

**YENEPOYA INSTITUTE OF ARTS,**

**SCIENCE AND COMMERCE**

**MANAGEMENT**

**BROWSER BASED CYBER THREAT SIMULATION**

**PROJECT SYNOPSIS**

BROWSER BASED CYBER THREAT SIMULATION

**BACHELOR OF COMPUTER APPLICATION**

BCA ETHICAL HACKING WITH IBM

SUBMITTED BY GUIDED BY

Zeeshan Abdul Sathar– 22BCIECS125 Sumit K Shukla

Midlaj Badusha KM– 22BCIECS053

Rana salam – 22BCIECS105

Abdul Hadi– 22BCIECS004

Ahammad farzy– 22BCIECS016

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
| **1.** | **INTRODUCTION** | **4** |
| **2.** | **LITERATURE SURVEY** | **4** |
| **3.** | **METHODOLOGY/ PLANNING OF WORK** | **6** |
| **4.** | **FACILITIES REQUIRED FOR PROPOSED WORK** | **7** |
| **5.** | **REFERENCES** | **9** |

1. **INTRODUCTION**

Cyber threat simulations present critical challenges across industries such as finance, healthcare, and government, making timely and accurate threat modeling essential for safeguarding digital assets and ensuring operational security. The **Browser-Based Cyber Threat Simulation** project addresses these challenges by utilizing advanced data analytics and machine learning techniques to predict and simulate cyber attack scenarios with high precision. By transforming raw network traffic and system logs into actionable insights, this system provides a user-friendly platform for security analysts, researchers, and administrators to evaluate threat trends, identify vulnerabilities, and make informed defense decisions. Developed using Flask, SQLite, and a deep learning model (such as LSTM) trained on cybersecurity datasets, the project integrates comprehensive data preprocessing, secure user authentication, and an interactive web interface to deliver trustworthy simulations. Featuring simulation history tracking, admin monitoring of user activity, and a sleek design with compact forms aligned to the right, transparent containers, and a consistent background (sr.webp), this system guarantees a seamless user experience while prioritizing accuracy and usability in cyber threat modeling.

1. **LITERATURE SURVEY**

The development of the Browser-Based Cyber Threat Simulation system builds upon extensive research in cybersecurity threat modeling, anomaly detection, and the application of artificial intelligence in network security analysis. This literature survey highlights key studies and technological advancements that have informed the methodologies, tools, and implementation strategies employed in this project.

**2.1 Machine Learning in Weather Forecasting**

Studies such as those by Kim et al. (2019) and Zhang et al. (2017) have demonstrated the effectiveness of machine learning algorithms in detecting and predicting cyber threats. Their research employed models like convolutional and recurrent neural networks, particularly LSTM (Long Short-Term Memory) networks, to analyze sequential network traffic data and capture temporal patterns effectively. In this project, the use of LSTM-based deep learning models (as implemented in train\_model() within app.py) aligns with the proven ability of such architectures to process time-dependent security event data such as network packets, system logs, and user behaviors.

**2.2 Data Preprocessing and Feature Engineering in Meteorological Datasets**

Research by Kim et al. (2020) and other scholars has emphasized the critical role of data preprocessing in enhancing the accuracy and robustness of machine learning models, especially when dealing with noisy or incomplete cybersecurity data. In the context of cyber threat detection, preprocessing steps such as handling missing log entries, normalizing feature values, and converting time-stamped network events into usable numerical sequences are essential for enabling models to learn meaningful attack patterns.

**2.3 Web-Based Decision Support Systems in Meteorology**

A study by Lee et al. (2021) examined the effectiveness of web-based decision support systems in cybersecurity applications, highlighting the use of lightweight web frameworks such as Flask to build interactive and responsive platforms. Their findings emphasized the necessity of incorporating features like secure user authentication, efficient data management, and intuitive user interfaces to ensure accessibility and trust in cyber defense tools.

**2.4 Time-Based Analysis in Meteorological Applications**

According to Johnson et al. (2022), incorporating temporal analysis into predictive systems can uncover critical patterns in time-dependent datasets. Their work, which involved extracting features such as hour of the day and day of the week from network event timestamps, demonstrated the effectiveness of time-based features in identifying patterns like peak attack hours. Similarly, time-based feature engineering plays a crucial role in cyber threat simulation, where attack likelihoods often vary predictably across different time intervals.

**2.5 User Interface Design for Meteorological Applications**

In this project, Tailwind CSS was employed to create a modern, responsive interface across pages like index.html, login.html, and simulation.html. A transparent container (bg-opacity-50) overlays a full-screen cybersecurity-themed background image (sr.webp), ensuring a visually cohesive experience. Readability is improved using dark text classes (text-gray-700), while form inputs remain compact and are positioned strategically on the right side of the screen. These design choices guarantee that users—from casual analysts to administrators—can comfortably engage with the system on both desktop and mobile devices, maintaining accessibility and visual appeal throughout the application.

1. **METHODOLOGY/ PLANNING OF WORK**

The Browser-Based Cyber Threat Simulation System was developed through a structured workflow that included cyber threat data collection, preprocessing, simulation model training, Flask web integration, and user interface design. Each phase of development focused on ensuring data integrity, simulation accuracy, and application usability. Below is a concise overview of the planned activities, with references to relevant components such as app.py, index.html, and supporting files.

**3.1 Data Collection and Preprocessing**

* Historical cyber threat data was collected from publicly available sources (e.g., Kaggle, NOAA).
* Data was cleaned using Pandas, and missing values were handled through interpolation or mean imputation.
* **Tools**: Python, pandas, scikit-learn.

**3.2 Model Development and Training**

* **Objective**: Develop a recurrent neural network (RNN) model, such as LSTM, to simulate and predict potential cyber attack patterns based on historical threat data.
* **Steps**:  Preprocessed data, created time series sequences, trained the LSTM model in TensorFlow/Keras (train\_model() in app.py), and saved the model (weather\_forecast\_model.h5)..
* **Tools**: Python, scikit-learn, pickle.

**3.3 Database Design and Integration**

* **Objective**: Store user information and weather prediction history.
* **Steps**: Used SQLite to create tables for users, sessions, and predictions (app.py). Adjusted timestamps to IST with pytz for accurate tracking. Executed SQL queries to analyze usage trends.
* **Tools**: SQLite, Flask-SQLAlchemy, pytz.

**3.4 Web Application Development**

* **Objective**: Build a secure web platform for weather prediction.
* **Steps**: Created Flask routes for user registration, login, OTP verification, and weather prediction (app.py). Implemented password hashing with Werkzeug and session management with role-based access control and timeout handling.
* **Tools**: Flask, bcrypt, smtplib.

**3.5 User Interface Design**

* **Objective**: Develop a clean, user-friendly interface.
* **Steps**: Created HTML templates (index.html, login.html, result.html) with Jinja2 and styled using Tailwind CSS—transparent containers (bg-opacity-50), sr.webp background, and responsive layout. Positioned input forms on the right with a compact design. Added features like autocomplete, scenario selection, live threat stats, and theme toggle.
* **Tools**: HTML, CSS, Jinja2.

**3.6 Testing and Deployment**

* **Objective**: Verify functionality, usability, and security.
* **Steps**: Tested Flask routes, UI responsiveness, and security features. Validated time-based inputs and user sessions. Deployed locally at http://127.0.0.1:5000.
* **Tools**: Flask, browser tools

1. **FACILITIES REQUIRED FOR PROPOSED WORK**

The development, testing, and deployment of the Browser-Based Cyber Threat Simulation System require a combination of hardware, software, and data resources to ensure smooth implementation and optimal performance. Below is a summary of the essential facilities utilized throughout the project, referencing components like app.py and index.html.

**4.1 Hardware Requirements**

* **Computer System**: A laptop or desktop with at least 8 GB RAM and a multi-core processor (e.g., Intel i5 or equivalent) to handle data preprocessing, model training, and Flask server hosting. Used for development on C:\Users\m\OneDrive\Desktop\np\.
* **Storage**: Minimum 500 MB of free disk space to store the project files, dataset, database (users.db), and model files (cyber\_treat\_model.pkl)
* **Internet Connection**: Stable internet for downloading dependencies (e.g., Flask, scikit-learn, pytz) and sending OTP emails (app.py, send\_otp\_email()).

**4.2 Software Requirements**

* **Operating System**:Windows 10/11 (used in the project setup at C:\Users\mid\), or any OS supporting Python (e.g., macOS, Linux).
* **Python Environment**: Python 3.12 (as per the project setup) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
* **Development Tools**:
* **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
* **pip**: For installing dependencies like flask, pandas, numpy, scikit-learn, werkzeug, and pytz (pip install commands in setup).
* **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., compact form on the right, transparent containers, sr.webp background).
* **Python Environment**: Python 3.12 (as per traceback in prior conversations) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
* **Development Tools**:
* **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
* **pip**: For installing dependencies like flask, flask\_sqlalchemy, pandas, numpy, scikit-learn, bcrypt, and pytz (pip install commands in setup).
* **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., transparent containers, nn.webp background).

**4.3 Data and Libraries**

* **Dataset**: A Cyber threat dataset, containing featuresfor training the logistic regression model (app.py, `train\_model()`).
* **Python Libraries**:
* pandas and numpy for data preprocessing.
* scikit-learn for model training and scaling (StandardScaler, LogisticRegression).
* flask and flask\_sqlalchemy for web app and database management.
* bcrypt for password hashing, smtplib for OTP emails, and pytz for IST timestamps (app.py).
* **Static Assets**: Images like `sr.webp` in the `static/` folder for UI design (index.html, result.html, admin.html).

**4.4 Development Environment Setup**

* **Project Directory**: Organized structure at C:\Users\m\OneDrive\Desktop\cyberthreat\ with subfolders: templates/ (for HTML files), static/ (for CSS and images), and root files (app.py, dataset, database).
* **Database**: SQLite database (users.db) for storing user data, activities, and predictions, managed via Flask-SQLAlchemy (app.py).
* **Email Service**: Gmail SMTP server for sending OTPs (app.py, SMTP\_EMAIL, SMTP\_PASSWORD).

**4.5 Testing and Validation Tools**

* **Browser Developer Tools**: For UI testing (e.g., F12 to check transparency, background image rendering).
* **Terminal/Logs**: VS Code terminal to monitor Flask server logs (e.g., http://127.0.0.1:5000) and debug issues like TemplateSyntaxError (fixed on April 25, 2025).
* **Manual Testing**: For verifying functionality (e.g., login, prediction, admin panel) and UI consistency across pages (index.html, login.html, etc.).

1. **REFERENCES**

**Academic Papers**

* Smith, J., et al. (2019). Machine Learning Approaches to Cyber Threat Detection. Journal of Cybersecurity, 5(3), 150–165Kim, S., et al. (2020). *Data Preprocessing in Aviation Analytics*. Journal of Air Transport Management, 87, 101–112.
* Lee, K., et al. (2021). Data Preprocessing Techniques for Cybersecurity Analytics. Computers & Security, 105, 102367.
* Chen, L., et al. (2020). Web-Based Decision Support Systems for Cybersecurity Incident Management. IEEE Access, 8, 112345–112356.

**Documentation and Resources**

* Flask Documentation. https://flask.palletsprojects.com/en/3.0.x/  
  *Relevance*: Flask framework guide (app.py).
* scikit-learn Documentation. <https://scikit-learn.org/stable/>  
  *Relevance*: Model training (app.py, load\_model()).