Yanez Multimodal Inorganic Identities Dataset (MIID) Subnet (ي)

Phase 1

V1.0 – April 2025

Phase 1: Threat Scenario Query Execution & Initial Deployment

Phase 1 introduces the Threat Scenario Query System, enabling validators to reward miners for generating execution vectors targeting name-based evasion tactics such as phonetic manipulations and orthographic variations.

How to query a subnet miner?

Query Structure

1- Seed Names Are Predefined

a. The validator generates and provides a list of seed names using internal logic (e.g., via faker).
b. Miners are **not** responsible for creating or selecting seed identities.

2- Dynamic Name Variation Generation (M Variations per Name)

Each seed name requires M variations, which miners must generate based on linguistic transformation rules rather than producing random outputs.

- Phonetic (Sound Similarity) Transformations
  - The request specifies what percentage of M variations should exhibit phonetic similarity to the original seed name.
  - Miners must create phonetic similar variations while following predefined similarity levels: Light, Medium, Far
- Orthographic Transformations
  - The request specifies what percentage of M variations should introduce orthographic modifications (e.g., script changes or spelling variations).

- Miners must apply script-based transformations based on similarity level such as edit-based metrics with predefined similarity levels, Light, Medium, Far

How This Prevents Simple Lookups and Functions Like a Hash

Our approach transforms straightforward name generation queries into complex linguistic and computational challenges akin to a "hash function" in cryptography. This ensures that resolving a query requires advanced NLP models rather than simple database lookups, preventing brute-force retrieval and ensuring proper generative synthesis.

1- Dynamic Name Seed Encoding
   a. Instead of directly providing names, Yanez applies attribute-driven transformations, ensuring that each query is unique, computationally complex, and resistant to lookup-based retrieval.
   b. Encoded Query Structure: Rather than explicitly listing names, queries define abstract constraints (e.g., language distribution, variation ratios).
   c. Contextual Name Synthesis: Miners must generate names dynamically while preserving linguistic characteristics, ensuring the results are not retrievable from a pre-existing dataset.
2- Layered Variation Generation
   a. Each query defines M variations per name, but instead of following direct mappings, miners must derive phonetic and orthographic transformations algorithmically based on similarity constraints.
   b. Non-Linear Similarity Scoring: Variations are generated across multiple levels of similarity (Light, Medium, Far), requiring miners to compute variations dynamically rather than applying simple letter substitutions.
   c. Adaptive NLP Models: The system forces miners to apply transformation rules rather than relying on pre-stored lists.
3- Attribute-Based Constraint Encoding
   a. Instead of explicitly stating how many phonetic or orthographic variations must be generated, the system encodes constraints into abstract feature representations that miners must decode before producing variations.

Why This Matters

By enforcing complex, NLP-driven transformations, Yanez ensures that subnet miners cannot rely on existing name databases, making the system:

- More Secure – Prevents direct retrieval from known datasets, reducing exploitation risks.
- More Adaptable – Enables context-aware name generation, supporting multiple languages and cultural variations.

- More Scalable – Works across different identity frameworks (AML testing, AI research, gaming, VR, decentralized ID verification).

Query Examples:

- "Given the name 'Mohammed,' generate 10 variations with the following phonetic constraints: 40% should have light phonetic shifts, 40% should have medium phonetic shifts, and 20% should be far variations."

- "Using the seed names [Ali, Fatima, Ivan, Chen], create 8 variations per name: 50% phonetic alterations (edit distance ≤2) and 50% script-based transliterations. Ensure no repeated patterns from existing databases."

How should a subnet validator evaluate the subnet miner response?

- Check the seed names if there are the same as the ones that have been sent in the request or generated as it has been requested in the query (right language/script distribution)
- Compute the phonetic and edit distance of the variations with respect to their seed names
- Compare the results with the requested distribution in the query (Attribute-Based Constraint Encoding)
- Scores the synthetic names if they meet the request in the query
    - Relevance to the request, based on the asked percentage and the transformation rules
    - Novelty: how the received name variations compared to similar generated names from the same seed if they exist in the database (in development for phase #)
    - Potential issue: measure if any of the created variations is closer to another name in the database
- Collect the data, push them to the hugging face as a file, and then concatenate them to the database.
    - If a miner submits too frequently, the validator may increase the file threshold accumulation limit.

The subnet validator ensures that miners generate high-quality, diverse, and compliant name variations while adhering to the constraints defined in the query. The evaluation

process consists of multiple validation steps, ensuring the generated synthetic identities' accuracy, compliance, and integrity.

- Validate Seed Names & Query Compliance
    - Ensure that the received seed names match the ones provided in the query (if specified) or conform to the requested language/script distribution when miner generated.
    - Confirm that the generated names adhere to the script constraints (e.g., Arabic, Cyrillic, Latin) as requested in the query.
- Compute Phonetic and Orthographic Distance
    - Measure the phonetic distance of each generated variation relative to its seed name, ensuring the correct degree of similarity (Light, Medium, Far) is applied based on the query constraints.
    - Compute the Levenshtein distance (or similar metric) to quantify how different each generated name is from the seed name, validating that variations align with query-defined transformation levels.
- Attribute-Based Constraint Matching
    - Compare the actual distribution of phonetic and orthographic transformations against the requested percentage breakdown specified in the query.
    - Reject responses that overuse simple modifications or fail to produce the diversity required in the variation generation process.
- Data Storage
    - Collect and push validated synthetic data to Hugging Face as structured files.
    - Periodically concatenate validated names into a larger database for future benchmarking and training purposes.
    - If a miner submits too frequently, the validator may increase the file threshold accumulation limit.

How should a subnet miner be rewarded for its response?

Miners in the Yanez MIID subnet are rewarded based on how well their generated **execution vectors** (e.g., name variations) match the specified threat scenario constraints. The reward mechanism evaluates each response across multiple quality dimensions and penalizes incomplete or misaligned outputs.

Reward Calculation Pipeline

For each miner, the final reward is calculated as:

$$\text{Final Reward} = \text{Average Quality Score Across Names } (Q) \times \text{ Completeness Penalty } (\mathrm{P})$$

Quality Score per Name (Q_i)

$$Q = \frac{1}{N}\sum_{i=1}^{N} Q_i$$

The quality of variations generated for each seed name is computed using a weighted sum of four factors:

$$Q_i = w_s \cdot S + w_c \cdot C + w_u \cdot U + w_l \cdot L$$

Where:

- S: Combined **similarity score** (phonetic + orthographic)
- C: **Count score**, rewarding the correct number of variations
- U: **Uniqueness score**, penalizing duplicate outputs
- L: **Length score**, rewarding reasonable variation lengths

With default weights:

$$w_s = 0.6, w_c = 0.05, w_u = 0.3, w_l = 0.05$$

Quality Score Components

A) Similarity Score (S)

This measures how closely the generated variations align with the phonetic and orthographic similarity constraints specified in the query.

$$S = \frac{S_{\text{phonetic}} + S_{\text{orthographic}}}{2}$$

Similarity is computed using:

• **Phonetic**: via Soundex + Levenshtein

• **Orthographic**: via normalized Levenshtein distance

Each sub-score is calculated using **level-based binning**, based on how well the distribution of generated variations matches the target distribution over similarity levels (Light, Medium, Far).

$$S_t = \sum_{l \in \{\text{Light, Medium, Far}\}} w_l \cdot \min\left(\frac{\text{actual}_l}{\text{target}_l}, 1.0\right)$$

Where:

- w_l: Target weight for level l
- actual_l: Number of variations in level l
- $\text{target}_l = w_l \cdot |V|$: Expected count in level l

Phonetic Ranges:
Light: 0.8–1.0, Medium: 0.6–0.8, Far: 0.3–0.6

Orthographic Ranges:
Light: 0.7–1.0, Medium: 0.5–0.7, Far: 0.2–0.5

The similarity distribution from the query (e.g., 50% Medium, 30% Far, 20% Light) is compared against actual distribution from the miner's output.

B) Count Score (C)

Measures whether the number of returned variations V is close to the expected count E.

A tolerance margin $\varepsilon = 0.2(20\%)$ is used.

$$C = \begin{cases} 1.0\,, & if\ |V\ -\ E|\ \leq 0.2\ \cdot E \\ 1.0 - \min\left(1.0, \dfrac{|V-E|}{E}\right), & Otherwise \end{cases}$$

C) Uniqueness Score (U)

Rewards miners for returning distinct variations and penalizes duplication.

$$U = \frac{\text{Number of Unique Variations}}{\text{Total Variations}}$$

D) Length Score (L)

Measures how proportionally long each variation is compared to its seed. Extremely short or long variations reduce the score.

$$L_v = \min\left(\frac{|v|}{|o|}, \frac{|o|}{|v|}\right)$$

$$L = \frac{1}{|V|}\sum_{v \in V} L_v$$

Where:

• $|v|$: Length of the variation

• $|o|$: Length of the original seed name

A high Length Score encourages **natural-looking, proportionally adjusted** variations that are realistic and useful for testing real-world screening systems. It helps filter out edge cases and improves dataset quality.

Completeness Penalty (P)

$$\text{Penalty} = \min\left(0.9, P_{\text{extra}} + P_{\text{missing}}\right)$$

The penalty is **additive penalty** logic, capped at 0.9 total.

$$\text{Completeness Penalty (P)} = \max(\ 0.1,\ 1.0\text{- Penalty})$$

A **minimum floor of 10%** of the total reward is maintained to avoid zeroing out valuable partial submissions.

Miners are penalized for:

- **Missing Seed Names**:

$$P_{\text{missing}} = \min\left(0.9, 0.2 \cdot N_{\text{missing}}\right)$$

    - 20% penalty per missing name
    - Capped at 90% total penalty
- **Extra Names Not Requested**:

$$P_{\text{extra}} = \min(0.7, 0.1 \cdot N_{\text{extra}})$$

    - 10% penalty per unexpected name
    - Capped at 70% total penalty