# Bug Hunter

White Paper

Version: 1.0.0-beta.1

Date 30th June 2025

## Executive Summary

Bug Hunter is an automated code review engine for solidity codebases. Utilizing an ensemble of machine learning layers and program analysis tools, it mimics auditor's workflow to identify vulnerabilities in solidity codebases. Analysis of popular solidity based Ethereum ecosystem projects as benchmarks reveal Bug Hunter's superior performance in terms of accuracy, speed and coverage.

For popular yield farming protocols, on average it is able to find 46.5% of the bugs that a security researcher shall find in an in-depth audit analysis. This success rate increased up to 66% in best case result. For NFT projects, its overall average performance was 50% in comparison to security researcher's benchmark audits results where it achieved 55% of best-case result in one of the projects.

Especially noteworthy is Bug Hunter's performance on common core functions in yield farming and NFT projects. For common core functions in yield farming projects, it achieved the best-case result of 75% with average of 58%. In NFT projects, the results on common core functions were 64% (overall average) and 88% (best-case). Bug Hunter achieved this superior Bug Hunting performance with <5% of false positive rate, making it highly reliable as a security review partner during product development stage of smart contract development.

# 1.0 What is Bug Hunter?

Bug Hunter is an automated code review engine for solidity codebases. It aims to find potential high impact vulnerabilities in solidity codebases.

# 2.0 Technological Notes

Bug Hunter combines advanced machine learning methods and program analysis techniques to detect vulnerabilities in solidity codebases. The technological underpinnings are as follows:

## 2.1 Technological Underpinnings

A typical security researcher (auditor) find bugs in solidity codebases by employing a multi-step process that involves manual review and program analysis techniques such as fuzzing and (or) formal verification. Bug Hunter aims to mimic this by automating the workflow of a typical security researcher (auditor). It uses an ensemble of machine learning layers to identify potential vulnerable functions in solidity codebases and uses program analysis layers (fuzzing and (or) formal verification) to verify (confirm or reject) the potential vulnerability.

Key features of Bug Hunter's output are deterministic nature of its output, high accuracy in terms of finding true positives and very low false positive rate (<5 % of total bugs found). This is enabled by the program analysis layers that verify (accept or reject) each potential vulnerability spotted by the ensemble of preceding machine learning layers.

## 2.2 Security Detectors

Bug Hunter possesses more than forty (40+) types of security detectors to identify vulnerabilities in solidity codebases. This includes all common types vulnerabilities classes such as Reentrancy, Integer Overflows, Rounding Errors, Tx Origin and etc. Detailed list of Bug Hunter's security detectors is available at https://docs.bughunter.live/.

# 3.0 Performance Notes

Bug Hunter is able to detect vulnerability with high accuracy in solidity codebases. We performed automated code reviews of variety of codebases types that exists in Ethereum ecosystem. To this end, we selected popular Yield farming and NFT projects for benchmarking of Bug Hunter. Detailed output reports generated by Bug Hunter are available at https://docs.bughunter.live/benchmarks.

We first report aggregate benchmark results for Yield Farming and NFT projects.

## 3.1 Yield Farming Projects – Aggregate Results

Yield farming projects encourage users to deposit cryptocurrencies into decentralized liquidity pools, offering rewards in the form of tokens or fees in return. These are most common types of projects on Ethereum ecosystem. They use smart contracts to automatically manage deposits, reward distribution, and facilitate decentralized trading or lending activities.

Table 1 below depicts yield farming protocols that we have analyzed using Bug Hunter along with relevant git hash of the project, Bug Hunter's results and reference results of its official audit. The results show that Bug Hunter was able to find up to 66% (best case results) of the vulnerabilities that were found by the security researchers during manual audit of the said protocol. The aggregate performance of Bug Hunter across all yield farming benchmarks was 46.5%.

Table 1: Comparison of Audit Findings and Bug Hunter Findings for Yield Farming Projects

| *Project* | *Audit Findings* | | *Bug Hunter Findings*[1] | | *Bug Hunter Accuracy (Bug Hunter Findings / Audit Findings)* |
|---|---|---|---|---|---|
| *Badger DAO* | Total | 21 | Total | 14 | 66% |
| | High | 1 | High | 1 | |
| | Medium | 2 | Medium | 2 | |
| | Low | 18 | Low | 11 | |
| *AAVE* | Total | 53 | Total | 25 | 47% |
| | High | 0 | High | 0 | |
| | Medium | 9 | Medium | 4 | |
| | Low | 44 | Low | 21 | |
| *Convex* | Total | 9 | Total | 5 | 55% |
| | High | 1 | High | 0 | |
| | Medium | 3 | Medium | 2 | |
| | Low | 5 | Low | 3 | |
| *Eigen Layer* | Total | 11 | Total | 3 | 27% |
| | High | 2 | High | 1 | |
| | Medium | 2 | Medium | 1 | |
| | Low | 7 | Low | 1 | |
| *Compound Finance* | Total | 20 | Total | 6 | 30% |
| | High | 1 | High | 0 | |
| | Medium | 7 | Medium | 3 | |
| | Low | 12 | Low | 3 | |
| *Aggregate - Yield Farming Protocols* | Total | 114 | Total | 53 | 46.5% |
| | High | 5 | High | 2 | |
| | Medium | 23 | Medium | 12 | |
| | Low | 86 | Low | 39 | |

### 3.1.1 Common Core Functions in Yield Farming and Bug Hunter's Performance

Each Yield Farming project aims to perform a unique function to impart its value to overall ecosystem. For example, AAVE introduced the concept of flash loans, fixed and variable interest rates and a variety of liquidation calls. EigenLayer introduced aspects of managed withdrawal credentials and enforces slashing for misbehavior. BadgerDao presented boosting & incentives where users staking BADGER or bBADGER can boost their vault rewards through a multiplier formula to increases their share based on their stake.

The above functions or characteristics are protocol specific. Apart from these protocol specific functions, there are common core function that exist in every protocol. For example, functions like Deposit / Supply, Withdraw / Redeem, Reward / Claim, Stake / Unstake and Emergency Withdraw / Pause functions are present in every protocol.

[1] Detailed Bug Hunter review reports are available at https://docs.bughunter.live/benchmarks

We now depict the Bug Hunter performance on common core functions in yield farming protocols. As stated earlier, these common core functions are present in every protocol. Results are depicted in Table 2.

Table 1: Bug Hunter's Performance on Common Core Functions of Yield Farming Protocols

| ***Project*** | ***Audit Findings on Common Core Functions*** | | ***Bug Hunter Findings on Common Core Functions[2]*** | | ***Bug Hunter Accuracy on Common Core Functions (Bug Hunter Findings / Audit Findings)*** |
|---|---|---|---|---|---|
| *Badger DAO* | Total Vulnerabilities | 16 | Total Vulnerabilities | 10 | 62.5% |
| *AAVE* | Total Vulnerabilities | 31 | Total Vulnerabilities | 18 | 58% |
| *Convex* | Total Vulnerabilities | 5 | Total Vulnerabilities | 3 | 60% |
| *Eigen Layer* | Total Vulnerabilities | 4 | Total Vulnerabilities | 3 | 75% |
| *Compound Finance* | Total Vulnerabilities | 13 | Total Vulnerabilities | 6 | 46% |
| *Aggregate – Yield Farming* | Total Vulnerabilities | 69 | Total Vulnerabilities | 40 | 58% |

Across the five protocols, Bug Hunter demonstrated strong performance in identifying vulnerabilities in common core functions as shown in Table 2. For example, in EigenLayer and Convex, Bug Hunter detected 75% and 60% of the issues respectively, while for Badger DAO and Aave it captured 62.5% and 58%. These common core functions are the most reused patterns in DeFi, and Bug Hunter's accuracy here shows its strength in handling battle-tested primitives.

## 3.2 NFT Projects

NFT projects create unique digital assets on blockchain networks like Ethereum, representing ownership of items such as digital art, collectibles, or virtual property. These projects allow creators to mint, market, and sell one-of-a-kind tokens whose authenticity and ownership are permanently recorded on the blockchain.

Table 3. depicts results of Bug Hunter analysis of popular NFT projects that we have analyzed along with relevant git hash of the project, Bug Hunter's results and reference results of its official audit. The results show that Bug Hunter was able to find up to 55% (best case results) of the vulnerabilities that were found by the security researchers during manual audit of the said project. The aggregate performance of Bug Hunter across all NFT benchmarks was 50%.

[2] Detailed Bug Hunter review reports are available at https://docs.bughunter.live/benchmarks

Table 3: Comparison of Audit Findings and Bug Hunter Findings for NFT Projects

| *Project* | *Audit Findings* | | *Bug Hunter Findings*[3] | | *Bug Hunter Accuracy (Bug Hunter Findings / Audit Findings)* |
|---|---|---|---|---|---|
| *Traitforge* | Total | 37 | Total | 18 | 48% |
| | High | 6 | High | 4 | |
| | Medium | 19 | Medium | 9 | |
| | Low | 12 | Low | 5 | |
| *Infinity* | Total | 27 | Total | 15 | 55% |
| | High | 11 | High | 6 | |
| | Medium | 9 | Medium | 6 | |
| | Low | 7 | Low | 3 | |
| *Opensea-seaport* | Total | 27 | Total | 13 | 48% |
| | High | 2 | High | 1 | |
| | Medium | 2 | Medium | 1 | |
| | Low | 23 | Low | 11 | |
| *NFTX* | Total | 22 | Total | 11 | 50% |
| | High | 4 | High | 2 | |
| | Medium | 8 | Medium | 5 | |
| | Low | 10 | Low | 4 | |
| *Unlock* | Total | 50 | Total | 26 | 52% |
| | High | 4 | High | 2 | |
| | Medium | 14 | Medium | 8 | |
| | Low | 32 | Low | 16 | |
| *Aggregate – NFT Projects* | Total | 163 | Total | 83 | 50% |
| | High | 27 | High | 15 | |
| | Medium | 52 | Medium | 29 | |
| | Low | 84 | Low | 39 | |

### 3.2.1 Common Core Functions in NFT Projects and Bug Hunter's Performance

Similar to yield farming protocols, NFT projects contain many common core functions. Some of these common core functions present in any NFT project may include Mint / Create, Transfer, Approve, Burn, Destroy, Metadata lookup, Enumeration, Rental extension, Circuit breaker, List, Offer, Cancel listing, fulfil order, Bid, Auction, Withdraw proceeds, and ownership management. These common core functions execute standard tasks that are often relevant to any NFT project.

Table 4 depicts the performance of Bug Hunter on common core functions in NFT projects. We observe that aggregate performance of Bug Hunter, that was 50% for complete NFT projects (see Table 3), now improves to 64% for common core functions in NFT projects. At individual project level, the best-case result is observed for Opensea-seaport, where Bug

[3] Detailed Bug Hunter review reports are available at https://docs.bughunter.live/benchmarks

Hunter is able to improve from 48% to 88% (i.e. able to find 88% of bugs that security researchers found in its official audit) for common core functions in the specific NFT project.

Table 4: Bug Hunter's Performance on Common Core Functions of NFT Projects

| ***Project*** | ***Audit Findings on Common Core Functions*** | | ***Bug Hunter Findings on Common Core Functions[4]*** | | ***Bug Hunter Accuracy on Common Core Functions (Bug Hunter Findings / Audit Findings)*** |
|---|---|---|---|---|---|
| *Traitforge* | Total Vulnerabilities | 16 | Total Vulnerabilities | 10 | 62.5% |
| *Infinity* | Total Vulnerabilities | 20 | Total Vulnerabilities | 14 | 70% |
| *Opensea-seaport* | Total Vulnerabilities | 9 | Total Vulnerabilities | 8 | 88% |
| *NFTX* | Total Vulnerabilities | 8 | Total Vulnerabilities | 4 | 50% |
| *Unlock* | Total Vulnerabilities | 17 | Total Vulnerabilities | 9 | 52% |
| *Aggregate – NFT Projects* | Total Vulnerabilities | 70 | Total Vulnerabilities | 45 | 64% |

## 4.0 Use Cases

Bug Hunter is optimally suited for teams engaged in the development of smart contracts using solidity language. Its sophisticated detectors can semantically match and detect vulnerabilities in all kinds of solidity-based underdevelopment code (including DeFi, NFT, Gaming projects etc.). A key feature of Bug Hunter is its deterministic output with very low (<5%) false positives. Hence, if it detects certain lines of code as vulnerable, they likely are.

Organizations preparing for a security audit will derive significant and immediate value from Bug Hunter. By proactively deploying Bug Hunter in their development workflow, teams can strategically direct the focus of audit firms or contest platforms towards the most challenging-to-detect vulnerabilities. This approach ensures a maximized return on investment for security expenditures.

Integrating Bug Hunter in the development workflow provides the equivalent of hiring a dedicated security researcher who is perpetually enhancing their expertise and actively seeking new vulnerabilities within the codebase. Even in the absence of code modifications, the threat landscape is dynamic, and our detectors will effectively flag new issues as their capabilities expand.

[4] Detailed Bug Hunter review reports are available at https://docs.bughunter.live/benchmarks