Part 1 – Planning:

Define the problem you are solving.

In large organizations, admin UIs to create user accounts, implement bans, and reset passwords are extremely common. Our current product improves the UI of an open-source pre-built auth server. However, organizations often have to manage many user accounts, both internal and external. This can sometimes lead to many accounts that are rarely used, expired, or “stale.”

These accounts can later become **vulnerabilities** due to their low use and rarely reset passwords. They also increase **database space usage,** saving companies thousands of dollars if properly identified and removed. To put it simply, organizations need a way to properly identify, analyze, and obtain metrics on “stale” user accounts (both internal and external) and methods of handling them (nudging, banning, deleting, etc.).

Explain how the AI component enhances your project (e.g., automation, decision-making, prediction, personalization, etc.).

To solve this problem, AI can be used to identify specific accounts that have “qualities” that can make them vulnerable. These could be easy-to-identify things such as a stale password (a password that hasn’t been reset in a long time) or perhaps even more difficult to identify things such as multiple user account creations from a single IP address, or with all the same user data (address, etc), or some other pattern that might be hard to identify and pre-code for.

We can train our in-house AI against sample data of user accounts, their logins, user information, etc and expect to receive a model that can accurately predict stale user accounts, spam user accounts, and other vulnerabilities or sources of excess space usage.

In other words, our AI will be used for automated detection of such accounts and recommendations on decisions to be taken to neutralize the problem.

Selection Criteria

1. Base Functionality
    a. A suitable AI model should be able to analyze user accounts and login information to predict which accounts are stale, spam, etc.
2. Performance & Scalability
    a. Since this AI helps reduce database space usage and prevents vulnerabilities, it *itself* must be fast and efficient.
    b. It must also scale to millions of users in a database effectively and be able to analyze just as accurately despite the increased scale.

Potential AI Models and Frameworks

1. Supervised Learning models such as Decision Trees can be used to classify stale or spam accounts.
2. Unsupervised Learning models involving Clustering can be used to group accounts with similar activity patterns based on logins and IP addresses and such.
3. Deep Learning models such as Recurrent Neural Networks or Transformers can be used for more complex pattern recognition in user logins..

Datasets and Training

In terms of datasets needed to train the AI, we must obtain datasets that contain user logins and user accounts that span a wide variety of potential stale and spam scenarios. These should include accounts that have all been created with a specific IP address or with the same names and addresses or the same phone number. User activity should also be included to train the AI in identifying accounts that start normal but eventually progress towards stale or spam.

Ethical Considerations

The AI model must not have biases in its training or functionality that classify certain demographics of users as spam or stale when they are not so. The AI must also maintain user privacy. Although user logins and data may be fed to the AI, it must not disrupt user privacy and should ensure that all user data is kept isolated and in-house, following user privacy rules and regulations.

Feature Justification

a12n-server as of now aims to provide an all-encompassing, OAuth-based authentication server with an Admin UI. Following OpenID Connect standards and OAuth 2.0 standards, it was built to ensure that end users do not have to worry about setting up an authentication server themselves. Security is, of course, one of the top priorities, which is why methods such as JWT authentication are used.

However, one part of this mission that can be improved is the identification of accounts existing in the database that are stale or contribute as spam. By identifying these accounts, client security and resource use can be optimized, which is following the initial mission of **a12n-server**: to provide an authentication server that saves time and improves security for developer users. Thus, it contributes to the mission of the original app.

Part 2 – Impact Analysis:

The final step is to analyze the impact of the improvement on the performance of the product. Your report should include details on:

2.1 Improvement of Products

1. Automated analysis of Unused Accounts
    a. AI models can analyze logs, access reports, usage patterns, and meta data to determine if an account is truly forgotten or inactive in real time.
    b. The system can then flag these accounts for deactivation or review, reducing security risks and account slumps.

2. Conscious Prioritization of Vulnerability Reports
    a. Machine learning algorithms assess severity by combining multiple data sources (public vulnerability databases, internal testing results, exploit attempts in the wild) to create a risk score for each vulnerability.
    b. This reduces the time required to identify critical issues, ensuring faster and more focused remediation efforts.

3. Resource Optimization & Continuous Improvement
    a. Automated suggestions enable security teams to focus on higher-priority tasks.
    b. Feedback loops from confirmed issues help the AI to improve continuously (e.g., retraining on newly discovered patterns or vulnerabilities).

2.2 Improvement Testing

1. Unit Tests
    a. Each component of the AI pipeline (e.g., data preprocessing, feature extraction, and scoring algorithm) can be tested in isolation.
    b. Ensure that data ingestion from logs and vulnerability databases works as expected.
    c. Mock or stub external services to validate that the system handles a variety of responses (e.g., missing data, malformed data, large data volumes).

2. Accuracy Evaluation
    a. Use historical data and known vulnerabilities/unnecessary accounts as a labeled dataset.
    b. Compare the model's predictions (e.g., flagged accounts, prioritized vulnerabilities) to the actual ground truth.
    c. Track scenarios where the AI incorrectly labels an account or vulnerability. This analysis helps refine thresholds and identify weaknesses in the model.

3. A/B Testing
    a. Split the workload between a "control" group (teams using the previous manual/rule-based approach) and a "test" group (teams using the new AI-driven approach).

   b. Monitor metrics like mean time to identify a threat, number of issues flagged incorrectly, and time saved per security employee.
   c. Gather direct input from security personnel about their experience using the AI system and ask questions about the improvements it provides.

2.3 Trade-offs

1. Performance and Latency
   a. Complex AI models may introduce some latency in generating recommendations, especially if large amounts of data are analyzed in real time.
   b. Solutions could include optimizing model architecture or caching frequently accessed results to reduce response times.

2. Resource Consumption
   a. Introducing AI might require new data pipelines, ML frameworks, or microservices, increasing operational complexity and maintenance overhead.
   b. The cost and energy usage need of having the agentic process always running needs to be balanced against the security benefits.'

3. Security and Data Privacy
   a. Centralizing logs and account details for AI analysis can introduce new data privacy and security considerations.
   b. Compliance with regulations might require additional anonymization or secure storage solutions.

Models/Technologies we plan on using

1. For Anomaly Detection: Isolation Forest (https://www.datacamp.com/tutorial/isolation-forest)
   a. Works well for identifying outliers
   b. It's a good starting point for the set of data that our application stores and uses
2. For Vulnerability Report Analysis: Transformer Model, specifically BERT (https://research.google/pubs/bert-pre-training-of-deep-bidirectional-transformers-for-language-understanding/)
   a. They will help classify vulnerability reports into severity buckets.
   b. Extract important entities (e.g., affected system, CVE references, software version).
   c. Summarize lengthy vulnerability descriptions into action items or a prioritized list.
3. Model/Tech Stacks: Python + scikit-learn + PyTorch / TensorFlow
   a. Use scikit-learn for anomaly detection (ex. Isolation Forest).
   b. Use PyTorch or TensorFlow for training / fine-tuning a transformer model on vulnerability data.