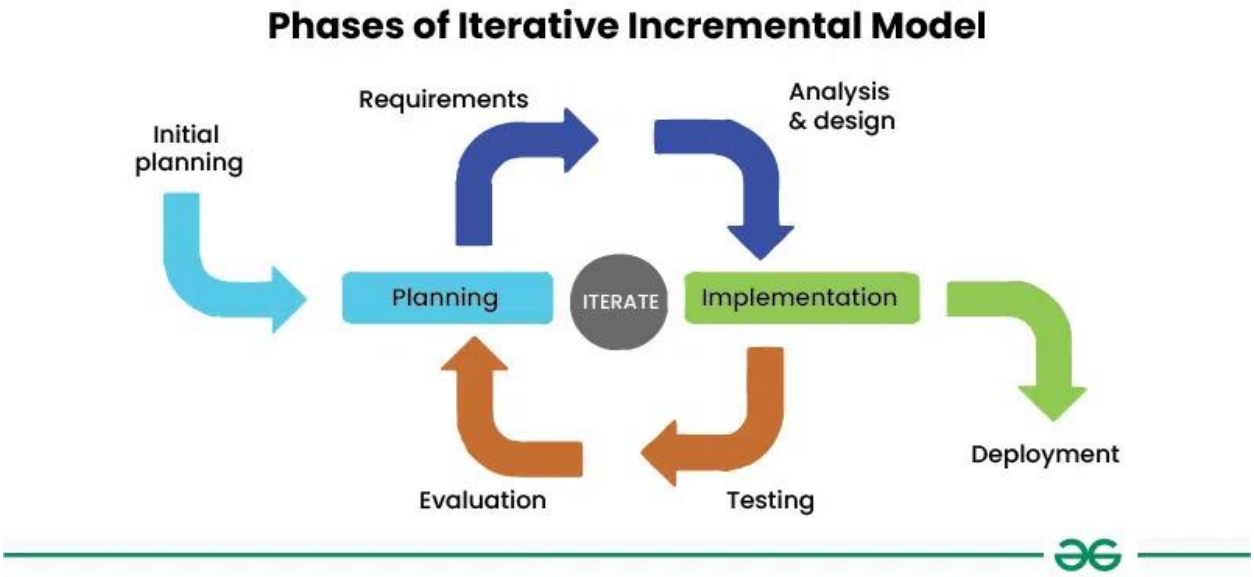

Figure 1 :Phases of Iterative Increment Model

1. Planning Phase

In this phase, the team identifies the goals and objectives of the project, along with the project scope, requirements, and constraints on them. The team then identifies different iterations that would be needed to complete the project successfully.

2. Requirements Analysis and Design Phase

In this phase, the requirements met are then analyzed and the according system is designed based on these requirements. The projected design should be modular, which would allow easy modification and testing in subsequent iterations.

3. Implementation Phase

In this phase, the system is implemented based on the design created in the previous phase. The implementation should be done in small, manageable pieces or increments, which can then be tested in the next phase of the cycle.

4. Testing Phase

In this phase, the system is tested against the requirements identified in the planning phase. Testing is done for each iteration, and any defects or issues are identified and resolved, and this helps in each iteration.

5. Evaluation Phase

In this phase, the team evaluates the performance of the system based on the results of testing. Feedback is gathered from users and stakeholders, and changes are made to the system as needed, which makes the system more scalable and flexible.

6. Incremental Release

In this phase, the completed iterations are released to users and stakeholders. Each release builds on the previous release, providing new functionality or largely improving existing functionality.

Overall, following a structured methodology ensures that the **Threat Detection System** is developed efficiently and effectively, meets the project requirements, and provides positive user experience for each generation of the user.

5. ALGORITHM: -

Here is a general algorithm for execution of our project:

- **Start Agent**: Agent runs in background after system boots.
- **Collect Logs**: Gather data on processes, files, network, and user activities.
- **Send Logs**: Securely transmit logs to the backend server.
- **Store Logs**: Server receives and stores logs in the database.
- **Analyze Logs**: ML model scans logs for unusual behavior.
- **Detect Threats**: If an anomaly is found, raise a security alert.
- **Notify Admin**: Show alerts on dashboard for admin action.

6. FLOWCHART: -

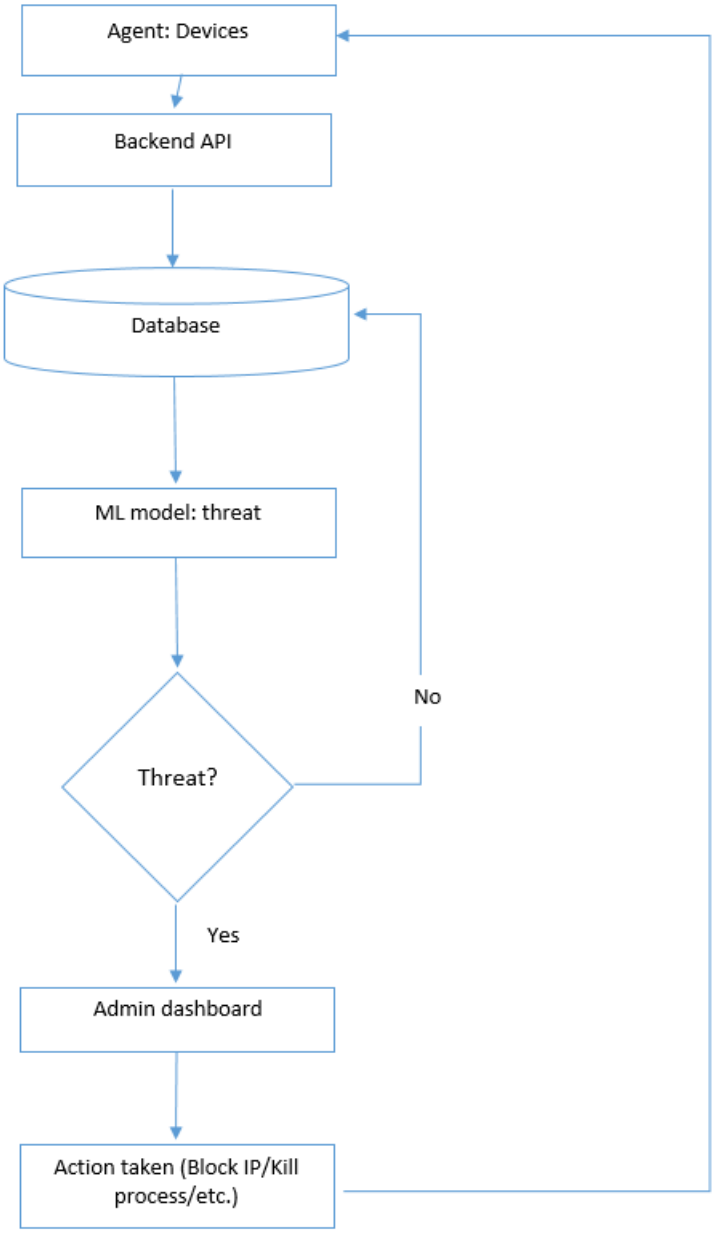

Figure 2 :Flowchart

7. USE CASE DIAGRAM: -

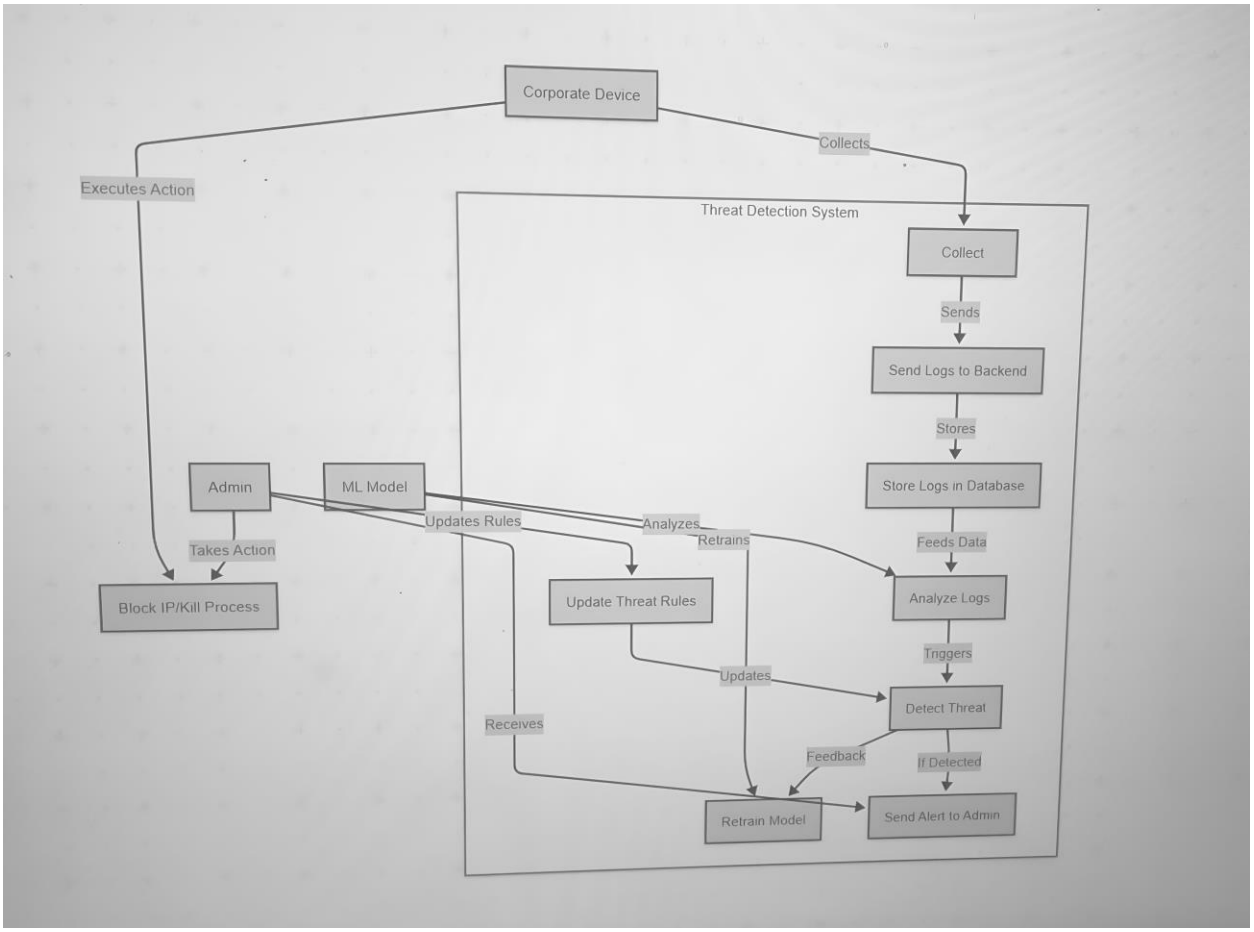

Figure 3 :Use Case Diagram

8. ER DIAGRAM: -

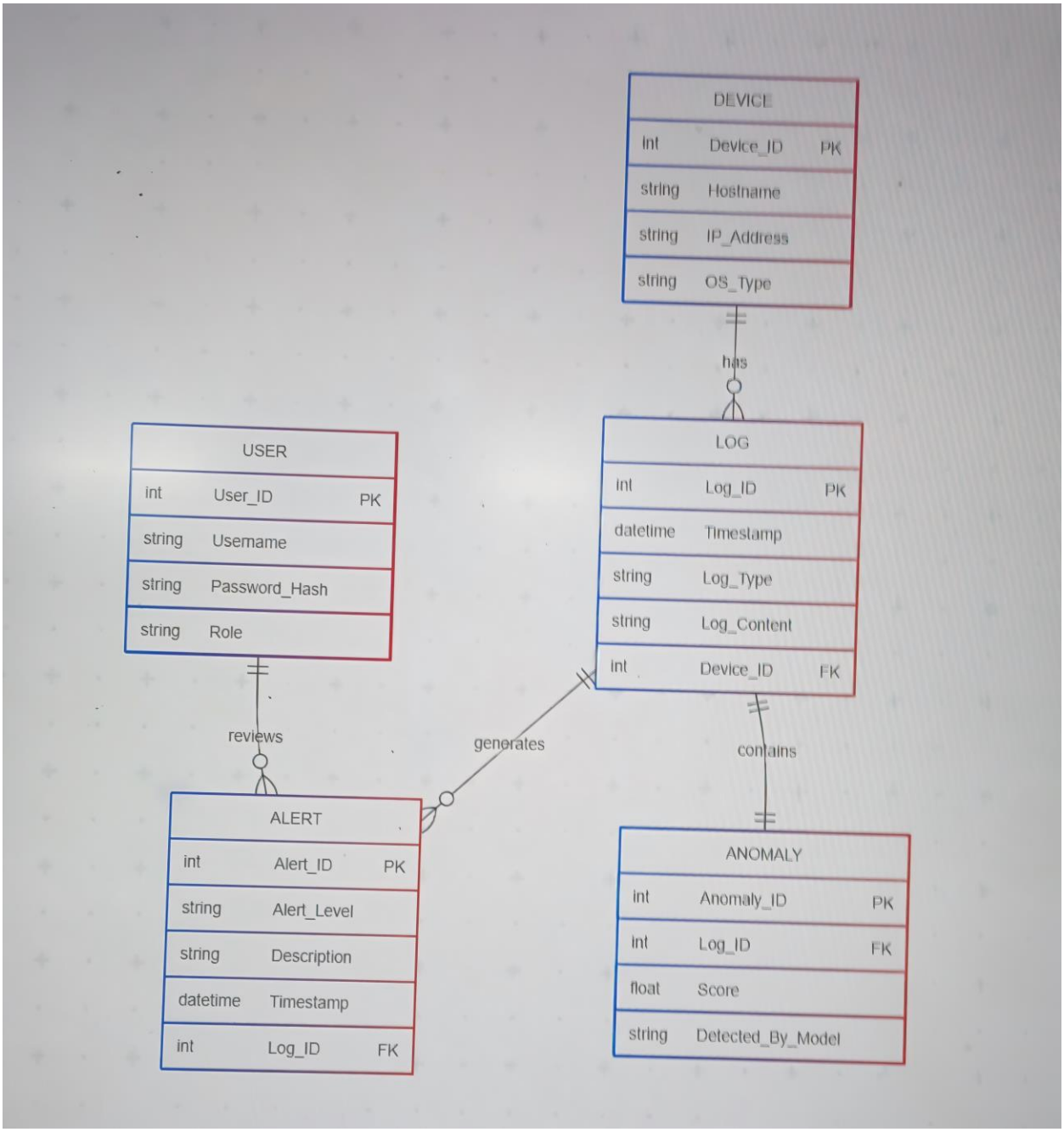

Figure 4 :ER Diagram

9. TIME SCHEDULE: -

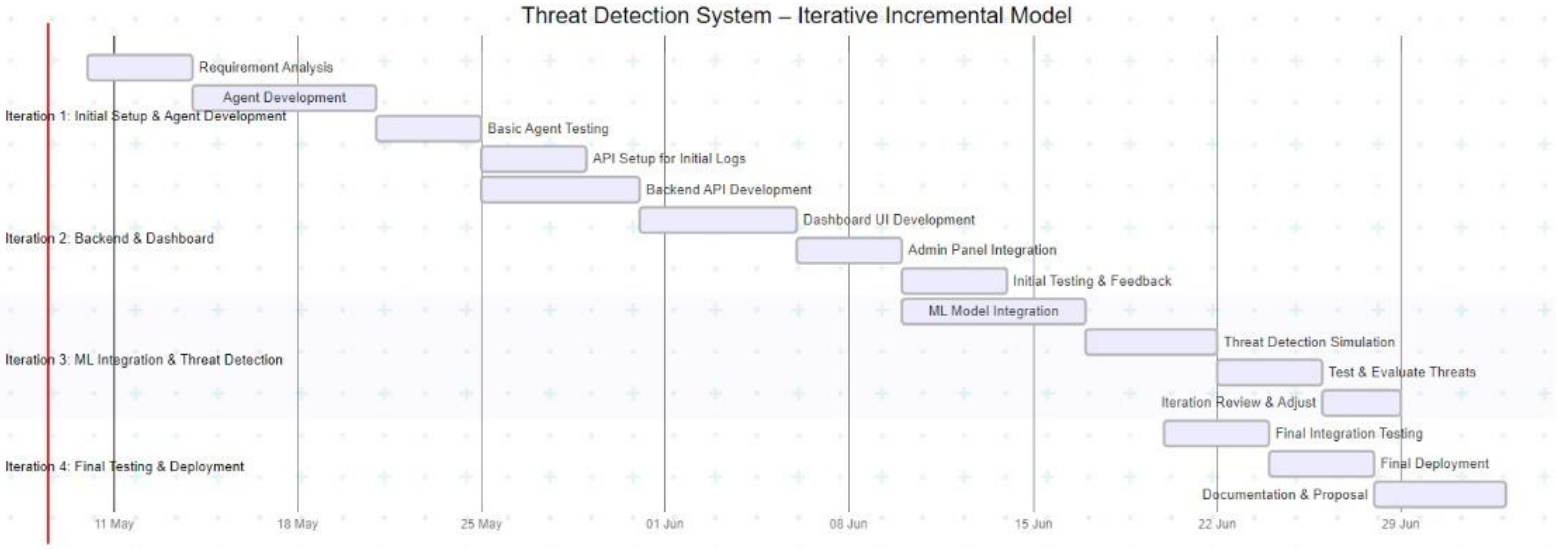

Fig 9.1 Gantt chart

10. CONCLUSION: -

Our project introduces a proactive and intelligent approach to cybersecurity through the development of a lightweight agent-based threat detection system. By continuously collecting system and user activity logs and analyzing them using machine learning techniques, the system can detect anomalies in real-time, allowing for quicker threat identification and response. This solution reduces reliance on manual monitoring and provides organizations with a scalable, cost-effective way to safeguard their digital infrastructure. With its modular design and automated alert system, our project demonstrates a significant step towards modernizing endpoint security and addressing evolving cyber threats.

11. BIBLIOGRAPHY: -

- Yadav, A. K. (2020). *Unsupervised Machine Learning for Anomaly Detection in Cybersecurity*. Journal of Computer Science and Technology, 38(2), 105–112

- Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting cybersecurity attacks in Internet of Things Using Artificial intelligence Methods: A Systematic Literature review. *Electronics*, *11*(2), 198. https://doi.org/10.3390/electronics11020198
- Kocyigit, E., Korkmaz, M., Sahingoz, O. K., & Diri, B. (2021). Real-Time Content-Based Cyber Threat Detection with Machine Learning. In *Advances in intelligent systems and computing* (pp. 1394–1403). https://doi.org/10.1007/978-3-030-71187-0_129
- Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of machine learning for anomaly based intrusion detection Systems in the CICIDS2017 dataset. *IEEE Access*, *9*, 22351–22370. https://doi.org/10.1109/access.2021.3056614
- Sheykhkanloo, N. M., & Hall, A. (2020). Insider threat detection using supervised machine learning algorithms on an extremely imbalanced dataset. *International Journal of Cyber Warfare and Terrorism*, *10*(2), 1–26. https://doi.org/10.4018/ijcwt.2020040101