🛡️ Enhancing DeFi Security: Upgradeability Detection Tool (UDT)

The Upgradeability Detection Tool (UDT) is a professional-grade Command Line Interface (CLI) application designed to audit smart contracts for upgradeability vulnerabilities (Proxies, Delegatecall, CREATE2) and assign a quantifiable risk score based on centralization of control.

This project utilizes a dual-path analysis methodology (Static Source Code analysis via Slither and Resilient Runtime analysis via Web3/Etherscan) to maximize security coverage.

⚙️ Project Setup and Requirements

1. Prerequisites

Component	Requirement	Status
Docker Engine	Must be installed and running (Docker Desktop).	**Required**
Solidity Compiler	solc v0.8.30	**Installed** (inside the Docker image).
Project Files	All .sol dependency files	Must be in the local project folder.

2. Environment Variables (Required for Runtime Analysis)

For the tool to perform live blockchain checks (-a argument), set these environment variables in your terminal session (PowerShell/CMD):

Example PowerShell Setup:
$env:INFURA_URL = "https://eth-sepolia.g.alchemy.com/v2/fyo2b5k1kvj_cSctPtlurjb7H1MPikmm"
$env:ETHERSCAN_API_KEY = "T5TZ6QEVWPV77NJF3FSRQGNE88M1335RAH"

3. Docker Build (Completed)

The build process successfully created the stable environment where all dependencies (Slither, Web3, solc) are correctly configured.

Final command used to build the image:
docker build -t udi-tool .

🚀 Execution Guide: Running the UDT Tool

Execution must be done using the docker run command to access the compiler and libraries inside the container. We use the **${PWD}:/app** volume mount to link your local project folder to the container's working directory.

CLI Usage Syntax

The tool supports combined analysis of both source files and deployed addresses.

Analysis Type	Input Flag	Example Input
Static Code Audit	-f, --file	Controller.sol
Runtime Check	-a, --address	0x404d33da717a89ec168f9be6dbd21a5a2accfe9a
Output	--csv	analysis_report.csv

Execution Command (The Only Command)

Run this command from your local project directory:

docker run -it -v "${PWD}:/app" udi-tool /bin/bash -c "python3 upgradeability_detector.py -f Controller.sol -a 0x404d33da717a89ec168f9be6dbd21a5a2accfe9a --csv analysis_output.csv"

Expected Output

Your console output will show a **staged analysis**, even though the source code analysis is currently failing due to the compiler version mismatch (see Errors below).

1. **Deployed Address Status (Top Panel):** Shows the result of the Etherscan/Web3 check for 0x404d...fe9a.
2. **Staged Analysis Panel (Source File):** Shows the compilation failure for Controller.sol.
3. **Final Summary Table:** Displays the combined, color-coded risk assessment.

🚨 Current Errors & Troubleshooting

The tool is currently demonstrating a critical security finding related to its inability to run the

compiler, which requires a manual fix of the source code itself.

1. Slither Compilation Error (The Blocker)

- **Error Message:** ❌ Slither Compilation Error for Controller.sol: Solidity compilation failed (Check imports/version).
- **Root Cause:** The Controller.sol file (and likely its 13 dependencies) requires a minimum compiler version of 0.8.30, but the system's execution path is failing to correctly utilize the necessary compiler version when called by Slither.
- **Status:** This prevents the execution of all the intelligence you wrote (proxy, admin, delegatecall checks).

Required Fix (Manual Pragma Change)

To proceed, you must manually edit your Solidity source files to bypass this versioning conflict:

1. **Locate** all 14 dependency .sol files in your project folder.
2. **Change the pragma line** in every single file from the current version (likely ^0.8.30) to the common stable version:
   pragma solidity ^0.8.0;

3. **Rerun the Docker Command.** This manual fix should resolve the compilation error, allowing your core UDT intelligence to execute successfully.

💻 Tool Architecture & Functional Summary

The UDT's Python code contains complete logic for all analysis steps.

Function	Analysis Stage	Role
slither_analyze	Static Analysis (-f)	Executes compilation and feature extraction (inheritance, functions, opcodes).
etherscan_proxy_check	Runtime Check (P1)	Primary method to find implementation address via Etherscan API.
web3_get_storage_at	Runtime Check (P2)	Fallback method to check standardized EIP-1967 storage slots via RPC.
analyze_and_score	Risk Scoring (B3)	Assigns final

		HIGH/MEDIUM/LOW risk based on centralization of control found in the detection stage.
print_staged_results	Reporting	Displays the step-by-step findings of the static analysis (e.g., Compilation Status, Inheritance Found, Access Control Type).