🧩Project Overview — TEE-Verified Phishing Analyzer (Solana-Aware, ASI-Native)

TL;DR

A privacy-preserving **security agent system** that detects and classifies suspicious **URLs, messages, emails, or Solana transactions** as **Safe / Unsafe** with **explainable evidence** and optional **TEE attestation**. Verdicts are cryptographically signed, logged to Solana for transparency, and accessible through **ASI:One** via **uAgents**.

🎯Vision

Move beyond static phishing blacklists to a **verifiable, explainable, decentralised threat intelligence layer**. Each agent in our system acts as a trusted, autonomous security oracle capable of reasoning, explaining, and verifying its conclusions.

Architecture Overview

Core Components

- **Go Analyzer Service**
- Deterministic feature extraction and rule evaluation.
- Performs static and contextual checks for phishing, social engineering, and wallet-draining patterns.
- Signs each `report.json`; optional integration with **Nitro Enclaves** for attestation.

- **Python uAgents Layer**

- Lightweight agent wrappers implementing **Intake**, **Analyzer**, **Referee**, and **Onchain** roles.
- Handles Chat Protocol, message routing, and Agentverse registration.
- Optionally loads and executes **MeTTa rules** for explainable scoring.

- **Solana On-chain Logger**

- Minimal **Anchor program** storing report hashes (`sha256(report.json)`), verdicts, and timestamps.
- Provides a public, tamper-evident record of verified analyses.

Agent Roles

Agent	Role	Description
`IntakeAgent`	Entry point	Receives input via Chat Protocol (URL, text, email, Solana tx).

Agent	Role	Description
`AnalyzerAgent`	Core worker	Calls Go service `/analyze` with snapshots; returns signed verdicts.
`RefereeAgent`	Verifier	Validates signatures and attestations; ensures code integrity.
`OnchainAgent`	Logger	Pushes report hashes to Solana smart contract.

Optional Frontend

A **Next.js dashboard** providing: - Paste box for testing inputs. - Verdict cards (Safe / Unsafe / Severity). - “Why?” section showing fired MeTTa rules and evidence. - "Verify" button for signature/attestation check. - “Log to Solana” button.

Core Detection Logic

Artifact types supported: - **URL / Webpage**: punycode domains, fake subdomains, redirects, wallet-drainer JS, forms posting to unknown endpoints. - **Text / Message**: urgency language, seed phrase requests, impersonation patterns. - **Email**: SPF/DKIM/DMARC checks, mismatched display names, suspicious reply-to headers. - **Solana Transactions**: risky SPL Token instructions (`Approve`, `SetAuthority`, `Transfer`, `CloseAccount`), non-allowlisted program IDs, sweeping patterns.

Outputs

```
{
  "verdict": "safe | unsafe | inconclusive",
  "severity": "low | medium | high | critical",
  "signals": [ {"id": "punycode-domain"}, {"id": "walletconnect-script"} ],
  "rules_fired": ["critical_wallet_drainer"],
  "ruleset_hash": "sha256:...",
  "signature": "...",
  "attestation": {"type": "mock | nitro", "measurement": "..."}
}
```

🌴TEE (Trusted Execution Environment)

- Developed initially with **QEMU guest ↔ host over vsock** (mock attestation).
- Final build optionally runs in **AWS Nitro Enclave**.
- **Inside enclave:** deterministic scoring + signing; emits Nitro Attestation Document binding `{EIF measurement, ruleset_hash, nonce}`.
- **Outside enclave:** verifies attestation and report signatures.

🧩 Solana Integration

- **Instruction analysis**: part of core phishing logic.
- **Anchor program** (`security_log`):

```
pub struct LogEntry { report_hash: [u8; 32], verdict: u8, timestamp: i64 }
```

- Enables anyone to verify analysis results publicly on-chain.

Tech Stack Summary

Layer	Language	Function
Analyzer Service	Go	Fast static checks, signing, (TEE-ready).
Agent Layer	Python	uAgents, Chat Protocol, MeTTa integration.
Frontend	TypeScript / Next.js	UI + wallet integration.
On-chain	Rust (Anchor)	Solana log contract.

🕹️ 10-Day Plan

Day	Goal
1–2	Define schemas, spin up Go service, scaffold uAgents.
3–4	Implement phishing & Solana checks, add rule evaluation.
5	Connect uAgents → Analyzer → Referee; manifest on Agentverse.
6	QEMU mock enclave; signature flow end-to-end.
7	Anchor contract + Solana logging.
8	Test corpus, determinism & reliability checks.
9	Documentation, badges, agent addresses.
10	Demo polish + video + optional Nitro integration.

⛳ Demo Flow

1. Paste suspicious artefact → agent responds with verdict and severity.
2. Click **“Why?”** → display MeTTa rules fired and evidence.
3. Click **“Verify”** → check signature / Nitro attestation.
4. (Optional) Log report hash to Solana and show tx link.

5. Display Agentverse registration for discoverability.

🎗️ Future Expansion

- Crowd-sourced threat graph (agents share verified indicators via MeTTa KG).
- Reputation staking for analyzer agents.
- Multi-chain support (EVM, Cosmos).
- Browser extension integrations.
- Full TEE rollout + ZK proof of rule execution.

🧩 Summary

A composable, explainable, and **attestable AI-driven phishing defense system** — built from modular **agents** that reason, verify, and collaborate. It bridges traditional security and decentralised AI ecosystems, demonstrating how **trusted AI agents** can protect Web3 users across chains.