Getting Started with Web3 Audits

Shadow Auditing CodeHawks FirstFlight “Inheritable Wallet”

Songhyun Bae

Who Am I



Songhyun Bae

Korea Univ CYDF, CyKor
Bug Hunter, Auditor, CTF Player

@bshyuunn

Agenda

- What is a Web3 Audit?
- Choosing a Target
- Understanding the Protocol
- Analyzing the Protocol
- Identifying Vulnerabilities
- Building a Proof of Concept (PoC)
- Writing the Audit Report
- Conclusion

What is a Web3 Audit?

What is a Web3 Audit?

ChainLight

WEB3 HACK POSTMORTEM 2024

Ver 1.0

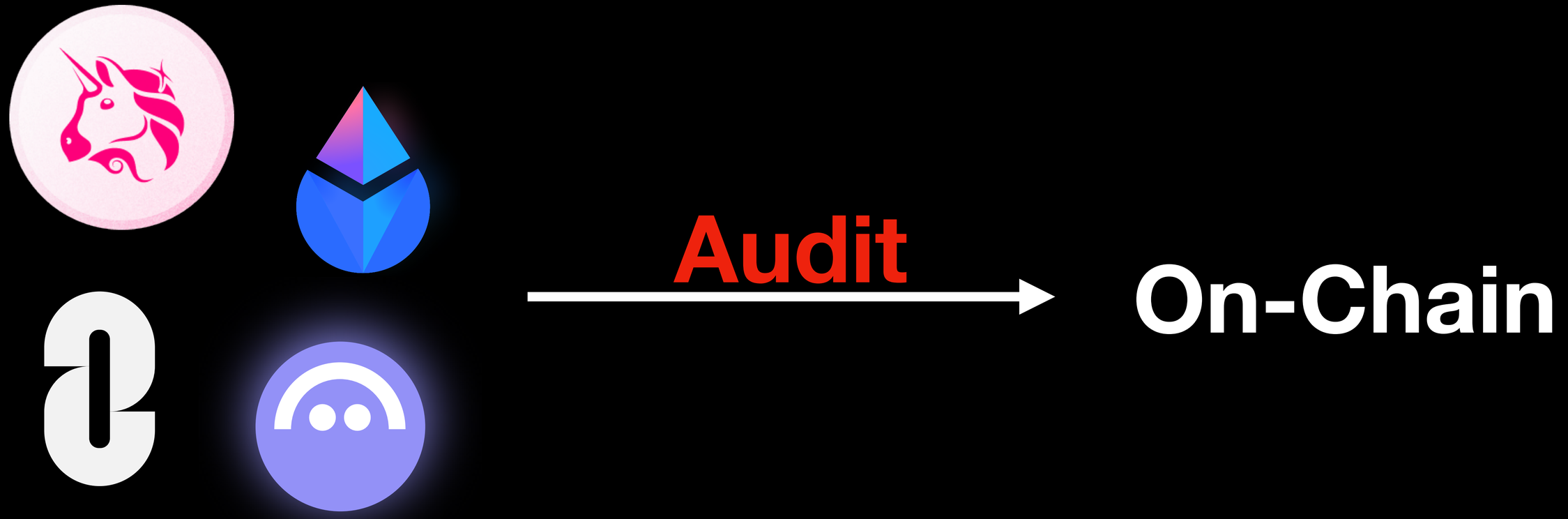


1. SECURITY INCIDENTS

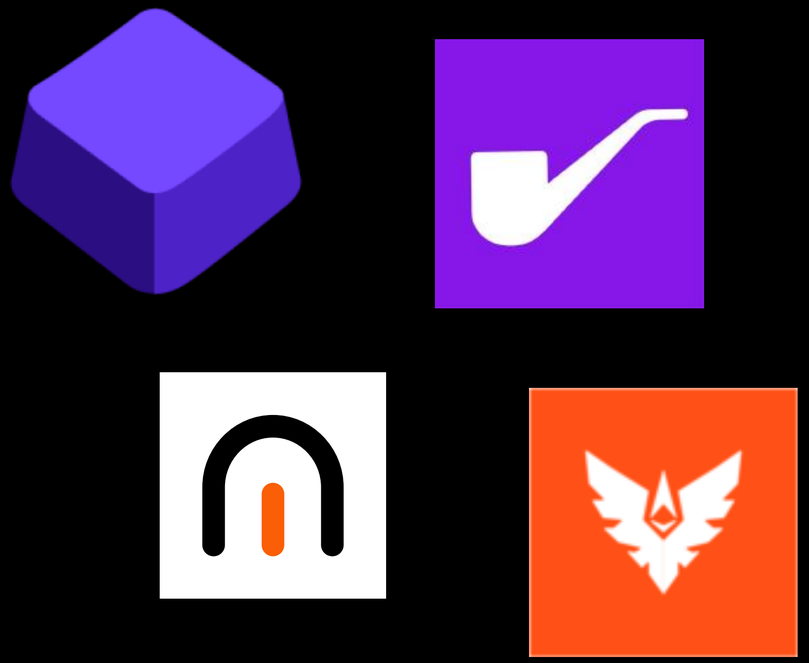
Scale

In 2024, we identified **204** security incidents with various root causes, occurring at an average interval of **1.78** days. The total damage amounted to at least **$1.88 Billion**, with each project suffering an average loss of **$9.57 Million**.

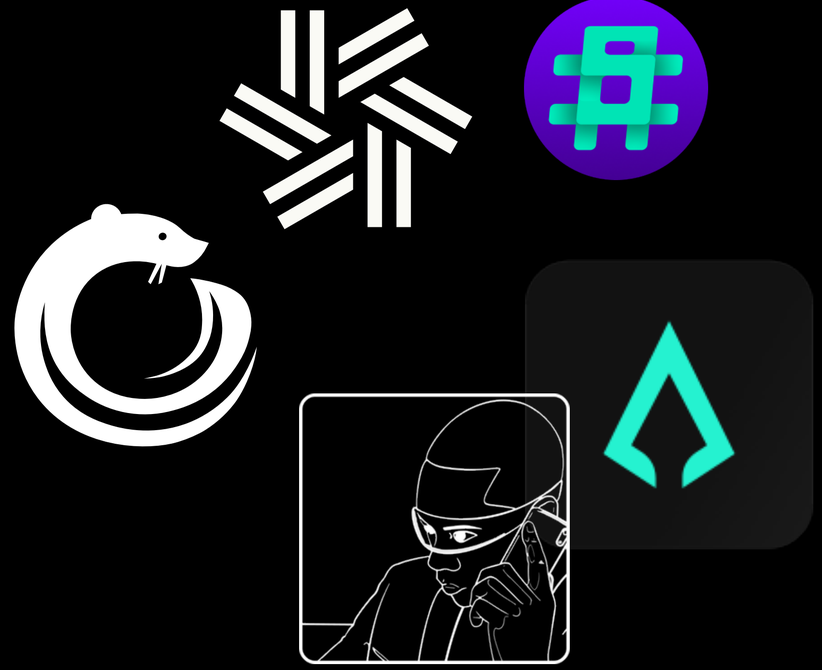
What is a Web3 Audit?



What is a Web3 Audit?



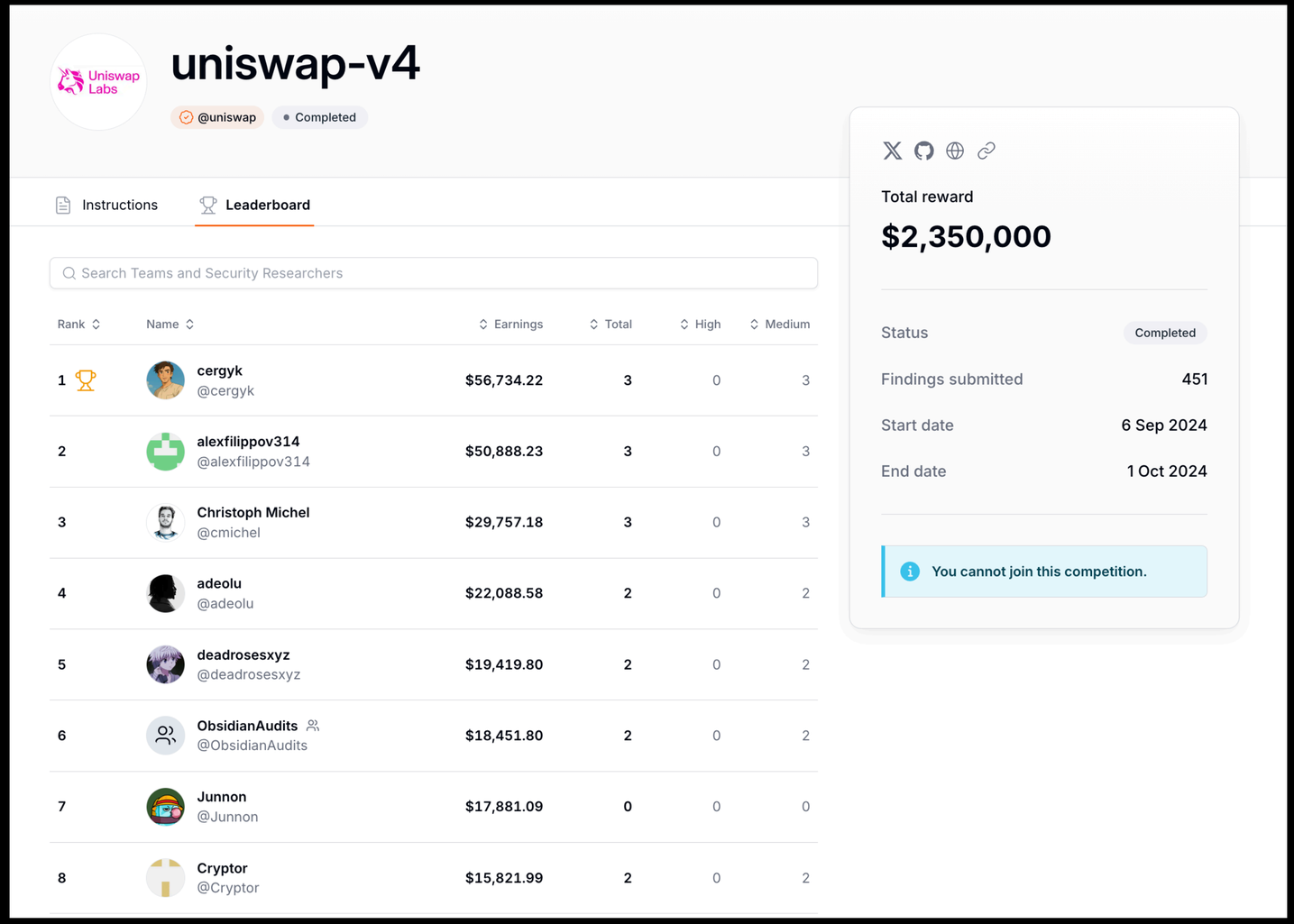
VS



Audit Contest

Audit Firm

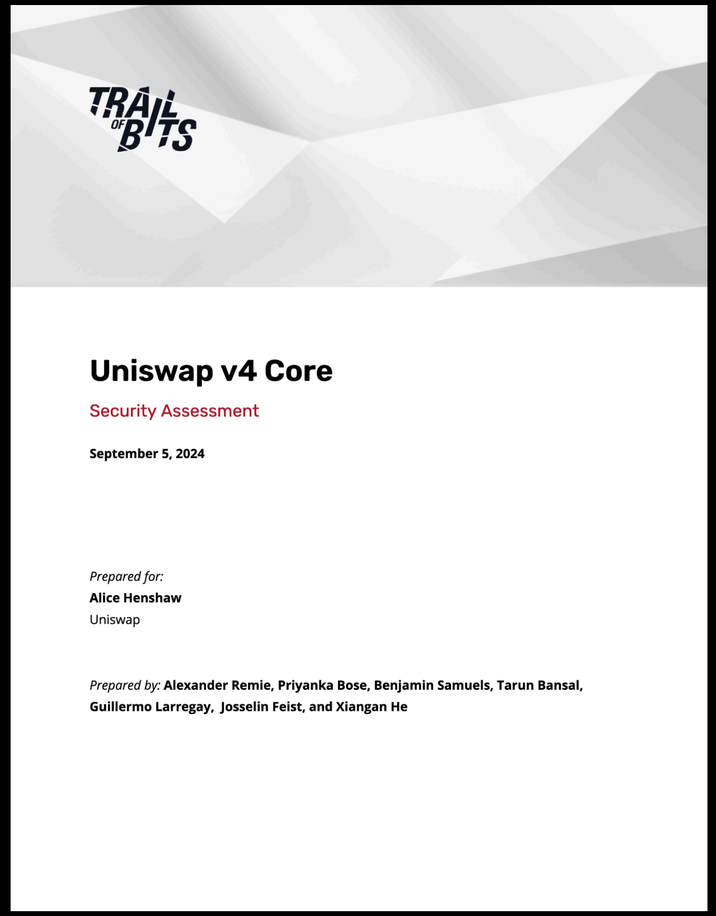
What is a Web3 Audit?



What is a Web3 Audit?

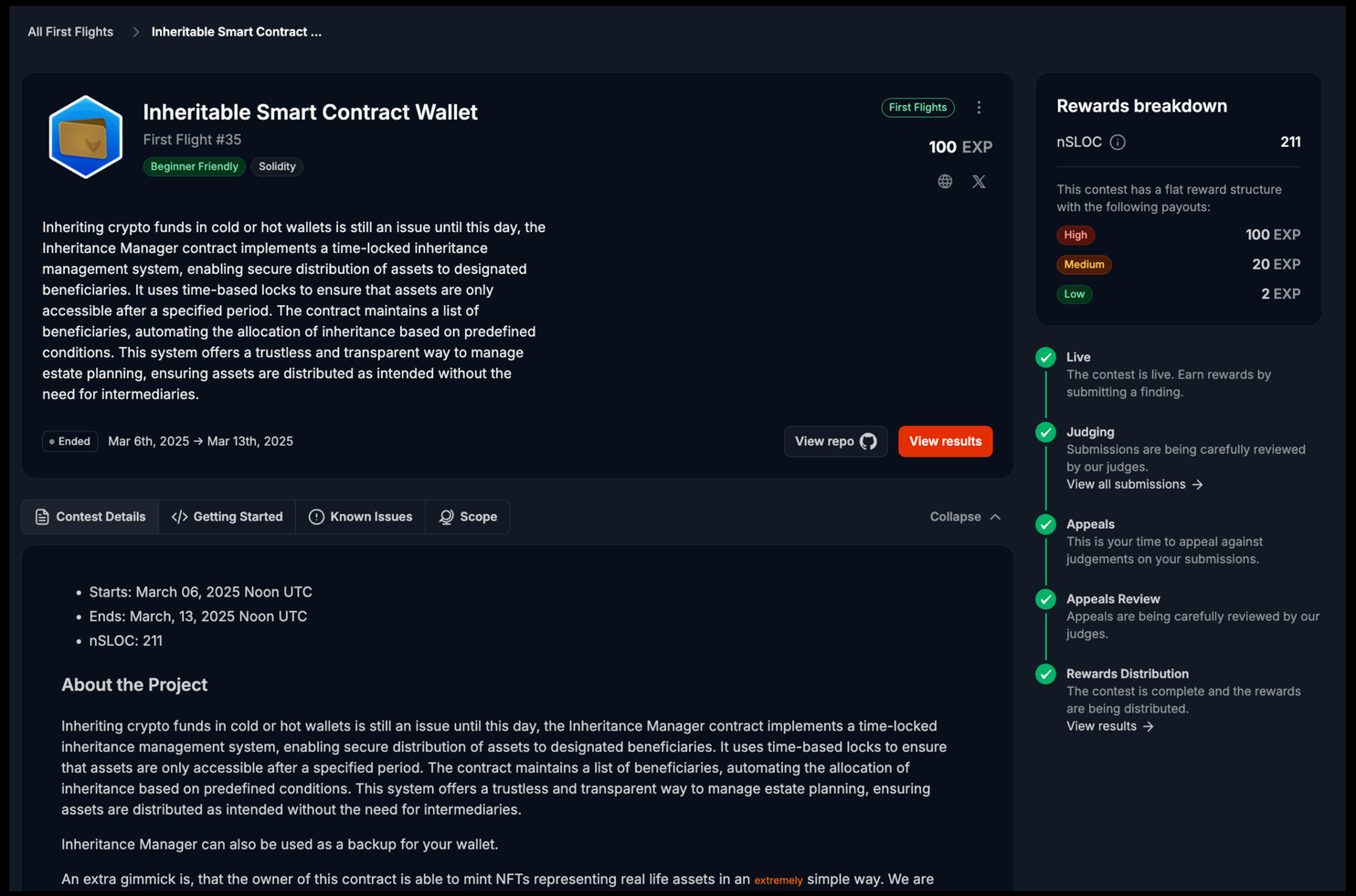




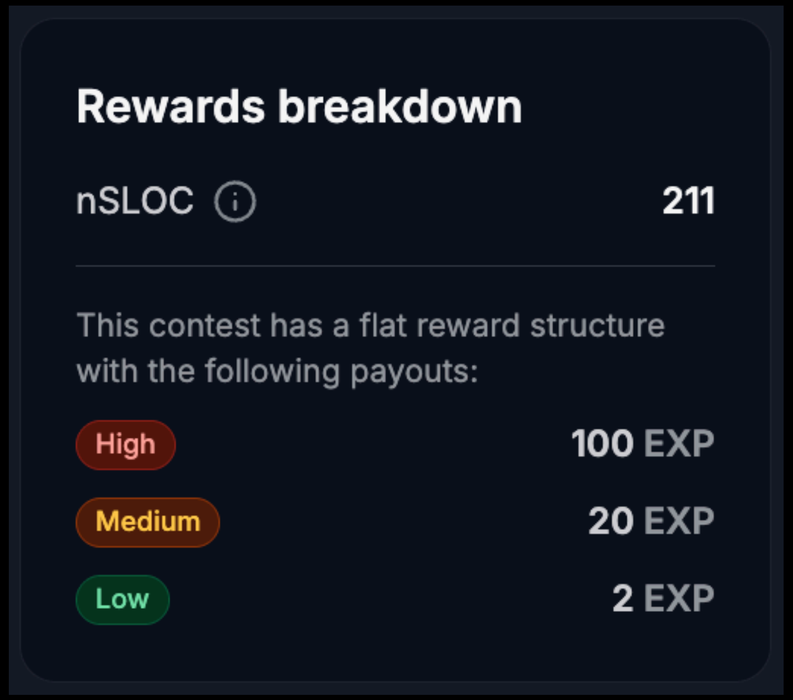


Choosing a Target

Choosing a Target



Choosing a Target



nSLOC

nSLOC stands for 'Normalized Source Lines of Code', a custom measurement we use (among other things) to evaluate the complexity of a codebase.

To get the NSLOC count of a file:

1. For all functions, reduce any multiline function declarations to a single line.
2. Remove all comments
3. Remove all empty lines
4. Count the remaining lines

Choosing a Target

```
Compatibilities:
  Blockchains:
      - Ethereum
  Tokens:
      - No limitations. Inheritance Manager needs to be compatible with ether and every ERC20 token.
```

Understanding the Protocol

Understanding the Protocol



The **Inheritance Manager** contract enables trustless **estate planning** by locking crypto assets until a set time has passed, then automatically distributing them to beneficiaries. It ensures secure, transparent transfers without intermediaries.

Understanding the Protocol

Owner: The Owner of the smart contract wallet

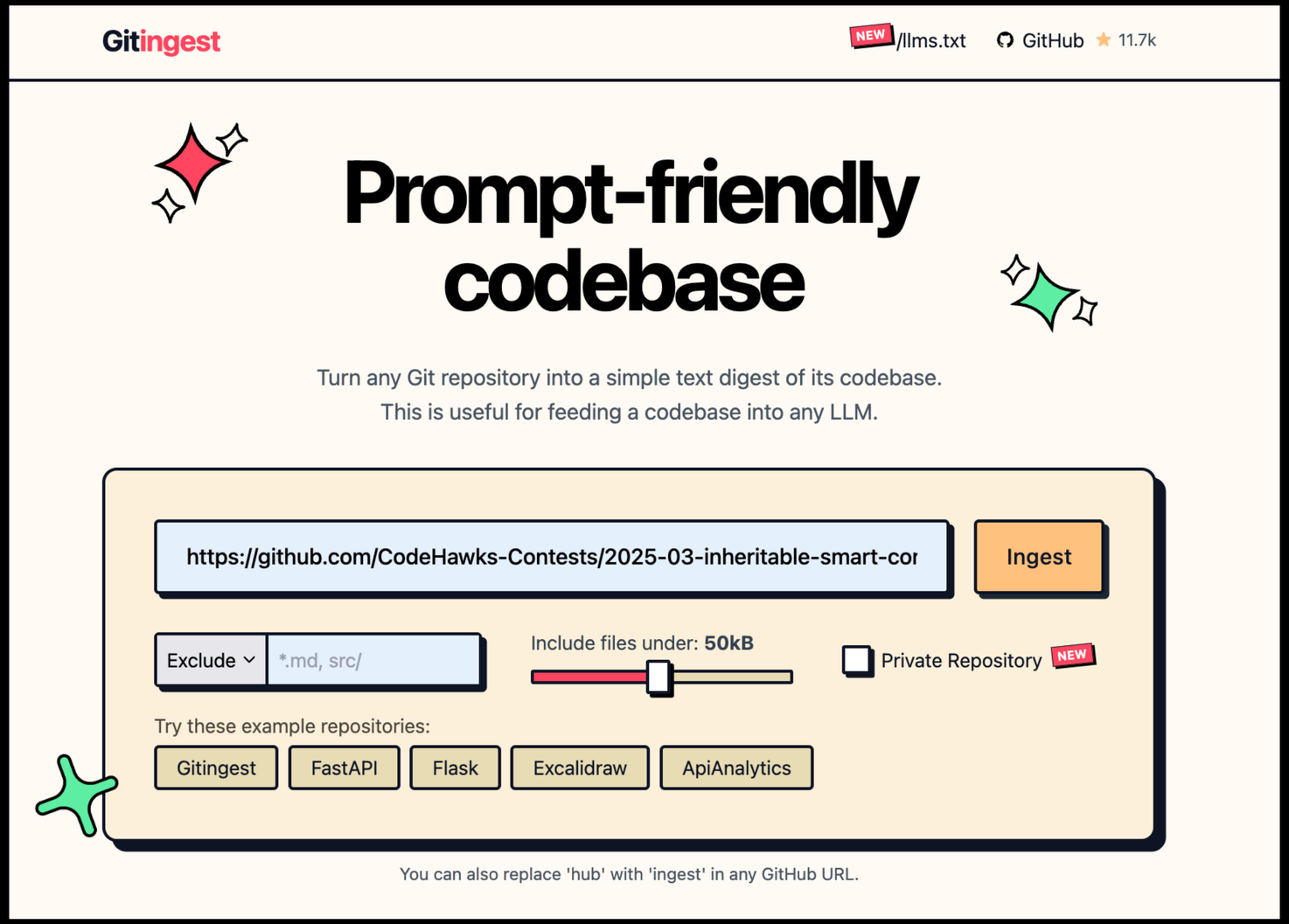
Beneficiary: Anyone set by the owner to inherit the smart contract and all it's balances.

Trustee: Not necessary role, but can be appointed by the beneficiearies in case underlaying estate values of the NFTs need to be reevaluated and/or the payout asset has to be changed.

Understanding the Protocol

```
src/
├── InheritanceManager.sol
├── NFTFactory.sol
├── modules/
    ├──Trustee.sol
```

Understanding the Protocol



Understanding the Protocol

```
# Imagine you are a security researcher and you have to describe in detail th
e project below.
- First, explain what the protocol as a whole is
- Describe use cases, flowcharts for protocol users
- List any additional attack vectors that may exist
- List any additional invariants that exist
- All Answers in Korean

---
### This is entire code

---
### This is entire doc
```

Analyzing the Protocol

Analyzing the Protocol

$ git clone https://github.com/CodeHawks-Contests/2025-03-inheritable-smart-contract-wallet.git

$ cd 2025-03-inheritable-smart-contract-wallet

$ code .

Analyzing the Protocol

```
$ curl -L https://foundry.paradigm.xyz | bash

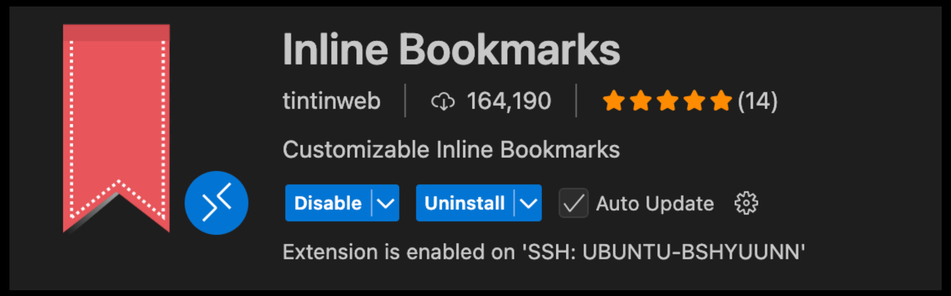
$ foundryup

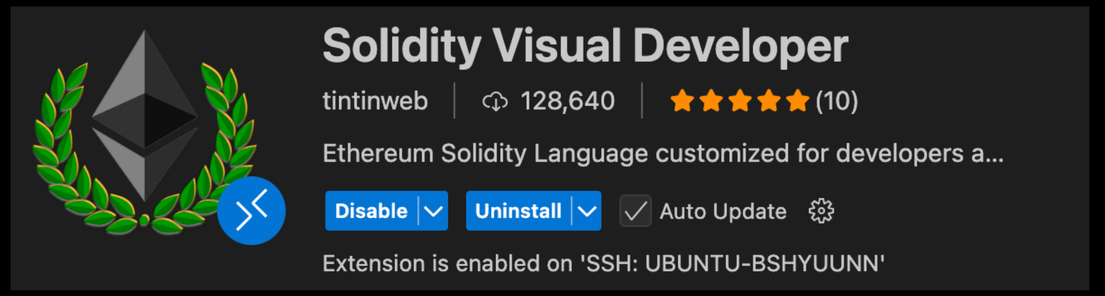
$ forge install

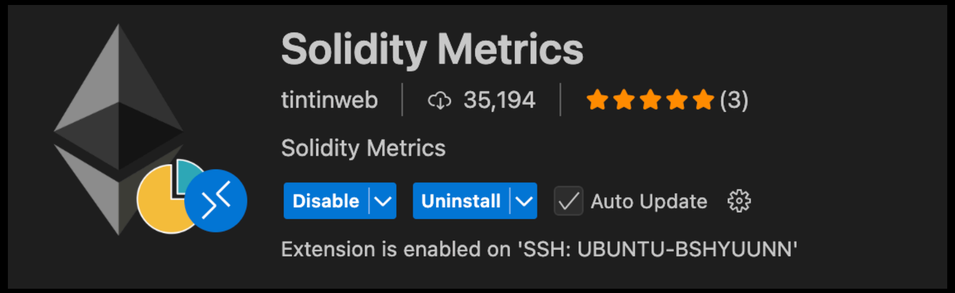
$ forge build

$ forge test
```

Analyzing the Protocol







Analyzing the Protocol

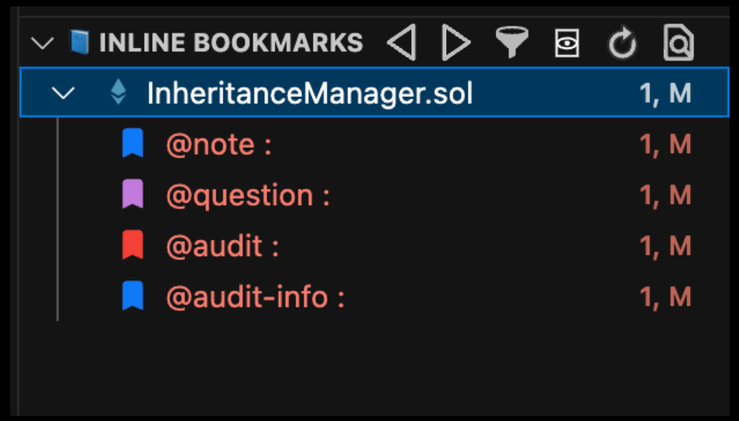
Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	src/InheritanceManager.sol	1	————	285	281	157	95	147	🖥️🌀
📝	src/NFTFactory.sol	1	————	39	39	29	1	18	————
🎨	src/modules/Trustee.sol	1	————	35	35	25	1	16	————
📝🎨	**Totals**	**3**	————	**359**	**355**	**211**	**97**	**181**	🖥️🌀

Analyzing the Protocol

```
function contractInteractions(address _target⬆, bytes calldata _payload⬆, uint256 _value⬆, bool _storeTarget⬆)
    external
    nonReentrant
    onlyOwner
{
    (bool success, bytes memory data) = _target⬆.call{value: _value⬆}(_payload⬆);
    require(success, "interaction failed");
    if (_storeTarget⬆) {
        interactions[_target⬆] = data;
    }
}
```

Analyzing the Protocol

```
286
287   // @note :
288   // @question :
289   // @audit :
290   // @audit-info :
291
```



Time to Analyze

Identifying Vulnerabilities

Identifying Vulnerabilities

```
/**
 * @dev On-Chain payment of underlaying assets.
 * CAN NOT use ETHER
 * @param _nftID NFT ID to buy out
 */
ftrace | funcSig
function buyOutEstateNFT(uint256 _nftID⬆) external onlyBeneficiaryWithIsInherited {
    uint256 value = nftValue[_nftID⬆];
    uint256 divisor = beneficiaries.length;
    uint256 multiplier = beneficiaries.length - 1;
    uint256 finalAmount = (value / divisor) * multiplier;
    IERC20(assetToPay).safeTransferFrom(msg.sender, address(this), finalAmount);
    for (uint256 i = 0; i < beneficiaries.length; i++) {
        if (msg.sender == beneficiaries[i]) {
            return;
        } else {
            IERC20(assetToPay).safeTransfer(beneficiaries[i], finalAmount / divisor);
        }
    }
    nft.burnEstate(_nftID⬆);
}
```

Identifying Vulnerabilities

return ??

```
/**
 * @dev On-Chain payment of underlaying assets.
 * CAN NOT use ETHER
 * @param _nftID NFT ID to buy out
 */
ftrace | funcSig
function buyOutEstateNFT(uint256 _nftID⬆) external onlyBeneficiaryWithIsInherited {
    uint256 value = nftValue[_nftID⬆];
    uint256 divisor = beneficiaries.length;
    uint256 multiplier = beneficiaries.length - 1;
    uint256 finalAmount = (value / divisor) * multiplier;
    IERC20(assetToPay).safeTransferFrom(msg.sender, address(this), finalAmount);
    for (uint256 i = 0; i < beneficiaries.length; i++) {
        if (msg.sender == beneficiaries[i]) {
            return;
        } else {
            IERC20(assetToPay).safeTransfer(beneficiaries[i], finalAmount / divisor);
        }
    }
    nft.burnEstate(_nftID⬆);
}
```

Identifying Vulnerabilities

```
/**
 * @dev On-Chain payment of underlaying assets.
 * CAN NOT use ETHER
 * @param _nftID NFT ID to buy out
 */
ftrace | funcSig
function buyOutEstateNFT(uint256 _nftID⬆) external onlyBeneficiaryWithIsInherited {
    uint256 value = nftValue[_nftID⬆];
    uint256 divisor = beneficiaries.length;
    uint256 multiplier = beneficiaries.length - 1;
    uint256 finalAmount = (value / divisor) * multiplier;
    IERC20(assetToPay).safeTransferFrom(msg.sender, address(this), finalAmount);
    for (uint256 i = 0; i < beneficiaries.length; i++) {
        if (msg.sender == beneficiaries[i]) {
            return;
        } else {
            IERC20(assetToPay).safeTransfer(beneficiaries[i], finalAmount / divisor);
        }
    }
    nft.burnEstate(_nftID⬆);
}
```

return ??

divisor ??

Identifying Vulnerabilities

```
/**
 * @dev removes entries from beneficiaries in case inheritance gets revoked or
 * an address needs to be replaced (lost keys e.g.)
 * @param _beneficiary address to be removed from the array beneficiaries
 */
ftrace | funcSig
function removeBeneficiary(address _beneficiary⬆) external onlyOwner {
    uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary⬆);
    delete beneficiaries[indexToRemove];
}
```

Identifying Vulnerabilities

```
/**
 * @dev removes entries from beneficiaries in case inheritance gets revoked or
 * an address needs to be replaced (lost keys e.g.)
 * @param _beneficiary address to be removed from the array beneficiaries
 */
ftrace | funcSig
function removeBeneficiary(address _beneficiary⬆) external onlyOwner {
    uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary⬆);
    delete beneficiaries[indexToRemove];
}
```

Just delete ??

Identifying Vulnerabilities

```
/**
 * @dev gas efficient cross-function reentrancy lock using transient storage
 * @notice refer here: https://soliditylang.org/blog/2024/01/26/transient-storage/
 */
modifier nonReentrant() {
    assembly {
        if tload(1) { revert(0, 0) }
        tstore(0, 1)
    }
    _;
    assembly {
        tstore(0, 0)
    }
}
```

Identifying Vulnerabilities

Index 1 ??

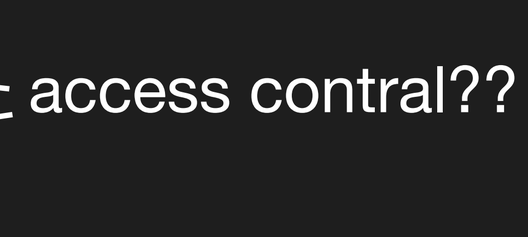
```
/**
 * @dev gas efficient cross-function reentrancy lock using transient storage
 * @notice refer here: https://soliditylang.org/blog/2024/01/26/transient-storage/
 */
modifier nonReentrant() {
    assembly {
        if tload(1) { revert(0, 0) }
        tstore(0, 1)
    }
    _;
    assembly {
        tstore(0, 0)
    }
}
```

Identifying Vulnerabilities

```
/**
 * @dev manages the inheritance of this wallet either
 * 1. the owner lost his keys and wants to reclaim this contract from beneficiaries slot0
 * 2. the owner was inactive more than 90 days and beneficiaries will claim remaining funds.
 */
ftrace | funcSig
function inherit() external {
    if (block.timestamp < getDeadline()) {
        revert InactivityPeriodNotLongEnough();
    }
    if (beneficiaries.length == 1) {
        owner = msg.sender;
        _setDeadline();
    } else if (beneficiaries.length > 1) {
        isInherited = true;
    } else {
        revert InvalidBeneficiaries();
    }
}
```

Identifying Vulnerabilities

```
/**
 * @dev manages the inheritance of this wallet either
 * 1. the owner lost his keys and wants to reclaim this contract from beneficiaries slot0
 * 2. the owner was inactive more than 90 days and beneficiaries will claim remaining funds.
 */
ftrace | funcSig
function inherit() external {
    if (block.timestamp < getDeadline()) {
        revert InactivityPeriodNotLongEnough();
    }
    if (beneficiaries.length == 1) {
        owner = msg.sender;
        _setDeadline();
    } else if (beneficiaries.length > 1) {
        isInherited = true;
    } else {
        revert InvalidBeneficiaries();
    }
}
```



Identifying Vulnerabilities

```
/**
 * @dev to allow the owner arbitrary calls to other contracts. e.g. deposit assets into Aave to earn yield, or swap tokens on exchanges
 * @notice allows transactions to be stored in interactions[] to make it clear to beneficiaries where to look for funds outside this contract
 * and potentially withdraw those. Obviously swaps would not need to be stored, therefor we give the option.
 * This function should generally only be used by very advanced users, and we assume appropriate diligence has to be done by owner.
 * @param _target address of the target contract
 * @param _payload bytes element with interaction instructions
 * @param _value value of ether to be send with the transaction
 * @param _storeTarget bool to decide if this transaction should be stored
 */
ftrace | funcSig
function contractInteractions(address _target⬆, bytes calldata _payload⬆, uint256 _value⬆, bool _storeTarget⬆)
    external
    nonReentrant
    onlyOwner
{
    (bool success, bytes memory data) = _target⬆.call{value: _value⬆}(_payload⬆);
    require(success, "interaction failed");

    if (_storeTarget⬆) {
        interactions[_target⬆] = data;
    }
}
```

Identifying Vulnerabilities

```
/**
 * @dev to allow the owner arbitrary calls to other contracts. e.g. deposit assets into Aave to earn yield, or swap tokens on exchanges
 * @notice allows transactions to be stored in interactions[] to make it clear to beneficiaries where to look for funds outside this contract
 * and potentially withdraw those. Obviously swaps would not need to be stored, therefor we give the option.
 * This function should generally only be used by very advanced users, and we assume appropriate diligence has to be done by owner.
 * @param _target address of the target contract
 * @param _payload bytes element with interaction instructions
 * @param _value value of ether to be send with the transaction
 * @param _storeTarget bool to decide if this transaction should be stored
 */
ftrace | funcSig
function contractInteractions(address _target⬆, bytes calldata _payload⬆, uint256 _value⬆, bool _storeTarget⬆)
    external
    nonReentrant
    onlyOwner
{
    (bool success, bytes memory data) = _target⬆.call{value: _value⬆}(_payload⬆);
    require(success, "interaction failed");

    if (_storeTarget⬆) {
        interactions[_target⬆] = data;
    }
}
```

after inheritance ??

Identifying Vulnerabilities

```
/**
 * @dev creates an NFT of an underlaying asset, for example real estate. Mints the nft and adds it
 * into nftValue mapping, connecting it to a real world price.
 * @param _description describes the asset, for example address or title number
 * @param _value uint256 describing the value of an asset, we recommend using a stablecoin like USDC or DAI
 * @param _asset the address of the asset in which beneficiaries would need to pay for that asset.
 */
ftrace | funcSig
function createEstateNFT(string memory _description, uint256 _value, address _asset) external onlyOwner {
    uint256 nftID = nft.createEstate(_description);
    nftValue[nftID] = _value;
    assetToPay = _asset;
}

/**
 * @dev adds a beneficiary for possible inheritance of funds.
 * @param _beneficiary beneficiary address
 */
ftrace | funcSig
function addBeneficiery(address _beneficiary) external onlyOwner {
    beneficiaries.push(_beneficiary);
    _setDeadline();
}

/**
 * @dev removes entries from beneficiaries in case inheritance gets revoked or
 * an address needs to be replaced (lost keys e.g.)
 * @param _beneficiary address to be removed from the array beneficiaries
 */
ftrace | funcSig
function removeBeneficiary(address _beneficiary) external onlyOwner {
    uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary);
    delete beneficiaries[indexToRemove];
}
```

Identifying Vulnerabilities

```
/**
 * @dev creates an NFT of an underlaying asset, for example real estate. Mints the nft and adds it
 * into nftValue mapping, connecting it to a real world price.
 * @param _description describes the asset, for example address or title number
 * @param _value uint256 describing the value of an asset, we recommend using a stablecoin like USDC or DAI
 * @param _asset the address of the asset in which beneficiaries would need to pay for that asset.
 */
ftrace | funcSig
function createEstateNFT(string memory _description, uint256 _value, address _asset) external onlyOwner {
    uint256 nftID = nft.createEstate(_description);
    nftValue[nftID] = _value;
    assetToPay = _asset;
}

/**
 * @dev adds a beneficiary for possible inheritance of funds.
 * @param _beneficiary beneficiary address
 */
ftrace | funcSig
function addBeneficiery(address _beneficiary) external onlyOwner {
    beneficiaries.push(_beneficiary);
    _setDeadline();
}

/**
 * @dev removes entries from beneficiaries in case inheritance gets revoked or
 * an address needs to be replaced (lost keys e.g.)
 * @param _beneficiary address to be removed from the array beneficiaries
 */
ftrace | funcSig
function removeBeneficiary(address _beneficiary) external onlyOwner {
    uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary);
    delete beneficiaries[indexToRemove];
}
```

_setDeadline ??

Identifying Vulnerabilities

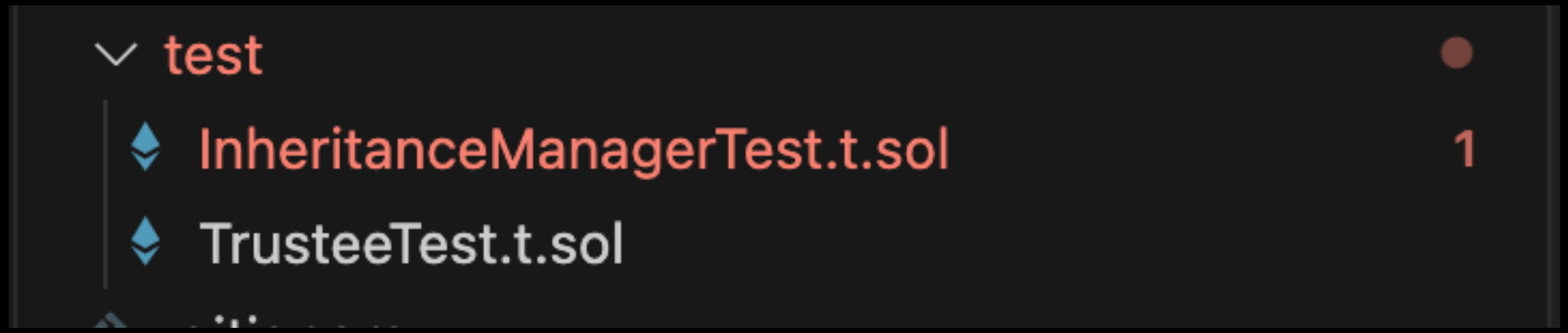
One more

Identifying Vulnerabilities

No possibility to send ETH to `InheritanceManager`

Building a Proof of Concept

Building a Proof of Concept



Building a Proof of Concept

```
pragma solidity 0.8.10;

import {Test} from "forge-std/Test.sol";

contract ContractBTest is Test {
    uint256 testNumber;

    function setUp() public {
        testNumber = 42;
    }

    function test_NumberIs42() public {
        assertEq(testNumber, 42);
    }

    /// forge-config: default.allow_internal_expect_revert = true
    function testRevert_Subtract43() public {
        vm.expectRevert();
        testNumber -= 43;
    }
}
```

Building a Proof of Concept

Console Logging

- Similar to Hardhat's console functions.
- You can use it in calls and transactions. It also works with view and pure functions.
- It always works, regardless of the call or transaction failing or being successful.
- To use it you need import it:
  - `import {console} from "forge-std/console.sol";`
- You can call console.log with up to 4 parameters in any order of following types:
  - `uint`
  - `string`
  - `bool`
  - `address`
- There's also the single parameter API for the types above, and additionally bytes, bytes1... up to bytes32:
  - `console.logInt(int i)`
  - `console.logUint(uint i)`
  - `console.logString(string memory s)`
  - `console.logBool(bool b)`
  - `console.logAddress(address a)`
  - `console.logBytes(bytes memory b)`
  - `console.logBytes1(bytes1 b)`
  - `console.logBytes2(bytes2 b)`
  - ...
  - `console.logBytes32(bytes32 b)`
- console.log implements the same formatting options that can be found in Hardhat's console.log.
  - Example: `console.log("Changing owner from %s to %s", currentOwner, newOwner)`

Building a Proof of Concept

```
/// function withdraw() public {
///     require(msg.sender == owner);

vm.prank(owner);
myContract.withdraw(); // [PASS]
```

```
function startPrank(address msgSender) external;
```

```
function startPrank(address msgSender, bool delegateCall) external;
```

```
function startPrank(address msgSender, address txOrigin) external;
```

```
function startPrank(address msgSender, address txOrigin, bool delegateCall) exte
```

Building a Proof of Concept

```
address alice = makeAddr("alice");
emit log_address(alice); // 0x328809bc894f92807417d2dad6b7c998c1afdac6
```

```
vm.warp(1641070800);
emit log_uint(block.timestamp); // 1641070800
```

```
address alice = makeAddr("alice");
emit log_address(alice);
vm.deal(alice, 1 ether);
log_uint256(alice.balance); // 1000000000000000000
```

Writing the Audit Report

Writing the Audit Report

01. Relevant Github Links
02. Summary
03. Vulnerability Details
04. Impact
05. Tools Used
06. Recommended Mitigation

Writing the Audit Report

```
# I'm writing a vulnerability report. So far, I've only filled out the vulner
ability description and the GitHub link. Based on that content, please comple
te the rest of the report.
- Write it in full report format
- Keep it as simple as possible, almost like you're just translating what I a
lready wrote
- Write it in English
- Remove any quoted lines

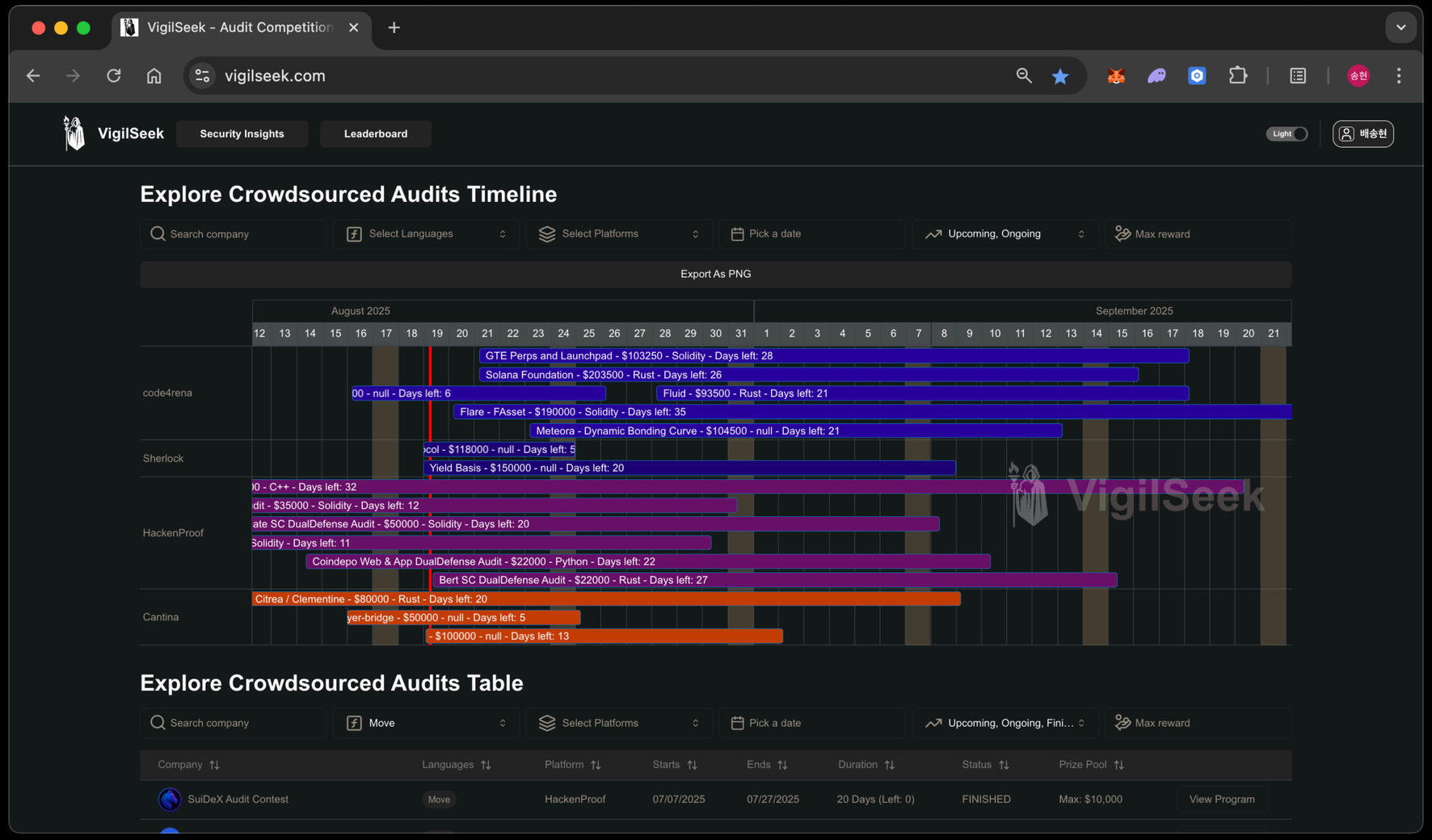
---
Here is the report template:
Title:
Severity: (High/Medium/Low)
Likelihood: (High/Medium/Low)
Impact: (High/Medium/Low)

---
Below is the full code of the contract for reference:
:

---
Here's a similar vulnerability report for reference:
:
```

Conclusion

Conclusion


VigilSeek - Audit Competition
vigilseek.com
VigilSeek
Security Insights
Leaderboard
Light
배송현
Explore Crowdsourced Audits Timeline
Search company
Select Languages
Select Platforms
Pick a date
Upcoming, Ongoing
Max reward
Export As PNG
August 2025
September 2025
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
code4rena
GTE Perps and Launchpad - $103250 - Solidity - Days left: 28
Solana Foundation - $203500 - Rust - Days left: 26
00 - null - Days left: 6
Fluid - $93500 - Rust - Days left: 21
Flare - FAsset - $190000 - Solidity - Days left: 35
Meteora - Dynamic Bonding Curve - $104500 - null - Days left: 21
Sherlock
ccol - $118000 - null - Days left: 5
Yield Basis - $150000 - null - Days left: 20
HackenProof
0 - C++ - Days left: 32
dit - $35000 - Solidity - Days left: 12
ate SC DualDefense Audit - $50000 - Solidity - Days left: 20
Solidity - Days left: 11
Coindepo Web & App DualDefense Audit - $22000 - Python - Days left: 22
Bert SC DualDefense Audit - $22000 - Rust - Days left: 27
Cantina
Citrea / Clementine - $80000 - Rust - Days left: 20
yer-bridge - $50000 - null - Days left: 5
- $100000 - null - Days left: 13
VigilSeek
Explore Crowdsourced Audits Table
Search company
Move
Select Platforms
Pick a date
Upcoming, Ongoing, Fini...
Max reward
Company
Languages
Platform
Starts
Ends
Duration
Status
Prize Pool
SuiDeX Audit Contest
Move
HackenProof
07/07/2025
07/27/2025
20 Days (Left: 0)
FINISHED
Max: $10,000
View Program

Conclusion



Getting Started with Web3 Audits

Auditing CodeHawks FirstFlight “Rock Paper Scissors” Contest

Songhyun Bae