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

Phase 1

V1.1 – April 2025

Phase 1: Threat Scenario Query Execution & Initial Deployment

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

In addition to phonetic and orthographic similarity constraints, queries may specify a set of explicit transformation rules that a subset of the generated variations must comply with. These rules could include, for example, character doubling, specific typos, homoglyph substitutions, or language-specific patterns.

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
    - Rule-based:
        - The query may contain a rule-based section specifying required transformation rules and a minimum coverage percentage for these rules.

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. At least 3 variations (30%) must apply one of these rules: double consonant, homoglyph substitution, or vowel omission."
- "Using the seed names [Ali, Fatima, Ivan, Chen], create 8 variations per name: 50% phonetic alterations (edit distance ≤2) and 50% script-based transliterations. At least 2 variations per name (25%) must use one of the following rules: character swap, accent removal, or repeated character insertion. Ensure no repeated patterns from existing databases."

How should a subnet validator evaluate the subnet miner response?

- Validate Seed Names & Query Compliance:
  - Ensure that the seed names in the miner's response exactly match those provided in the query, or, if names are generated, verify that they conform to the requested language/script distribution and adhere to script constraints (e.g., Arabic, Cyrillic, Latin) as specified.
- Evaluate Attribute-Based and Rule-Based Constraints:
  - Compute the phonetic distance and edit distance (e.g., Levenshtein) for each variation relative to its seed name.
  - Confirm that the actual distribution of similarity levels (Light, Medium, Far) in the generated variations matches the requested breakdown in the query.
  - If the query includes explicit rule-based requirements, verify that at least the required percentage of variations match one of the requested transformation

rules (e.g., double consonant, homoglyph substitution), and assess coverage across all specified rules.

- Score the Synthetic Names:
  - Assess relevance to the query, based on requested attribute and rule-based percentages and transformation rules.
  - Evaluate novelty: check if generated variations are sufficiently different from previously generated names for the same seed (future phase).
  - Identify potential issues: determine if any generated variation closely resembles a name in the exclusion database.
- Enforce Diversity and Constraint Compliance:
  - Reject responses that fail to meet phonetic, orthographic, or rule-based constraints, overuse trivial modifications, or lack required diversity in transformations.
- Data Storage and Logging:
  - Save all validator results, including queries, responses, and computed rewards, locally as structured JSON files for traceability and auditability.
  - After validation and signing, upload results as JSON payloads to a centralized, authenticated backend API endpoint for aggregation, benchmarking, and further processing.
  - Log detailed metrics and rewards to Weights & Biases (wandb) for experiment tracking, analysis, and reproducibility.

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:

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

* Completeness Penalty (P) a multiplier, not a penalty in the strict sense; 1 means no penalty, lower values apply a penalty.

Quality Score per Name (Q_i)

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

For each **seed name**, a "variation quality" is computed:

$$Q_{\text{i}} = (1 - w_{\text{rule}}) \times Q_{\text{base}} + w_{\text{rule}} \times Q_{\text{rule}}$$

Where:

- w_{rule} :Rule compliance weight
- Q_{base}: Base quality score (from similarity, count, uniqueness, length)
- Q_{rule} : Rule compliance score

Base Quality Score per Name (Q_{base}):

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

$$Q_{base} = w_s \cdot S + w_c \cdot C + w_u \cdot U + w_l \cdot L$$

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 in lengths

With default weights:

$$w_s = 0.6, w_c = 0.15, w_u = 0.1, w_l = 0.15$$

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).

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

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.

Combining First and Last Name Parts

When a name has multiple parts (e.g., first and last), each part gets its own base score; then they are combined using weights:

$$Q_{\text{base}} = w_{\text{first}} \cdot Q_{\text{first}} + w_{\text{last}} \cdot Q_{\text{last}}$$

Each name part's weight is set proportional to its character length within the full name, then adjusted by up to ±20% random variation (with deterministic seeding), and finally normalized so all weights sum to one. In this version, we will utilize only two weights for the first name and last name.

Rule Compliance Score (Q_{rule}):

The Rule Compliance Score, Q_{rule}, is the product of a quantity score, q , measuring how many unique variations meet the rule-based requirements compared to the target percentage, and a diversity factor, d, the proportion of requested rules that are covered by at least one variation. This structure ensures that the miner's output not only meets the minimum requested compliance but also distributes compliance across the full set of target rules, maximizing both coverage and diversity.

$$Q_{\text{rule}} = q \times d$$

$$q = \begin{cases} 0 & if\ V_{comp} = 0 \\ \frac{V_{comp}}{E} & if\ V_{comp} \leq E \\ 1.5 - 0.5\ \frac{V_{comp}}{E} & if\ V_{comp} > E \end{cases}$$

Where:

- V_{comp}: unique variations matching any target rule
- E : expected compliant count

$$d = \frac{R_{\text{met}}}{R}$$

Where:

- R_{met} : number of distinct target rules satisfied by at least one variation
- R : total number of target rules

Completeness Penalty ($\boldsymbol{P}$)

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

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

Completeness Penalty (P) = max(0.1, 1.0- 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