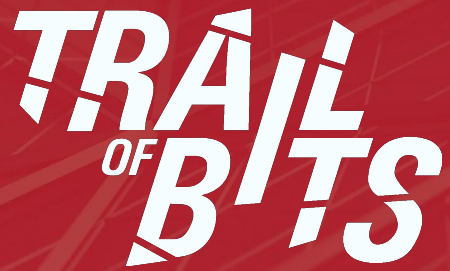


Introducing Slither-MCP

Benjamin Samuels

Who are we?

Trail of Bits - @trailofbits

We help developers design and build bug-free software.

- Early stage consulting to ensure the software produced by clients is bug-free
- Design Reviews (one of which this talk is inspired by!)
- Audits / Code Review

We also maintain tools for building safer software:

- Slither (static analysis)
- Slither-mutate (mutation testing)
- Medusa/Echidna (stateful fuzzing)
- And about two dozen more

Who am I?

Benjamin Samuels @thebensams
Director of Engineering, Blockchain

Check out what we've been up to:

A Short History of Slither

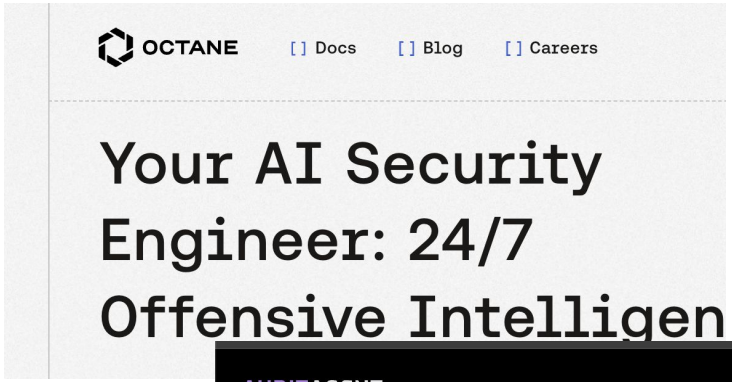
Built in 2018 because we were tired of looking for re-entrancy issues

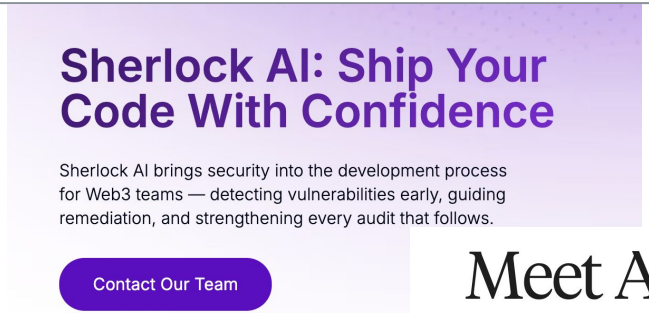
Over time, Slither has helped categorically eliminate the bugs it was the best at detecting, making the tool less useful for bug detection

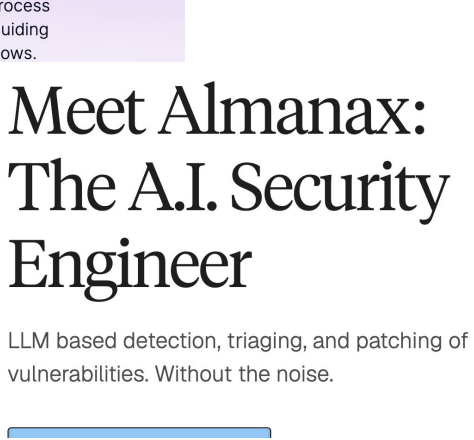
Value has shifted over time into printers. Still helpful, but not groundbreaking.

What is the future of Slither?

LLM-based bug detection







& if you aren't building a tool, you're probably using claude code, cursor, codex, etc.

Problem: What happens when you ask an LLM to analyze a codebase?

Tools available for use:

- grep
- read_file
- ls
- (sometimes) RAG-based search

Example: Where is a function implemented?

ERC20.sol

```
contract ERC20{
    function transfer(...){...}
}
```

MyERC20.sol

```
import "./ERC20.sol"
contract MyERC20 is ERC20{...}
```

Dapp.sol

```
import "./MyERC20.sol"
contract Dapp {
    function foo(address addr) public {
        MyERC20(addr).transfer(...)
    }
}
```

Prompt: Detect bugs in Dapp.sol

LLM has to find where `MyERC20.transfer()` is implemented

Option 1: Use grep_tool, e.g.

`grep('*function transfer(*')`

Problem: Will misunderstand inheritance hierarchy

Option 2: Follow import paths

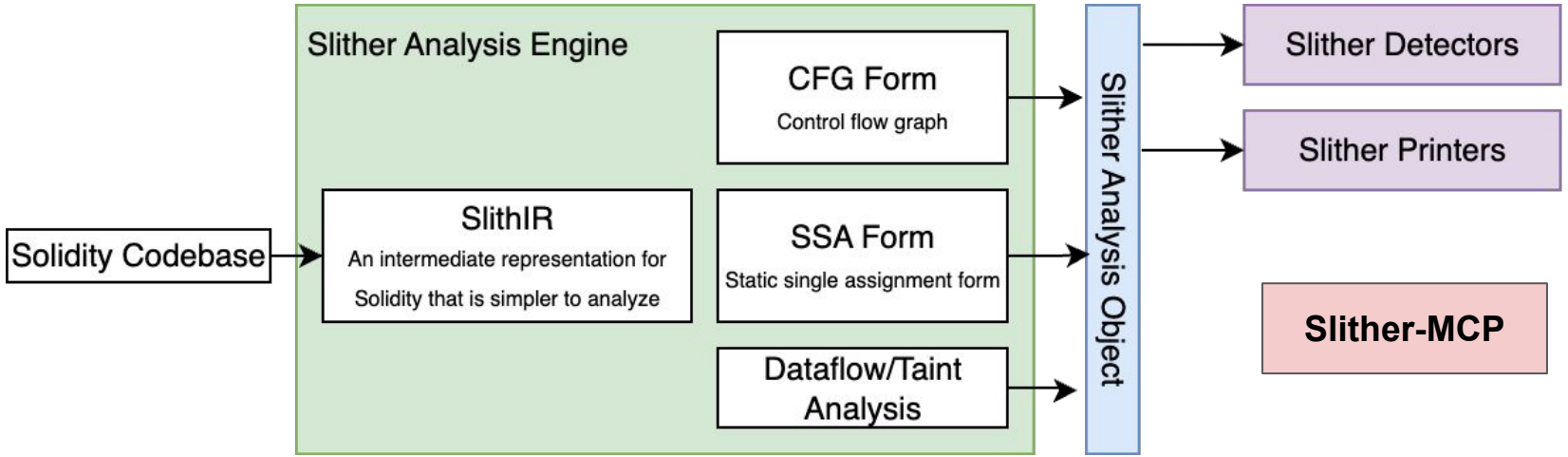
`read_file("./MyERC20.sol")`
`read_file("./ERC20.sol")`

Problem: Resolving path has high error rates for large projects

Problem: Extremely token-hungry

Basic code navigation tasks do not have high reliability, and this is very bad for LLM-based analysis

This is where Slither comes in



Example 1: Identify a function's callers

Problem: The LLM needs to find every code location that calls `IOracle.price(address)`

Without Slither-MCP

1. grep *(CONTRACT).price(*)*
2. For each result, call read_file to identify the type of CONTRACT and ensure the argument type is `address`
3. Call read_file for CONTRACT's source code to see if IOracle is in its inheritance hierarchy. May involve multiple read_file calls to check parent contracts for the interface.

With Slither-MCP

```
get_function_callers({
   signature: "price(address)",
   contract_name: "IOracle",
   path: "src/interfaces/IOracle.sol"
})
```

Example 2: Identify the functions in a contract

Problem: The LLM needs to understand all of the functions a contract has

Without Slither-MCP

1. read_file to read contract
2. Identify inherited contracts
3. read_file on each inherited contract, resolving function overrides along the way
4. Go to step 2 and repeat for the contract's full inheritance hierarchy

With Slither-MCP

```
list_functions({
   contract_name: “IOracle”,
   path: “src/interfaces/IOracle.sol”
})
```

How can I use Slither-MCP?

Run it as an MCP for Claude Code/Cursor:

```
claude mcp add --transport stdio --scope user
slither -- uvx --from
git+https://github.com/trailofbits/slither-mcp
slither-mcp
```

```
Run sudo ln -s ~/.local/bin/uvx /usr/local/bin/uvx
Then use this config:
{
 "mcpServers": {
    "slither-mcp": {
      "command": "uvx --from
git+https://github.com/trailofbits/slither-mcp slither-mcp"
    }
  }
}
```

Use it in your agents as a Pydantic tool (e.g. `create_get_function_source_tool`, etc.)

Use it as a Slither API client using SlitherMCPClient

Questions?

Problem: How do we design the MCP API?

- **Multiple contracts of the same name can exist in the same Solidity project, so we can't have a simple API like:** `get_inherited_contracts("ERC20")`
- **We need a unified naming convention for functions.** `ContractName.FunctionName()` **isn't adequate due to the contract name scoping issue.**

Problem: How do we design the MCP API?

Solution: ContractKeys and FunctionKeys

```
get_contract_source({
     contract_name: “ERC20”,
     path: “lib/openzeppelin/contracts/ERC20.sol”
})

get_function_callees({
     signature: “transfer(address,uint256)”,
     contract_name: “ERC20”,
     path: “lib/openzeppelin/contracts/ERC20.sol”
})
```

This means the LLM needs to be made aware of the project’s paths, ContractKeys, etc.

```
list_contracts() =>
[
    {
        contract_name: “ERC20”,
        path: “lib/openzeppelin/contracts/ERC20.sol”
    },
    ...
]
```