# 🛡️ Enhancing DeFi Security: Upgradeability Detection Tool (CLI)

**Major Project for B.Tech CSE**

This Command Line Interface (CLI) tool performs rapid, modular security analysis on Solidity smart contract source code and deployed contract addresses. It is designed to automatically detect and classify common upgradeability patterns (UUPS, Transparent Proxies, CREATE2) and assign a quantifiable risk score based on centralization of control.

## ⚙️ System Requirements & Setup

### Prerequisites



| **Component** | **Requirement** | **Installation Method** |
| --- | --- | --- |
| **Python** | 3.8+ |  |
| **Solidity Compiler** | solc executable (Version 0.8.30 recommended) | **Critical:** Must be added to your System PATH for the tool to run. |
| **Project Files** | All .sol files (including OpenZeppelin dependencies) | Must be placed in the same directory as upgradeability\_detector.py. |

### Installation (Required Libraries)

Install all necessary Python dependencies using pip:

pip install slither-analyzer web3 requests rich

### Configuration (API Keys)

The deployed analysis (-a argument) requires access to the Ethereum network. You must replace the placeholder values at the top of the script (upgradeability\_detector.py) with your working keys:

| **Variable** | **Description** | **Example (Replaced in Code)** |
| --- | --- | --- |
| INFURA\_URL | Your RPC endpoint for Sepolia (e.g., Alchemy). | "https://eth-sepolia.g.alchemy.com/v2/..." |
| ETHERSCAN\_API\_KEY | Your official Etherscan API key. | "T5TZ6QEVWPV77NJF3FSRQGNE88M1335RAH" |

## 🚀 How to Use the Tool

The tool supports two primary analysis modes: **Static Analysis** (Source Code) and **Runtime Analysis** (Deployed Address).

### CLI Usage Syntax

The tool requires at least one input (-f or -a) and supports optional output flags (--csv).

python upgradeability\_detector.py [INPUT FLAGS] [OUTPUT FLAGS]

### Examples

| **Goal** | **Command** | **Output** |
| --- | --- | --- |
| **Analyze Source & Deployed Address** | python upgradeability\_detector.py -f Controller.sol -a 0x404d...fe9a | Console output showing staged analysis for file and deployed status. |
| **Static Analysis Only** | python upgradeability\_detector.py -f Controller.sol | Console output + Staged Analysis Panel. |
| **Generate CSV Report** | python upgradeability\_detector.py -f Controller.sol --csv report.csv | Generates a clean report.csv file with final summaries. |

## 🛠️ Tool Architecture & Function Breakdown

The tool is structured into three layers (Acquisition, Detection, Reporting). The core intelligence is found in slither\_analyze and analyze\_and\_score.

### Static Analysis Engine (slither\_analyze & Helpers)

| **Function** | **Role in Analysis** | **Status** |
| --- | --- | --- |
| **slither\_analyze(file\_path)** | **Core Analysis Engine.** Loads and compiles the Solidity file using solc, extracts all contract details (inheritance, functions, modifiers, opcodes), and sets the initial flags for the risk scorer. | **Logic Complete** |
| **get\_main\_contract** | **Helper.** Identifies the primary contract to analyze in multi-file projects (e.g., finds Controller among all its imported files). | ✅ |
| **delegatecall\_detector** | **B1 Check.** Returns True if any function in the contract performs a raw DELEGATECALL opcode. | ✅ |
| **create2\_detector** | **B2 Check.** Returns True if any function uses the CREATE2 opcode (flags deterministic deployment). | ✅ |

### Runtime & Risk Scoring

| **Function** | **Role in Analysis** | **Status** |
| --- | --- | --- |
| **etherscan\_proxy\_check** | **P1 Deployed Check.** Primary method for finding the implementation address of a deployed contract using the Etherscan API. | ✅ |
| **web3\_get\_storage\_at** | **P2 Deployed Check.** Fallback method to check standardized EIP-1967 storage slots directly via the RPC node. | ✅ |
| **proxy\_detector** | **Pattern Selector.** Determines the contract type (UUPS, Transparent, Etherscan Verified, or Non-Upgradeable) based on the output of the two checks above. | ✅ |
| **analyze\_and\_score** | **Risk Assessor.** Takes all the flags and applies the final scoring logic (HIGH/MEDIUM/LOW) based on: 1. Proxy Type, 2. Admin Control (e.g., onlyOwner), and 3. Vulnerability flags (e.g., unprotected upgrade function). | ✅ |

## 🚨 Current Errors & Troubleshooting

As of the last execution, the core functionality is blocked by a persistent environment issue, despite correct Python code logic.

| **Error** | **Cause** | **Status & Required Fix** |
| --- | --- | --- |
| **❌ Slither Compilation Error... [WinError 2]** | The solc version required by your Solidity files (0.8.30) is failing to execute when called by Python/Slither, even though solc --version works in the terminal. This is a common Windows security/path restriction. | **Critical Fix:** You must ensure your local Solidity files are compatible with the compiler Slither is running. The fastest solution is to **manually change the pragma lines in ALL 14 Solidity files to pragma solidity ^0.8.0;** and ensure the solc version in the Python code is also set to '0.8.0'. |
| **KeyError: 'name'** | (Resolved by latest code) | This crash occurred because the compilation failure overwrote the reporting structure. The fix ensures the name is preserved so the tool can print the final summary table. |

### Static Analysis Stages (Staged Reporting)

When the tool successfully analyzes a source file, the detailed output panel provides the following staged results:

| **Stage** | **Finding** |
| --- | --- |
| **Compilation** | Success / Failed |
| **Inheritance** | UUPS Found / Transparent Found / CREATE2 Factory |
| **Upgrade\_Fn** | upgradeTo() Found / Not Found |
| **Access\_Control** | Centralized / Secure/Timelock / UNPROTECTED! |
| **Low\_Level\_Calls** | DELEGATECALL detected / Clean |