Introduction and project roadmap

Initiative: AI Big Thinkers Lab – Group 4: AI for Cybersecurity (Cyber Mapping Project)

Summary:
SalamaNetAI is an AI-driven cybersecurity initiative focused on developing advanced models for threat detection, anomaly monitoring, and phishing prevention. The project primarily targets African education and public sector systems, aiming to reinforce the digital infrastructure's resilience against modern cyber threats.

Objective:
To enhance the cyber readiness and security posture of Africa's education institutions and public agencies by integrating AI-powered security solutions tailored to regional challenges and infrastructure realities.

Project pitch-desk: **here**
Project Tracking : Trello
Github Repo: Here

📊 SalamaNetAI Project Roadmap

📌 Project Overview

SalamaNetAI is an AI-driven cybersecurity initiative targeting threat detection, anomaly monitoring, and phishing prevention for African education and public sector systems. The goal is to reinforce the cyber readiness of Africa's digital education infrastructure.

📅 Project Phases

📍 Phase 1: Ideation & Research

- Finalize problem statement and objectives
- Define system scope and target systems
- Identify relevant threats and risks in the African context
- Research existing AI cybersecurity solutions
- Identify and shortlist available public/open-source datasets

Deliverables:

- Problem Statement Document

- Scope Definition Document
- Dataset Sources List

📍 Phase 2: Design & Architecture

- Draft high-level system architecture
- Define AI model objectives (threat detection, anomaly detection, phishing classifier)
- Decide on data processing workflows
- Plan data storage, processing, and access control
- Select preliminary tech stack

Deliverables:

- Architecture Diagram
- AI Model Requirements
- Tech Stack Proposal

📍 Phase 3: Data Collection & Preparation

- Acquire datasets (open-source, simulated, or anonymized real data)
- Clean, preprocess, and label datasets
- Conduct exploratory data analysis (EDA)
- Prepare datasets for model training

Deliverables:

- Cleaned and annotated dataset
- EDA Report

📍 Phase 4: Model Development & Training

- Build AI models for:
    - Threat detection
    - Anomaly detection
    - Phishing URL/email classifier
- Train, validate, and test models
- Evaluate performance metrics

Deliverables:

- Trained Models
- Performance Evaluation Report

📍 Phase 5: System Development & Integration

- Develop platform backend and API services
- Create user interface/dashboard for security alerts and monitoring
- Integrate AI models into system
- Set up logging, audit, and notification systems

Deliverables:

- Functional prototype of SecuraNetAI
- Integrated AI services

📍 Phase 6: Testing & Security Hardening

- Perform penetration testing and vulnerability assessment
- Implement necessary security measures

- Conduct User Acceptance Testing (UAT)

Deliverables:

- Penetration Testing Report
- UAT Feedback Report

📍 Phase 7: Deployment & Monitoring

- Deploy system to staging/production
- Set up real-time monitoring and logging tools
- Document deployment and maintenance process

Deliverables:

- Deployed System
- Monitoring Dashboard
- Deployment Documentation

📊 Project Management Tools

- Notion or Trello for task tracking
- GitHub Projects for version control and issue tracking

📎 Next Steps

- Set up a Notion or Trello board structure
- Finalize tech stack proposal

End of Roadmap v1.0

The tech stack

⚙️ SalamaNetAI Proposed Tech Stack

👨‍💻 Backend / API:

- Python (FastAPI) — lightweight, async-ready, perfect for AI model serving.
- Django (if you need an admin portal + user management quickly)
  (Could even pair both — FastAPI for model APIs, Django for management portal)

🤖 AI/ML:

- PyTorch — future-proof, widely supported, and ideal for research-grade projects.
- scikit-learn — for baseline anomaly detection & classic ML models.
- Hugging Face Transformers — for phishing email classification via NLP.
- Jupyter Notebooks — for rapid prototyping of models.

📊 Database:

- PostgreSQL — reliable, scalable, and better JSONB support if storing log-style or anomaly events.
- TimescaleDB (PostgreSQL extension) — if anomaly monitoring needs time-series data.
- Redis — for caching model inferences / session data.

🛡️ Cyber Mapping / Threat Intelligence:

- OpenCTI or MISP (open-source threat intelligence platforms)
- Integrate via REST APIs for contextual African-specific threat mapping

📈 Dashboard / Frontend:

- React.js — fast, modular, future-proof.
- Tailwind CSS — lightweight, modern styling.

- Recharts / Chart.js — for threat visualization dashboards.

🐳 Deployment:

- Docker — containerize models & API services.
- Docker Compose — orchestrate services during dev/test.
- Ubuntu Server 22.04 LTS — target deployment environment.

☁️ Hosting (for PoC / MVP):

- Render.com or DigitalOcean App Platform — lightweight, fast deploys.
- GitHub Actions — for CI/CD.

🎯 Optional:

- Rust (if you want to build a super-fast threat detection agent for endpoint or network logs)
  Could be a phase 3 thing.

Summary:

📊 **Tech Stack Decision:**

Backend:

- **FastAPI** — for lightweight, asynchronous REST APIs
- **Celery + Redis** — for background jobs, scheduled tasks
- **SQLAlchemy (PostgreSQL)** — relational DB for logs, users, AI scan results

AI/ML:

- **TensorFlow 2.x** — for deep learning models (anomaly detection, phishing classifiers)
- **Hugging Face Transformers** — for NLP-based phishing and log anomaly detection
- **scikit-learn** — for lightweight ML baselines and quick anomaly detection

Frontend:

- **React (with Vite)** — fast modern frontend
- **Tailwind CSS** — clean, responsive UI framework
- **React Query** — handles API state management gracefully

DevOps:

- **Docker** — containerized services
- **GitHub Actions** — for CI/CD
- **Render.com** — for affordable cloud hosting with scaling (migratable later to DigitalOcean or AWS)

📌 **Notes:**

- Light, scalable, and AI-ready.
- Focus on modular services for easy AI model plug-ins.
- Can pivot cloud providers in later phases if scaling demands it.

✅ **Status:** Locked as of **June 7, 2025 ??**

📌2️⃣What is ultra-low-latency C++/Rust with custom AI serving and what does *custom* mean here?

Explanation:
When AI models need to respond in *real-time at sub-millisecond speeds* (think intrusion

detection systems at datacenter firewalls, fraud detection on credit card networks, real-time gaming AI), Python frameworks can be too slow because of:

- **The Global Interpreter Lock (GIL)**
- **Python's single-threaded nature for many AI operations**
- **Overhead from general-purpose servers (like FastAPI)**

Ultra-low-latency serving =
➡️ AI model inference engines written directly in C++ or Rust
➡️ Served via purpose-built servers like TensorRT (for NVIDIA GPUs) or custom REST/gRPC services built in Rust/C++

Custom here means:

- **Not relying on off-the-shelf AI libraries/frameworks**
- **Building your own inference serving pipeline optimized for speed**
- **Tailoring memory management, threading, batching, etc. yourself**

Why we're not doing this now:
That's overkill for MVP and African edu-sector systems which don't need 0.5ms response times. FastAPI + TensorFlow/Hugging Face + Docker is plenty for our threat detection dashboards and AI-assisted alerting.

Datasets sources

Here are the direct links and access paths for each dataset, formatted for easy copy-pasting into your Google Doc or terminal scripts:

📧 Email-Based Phishing Datasets

1. Phishing Email: 11 Curated Datasets (Champa et al.)

Zenodo link:
https://zenodo.org/records/8339691

Hit "Download" for the ZIP of CSV files.

---

2. Kaggle / Hugging Face “Phishing Email Dataset”

Hugging Face:
https://huggingface.co/datasets/zefang-liu/phishing-email-dataset

Original Kaggle page:
https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset

---

3. compiled-phishing-dataset (René Mello)

Hugging Face:
https://huggingface.co/datasets/renemel/compiled-phishing-dataset

---

4. Curated Seven Phishing Email Datasets (Figshare)

Figshare link:

https://figshare.com/articles/dataset/Curated_Dataset_-_Phishing_Email/24899952

---

5. Phishing Validation Emails (Miltchev et al., 2024)

Zenodo link:
https://zenodo.org/records/13474746

---

6. GitHub “Phishing-Email” Dataset

Typically available in the repository cloned from GitHub (search “Phishing-Email dataset GitHub”). Let me know if you want the exact repo—happy to fetch the link.

📱 SMS/Smishing Dataset

Mendeley Data – SMS Phishing Dataset:
https://data.mendeley.com/datasets/f45bkkt8pr/1
(5,971 SMS labeled as ham, spam, smishing)

Example Bash Snippets to Download (for Google Docs users)

```
# 1. Zenodo 11-dataset ZIP
wget https://zenodo.org/record/8339691/files/Phishing_Email_11_Curated_Datasets.zip

# 2. Hugging Face CLI for compiled-phishing-dataset
pip install huggingface_hub
huggingface-cli download renemel/compiled-phishing-dataset

# 3. Zenodo Validation Emails
```

wget https://zenodo.org/record/13474746/files/phishing-validation-emails.csv

---

You can paste the links into your Google Doc as-is, or use the script blocks if you’re comfortable running terminal commands. Let me know if you'd like me to draft the Google Doc with clickable URLs and download instructions built in!

Datasets:

Datasets:

11:18 (1 hour ago)

Recommended Steps for Roberto

1. **Download/Fetch PhishTank URLs** (via API or scraping)
2. **Combine with Benign URLs** from another source
3. **Extract Features**:
    - Length, subdomains, TLD, use of IPs
    - Character patterns, suspicious words
    - WHOIS data (optional)
4. **Train ML/DL Models**:
    - Logistic Regression, Random Forest
    - CNNs or LSTMs for sequence modeling of URLs

Why Roberto *Can* Use PhishTank Data

- **Openly Available:** PhishTank provides public access to phishing URLs submitted and verified by a community.
- **Regular Updates:** New URLs are added constantly, which helps keep the model current.
- **Good for Real-World Detection:** The URLs are actual phishing links seen in the wild.
- **Binary Labels:** URLs are labeled as *phishing* or *not verified*, making them suitable for classification tasks.
- For **NLP models**, use email body and subject.
- For **URL-based detection**, tokenize URL characters or use word embeddings.
- Combine **multiple datasets** for better generalization.
- Use tools like `BeautifulSoup` for HTML cleaning and `whois`, `tldextract`, and `urlparse` for feature extraction

Links

https://archive.ics.uci.edu/ml/datasets/phishing+websites
Source: https://monkey.org/~jose/phishing/
https://www.phishtank.com/

AI/ML Models Selection and fine tuning

Phishing Model Selection and Fine-tuning

We finetune pre-trained model and use as classifier.

The dataset: https://huggingface.co/datasets/zefang-liu/phishing-email-dataset

The link for the mode prototypel: \
1. https://huggingface.co/BEE-spoke-data/smol_llama-220M-GQA-fineweb_edu
2.https://huggingface.co/ElSlay/BERT-Phishing-Email-Model

Pipelines:

- Text /classifier
- URL scanner
- Malware Scanner for attached files

Tab 6