FREE GUIDE

BUG BOUNTIES: LIVE PROGRAMMES AND HUNTING WITH AI

Live programmes checked 1 November 2026, how to use AI the legal way, and how four hunters found critical flaws and got paid.

LIVE PROGRAMMES + THE AI PLAYBOOK

Checked 1 November 2026 · free · no earnings promised

The 20-second version

- Bug bounties pay you for finding security flaws in code and reporting them privately to the owner.
- Below: live programmes checked 1 November 2026, including one with a stated top reward of $3,000,000.
- AI helps you learn and read code faster. Sending reports AI wrote without proving them gets accounts banned.
- Four hunters in this guide received five payouts totalling $20,050,000. That is the very top of the field.
- The honest catch: it takes months of study, and nobody is promised a single dollar.

Live bug bounty programmes, checked 1 November 2026

Each amount below is the programme's own stated top reward or range on the day we checked. Programmes change, pause and close, so open the link and read the rules before you start. Most pay the top amount only for the most serious ("critical") bugs.

Crypto code (smart contracts) on Immunefi

Immunefi listed 156 programmes when we checked on 1 November 2026. Here is a sample of nine with their stated top rewards:

Programme	Top reward	Worth knowing
Ethena	$3,000,000	Critical smart-contract bugs pay from $100,000
Lombard Finance	$500,000	Charges a non-refundable fee per report
DeXe Protocol	$500,000	Page last updated November 2024
SSV Network	$250,000	Pays in its own SSV token
ENS	$250,000	Critical bugs pay from $10,000 up, depending on funds at risk
Zest Protocol V2	$100,000	
StackingDAO	$100,000	
The Graph	$50,000	
Hedera	$30,000	

HackenProof lists more crypto programmes.

Websites and apps

Platform	Programme	Reward as stated
Bugcrowd	Vinted	$75 to $8,000
Bugcrowd	Rapyd	$100 to $7,500
Bugcrowd	Lightspeed Hospitality	$250 to $7,500
Bugcrowd	MATLAB Online	$200 to $7,000
YesWeHack	Infomaniak	€100 to €7,000
YesWeHack	CoinDCX	$50 to $2,000
YesWeHack	DataDome	€200 to €1,000

HackerOne's public programmes include Shopify, GitHub and Uber; each programme page shows its current rewards. Intigriti's public programmes include Adobe and NVIDIA.

Audit contests

A project opens its code for review for a fixed period and pays findings from a prize pool. New contests open most weeks on Code4rena, Sherlock and Cantina. They are good practice before hunting big bounties. Gerhard Wagner lists himself as a Watson on Sherlock, the name Sherlock uses for its auditors.

What a bug bounty is

Crypto projects write their rules in code. They publish this code and invite people to look at it. They promise a reward to anyone who finds a serious flaw and reports it privately instead of stealing money with it. This is a bug bounty.

You will hear the term "white-hat hacker". This means someone who finds security flaws to fix them, not to exploit them.

The code they look at is usually a "smart contract". A smart contract is the code that runs on a blockchain. It holds the rules and the money.

Many of the biggest bounties involve a "bridge". A bridge is software that moves money between blockchains.

When you find a bug, most programmes want more than your word for it. They ask for a "proof of concept": code you write to prove the bug actually works.

Finally, projects do not pay for every bug. The code must be "in scope". In scope means the exact list of code the project has promised to pay for if you find a bug in it.

The four hunters and the bugs they found

SATYAOX AND WORMHOLE

$10,000,000

satya0x is a security researcher who goes by that pseudonym. Wormhole is a bridge. On 24 February 2022, satya0x privately reported a critical bug in the Wormhole core bridge contract on Ethereum. Immunefi's write-up calls it "an upgradeable proxy implementation self-destruct bug that could have led to a potential lockup of user funds".

In plain words, Wormhole's code was built to be upgradeable. It runs through a proxy contract that points to the real logic contract. The logic contract had been left uninitialised: an earlier bug fix had undone its setup. Whoever set it up first could take control of it and destroy it. This could have frozen the money held by the bridge. Wormhole verified and fixed it the same day it was reported. No user funds were lost.

Wormhole paid satya0x $10,000,000 through Immunefi, which called it a record bug bounty. The payout was announced on 20 May 2022. Immunefi said: "Wormhole is sending a clear message with this

payout to the best, most talented whitehats on the planet that if they responsibly disclose security vulnerabilities to Wormhole, they'll be well taken care of." The skill it took was knowing how upgradeable proxy contracts work on Ethereum and the classic mistakes in setting them up.

PWNING.ETH AND AURORA, THEN THREE POLKADOT NETWORKS

$6,000,000 + $1,000,000

pwning.eth is a researcher who goes by that handle. On 26 April 2022, pwning.eth reported a critical bug through Immunefi. It was an "infinite spend" bug in Aurora, in the code that moves ETH out of Aurora to NEAR. If exploited, it "could have led to a direct loss of 70k ETH and $200m in other assets". No user funds were lost and Aurora patched it quickly.

In plain words, Ethereum code can call other code in different ways. One way is called DELEGATECALL. This runs the other code without actually sending any ETH along. Aurora had a built-in contract for withdrawing ETH to NEAR. By reaching it through DELEGATECALL, an attacker could make Aurora believe ETH had been sent when it had not. They could withdraw money that was never paid in. Aurora paid pwning.eth $6,000,000 in AURORA tokens. This was "the second largest bounty payout in history" at the time.

Later, pwning.eth reported a second critical bug. This bug was in Frontier. Frontier is the code library that lets Polkadot networks run Ethereum-style apps. It affected three networks at once: Moonbeam, Astar and Acala. It made it possible to create wrapped tokens out of nothing. Immunefi described this as "minting valid but depegged wrapped tokens". The estimated potential damage was about $200m

across the three networks. pwning.eth was awarded $1,000,000 for it. Parity is the company that wrote the Frontier library. Parity contributed $250,000 towards that $1,000,000 despite having no bug bounty with Immunefi. Immunefi called it "one of the most detailed and in-depth vulnerability report cases to exist in blockchain history". The skill it took was deep knowledge of how the Ethereum virtual machine handles calls and value, and of how bridges and Ethereum-compatible chains are wired together.

GERHARD WAGNER AND POLYGON

$2,000,000

Gerhard Wagner wrote up the bug himself, under his own name. His profile describes him as "Bounty Hunter @immunefi | Watson @sherlockdefi".

Here is how it started in his own words: "I thought I was out of the security game for a while now and that my interests have moved on to other fields. Last week I came back from an extended holiday, and refreshed I looked around on CT on what had happened. Somehow I stumble over the Polygon announcement that they launched their bug bounty program on Immunefi. I have worked on fraud-proof systems before, specifically on Plasma." CT means Crypto Twitter.

His method in his own words: "Taking a quick look at the bounty page, I skim through the published contracts to find the contract that locks up the funds on L1 when users move funds to and from the Polygon network." When he wrote his post, that contract held tokens valued at close to one billion USD. He said: "I am intrigued and down I fall that rabbit hole."

He reported it on 5 October 2021. The bug was in the Polygon Plasma Bridge. A withdrawal is called an exit. An exit could be claimed again and again because one piece of the proof came straight from user input. This piece is the branch mask. It could be written in many different ways that all passed the check. Immunefi said an attacker could "exit their burn transaction from the bridge multiple times, up to 223 times". Immunefi put the amount at risk at around $850M. Polygon confirmed the bug 30 minutes after Immunefi escalated it. The issue was mitigated within one week.

Polygon paid the maximum on its bounty page: $2,000,000. Immunefi called it the highest bounty ever paid out at the time. His lesson, in his words: "It's OK to use [existing] building blocks when you write smart contracts, but you must understand all implications of doing so. At the end of the day, it's your code; it does not matter if you or someone else wrote it." His guess is that the bug came from a third-party library the project used without fully understanding it. The skill it took was earlier work on the exact kind of system, Plasma and fraud proofs, plus careful reading of how the proof data was encoded.

ALEXANDER SCHLINDWEIN AND BELT FINANCE

$1,050,000

Alexander Schlindwein is named publicly. Belt Finance describes him as "a notable whitehat hacker". He found a critical bug in Belt Finance. This is a stablecoin swapping and yield protocol on Binance Smart Chain. The bug "put more than $10 million of capital at risk". It was patched and no user funds were lost. The announcement did not give the technical details.

He was paid $1,050,000. This was $1,000,000 from Belt Finance plus $50,000 from Binance Smart Chain's Priority ONE bounty-matching programme. It was announced on 30 September 2021 as "the biggest single bug bounty payout ever; publicly disclosed". It was "larger than payouts from giants such as Apple, Microsoft, or Firefox".

Earlier that year he had received a bounty in Armor tokens from Armor Fi for a critical bug. He also received $800,000 in TRIBE tokens from Fei Protocol for another critical bug. So he was a repeat hunter. The skill it took was the same core skill as the others. He was reading DeFi smart contracts for flaws. The repeat payouts show it was a practised skill, not one lucky find.

FOUR HUNTERS, FIVE PAYOUTS

$20,050,000

$10,000,000 + $6,000,000 + $1,000,000 + $2,000,000 + $1,050,000 = $20,050,000.

All five were paid through bounty programmes run with Immunefi. None of these hunters exploited the bug. Each reported it privately and was paid a bounty.

The pattern they share

None of this is a formula, and nothing here means you will find the same. But in these five finds, a few things keep showing up. This part is what these hunters actually did; the skill path further down is a free route to the same kind of skill, not the courses they took.

- **Experience came first.** Gerhard Wagner had worked on Plasma and fraud-proof systems before. Alexander Schlindwein had been paid earlier bounties by Armor Fi and Fei Protocol. pwning.eth made two critical finds in two different systems.
- **They looked where the money sits.** Every bug here put large sums at risk: Wormhole's core bridge contract, Aurora's withdrawals to NEAR, Polygon's Plasma bridge (around $850M inside when the bug was reported, per Immunefi), wrapped tokens on three networks, more than $10 million at Belt Finance.
- **They started from the published code.** In Gerhard Wagner's own account, he skimmed "the published contracts" on Polygon's bounty page and went straight for "the contract that locks up the funds".
- **The bug was a broken assumption.** A contract left uninitialised after an earlier fix. A call type that sends no money. A user-supplied value that could be written many ways. (Belt Finance's bug was not described in its announcement.)
- **They reported privately instead of exploiting.** All five were paid through bounty programmes run with Immunefi. The projects paid the bounties, and sometimes others added to them: Parity put $250,000 towards pwning.eth's $1,000,000, and Binance Smart Chain's Priority ONE added $50,000 for Belt Finance. Polygon paid the maximum listed on its bounty page.

The skill path

This takes months, not days. You move from zero to your first report in stages.

Stage 1: Learn the language

You need to read smart contracts. Start with CryptoZombies. This is a beginner Solidity course built as a game. Then read the Solidity documentation. Solidity is the main language of Ethereum smart contracts. Done looks like understanding how basic contracts are written and deployed.

Stage 2: Set up your tools

Install Foundry. This is the toolkit hunters use to write and run tests and proofs of concept on a local copy of the chain. Done looks like running a local test environment.

Stage 3: Practice hacking

Play The Ethernaut by OpenZeppelin. These are hacking puzzles in a browser game. Move to Damn Vulnerable DeFi. These are harder hacking challenges based on DeFi. Done looks like solving these puzzles without looking at the answers right away.

Stage 4: Learn security and auditing

Take the Cyfrin Updraft Smart Contract Security and Auditing course. Its page says the course is "completely for free" and marks it advanced: it lists Blockchain Basics, Solidity Smart Contract Development, Foundry Fundamentals and Advanced Foundry to take first, on the same site. Done looks like finishing the course and understanding common flaws.

Stage 5: Study real hacks

Read Immunefi's Web3 Security Library. This is a free reading list on web3 security. Search Solodit for real smart contract vulnerabilities from audits. Look at DeFiHackLabs to reproduce real DeFi hacks using Foundry. Check the Rekt leaderboard for the biggest crypto hacks ranked. The write-ups behind these cases (listed under Sources at the end) are great study material: Immunefi's reviews of the Wormhole, Aurora, Polygon and Frontier bugs, and Gerhard Wagner's own post. Read them slowly and rebuild the bug on your own machine. Rebuilding a public bug is practice only: reporting it is against the rules. Done looks like rebuilding a real past bug on your own machine.

Stage 6: Your first report

Enter an audit contest on Code4rena, Sherlock or Cantina before you hunt the big bounties on Immunefi. Read each programme's scope and rules first. Done looks like your first finding submitted, with a proof of concept that runs on your own machine.

Hunting code holes with AI, the legal way

AI is now part of bug hunting. It can help you learn and read code faster. Platforms welcome that help, but an AI-written claim you have not proved can get your account suspended or banned.

What the platforms say

- **Immunefi:** Its rules prohibit AI-generated or scanner reports that leave out the required explanation of how the bug affects the listed asset. Its reporting guide says it will "warn or ban" people for submitting AI-generated or automated reports that waste reviewers' time. It says to test proofs of concept on a local network, such as a mainnet fork.
- **HackerOne:** Its Code of Conduct encourages responsible AI use for learning, finding bugs, developing proofs of concept and improving report quality. You still own every claim. Unverified bugs, invented endpoints and unsupported claims about harm are prohibited.
- **Bugcrowd:** Its AI submission policy encourages experimenting with AI, but says the researcher is responsible for accuracy. If 10 or more of your reports in a row are invalid and traced to unvalidated AI or automated activity, your account may be suspended for 30 days.
- **curl:** Maintainer Daniel Stenberg said AI slop was about 20% of submissions in the first half of 2025. SC World reported that curl later ended its bug bounty programme amid an influx of AI-generated contributions. Bad reports cost real people time.

AI has found real flaws. Google's Big Sleep agent found an exploitable stack buffer underflow in SQLite, fixed the same day. XBOW said its AI system reached the top of HackerOne's leaderboard in mid-2025. OpenAI's Aardvark is now Codex Security. These are efforts by big teams with expert staff, not evidence that a beginner can press a button and earn a bounty.

Where AI genuinely helps a beginner

- **Read unfamiliar code:** Paste a small function and ask, "Explain this line by line. Which lines change stored balances?" Check its answer against the code.
- **Map who controls money:** Ask, "What does this contract do, and which functions can move funds?" Trace those functions yourself before trusting the summary.
- **Turn a hunch into a test:** Ask, "How could I test whether this withdrawal check can be bypassed?" Write and run the test yourself with Foundry on a local fork, within the programme's rules.
- **Understand tool warnings:** Run Slither on Solidity or Vyper code, or Semgrep to search code for bugs. Ask, "What does this warning mean, and what would prove it matters here?" A warning alone is not a finding.
- **Check your writing:** After you have proved the issue, ask, "Is any step in my report unclear or unsupported?" Verify every suggested change yourself.

Where it gets you banned

Do not send a report that AI wrote if you have not reproduced the bug. Do not include invented code paths, endpoints or effects. Immunefi's rules say testing on mainnet or a public testnet is grounds for an immediate and permanent ban. Stay within the exact scope and testing rules of the programme: HackerOne also forbids unsafe testing, excessive traffic and out-of-scope activity. A local fork is for testing your proof without touching live contracts.

A simple loop

1. Pick a programme and check that the code you want to study is in scope.
2. Read its rules, including what testing and proof it permits.
3. Read the code with AI as a tutor. Check its explanations against the code.
4. Prove the bug yourself on a local fork, within those rules.
5. Write the report yourself, with a working proof of concept and a clear explanation of what could happen to the in-scope asset.

AI can make study faster. It cannot make you a hunter overnight, and no earnings are promised.

The rules that get you banned

From Immunefi's rules.

- "Any testing with mainnet or public testnet contracts. Testing on mainnet or public testnet is grounds for an immediate and permanent ban." Test a project's contracts only on your own local copy. (Practice games like Ethernaut are built to be attacked; a project's live contracts are not.)
- Prohibited: "Exploiting/attacking or threatening to exploit/attack a project on Immunefi".
- Prohibited: "Misrepresenting severity: claiming that a bug report is critical when it clearly is not" and claiming a bug hits code that is not in scope.

- Prohibited: "Automated testing of services that generates significant amounts of traffic".
- Prohibited: "Reporting a bug that has already been publicly disclosed".
- Prohibited: "Submitting AI-generated/automated scanner bug reports that lack the required information regarding the vulnerability's impact on the reported asset".
- Prohibited: "Publicly disclosing a bug report--or even the existence of a bug report for a specific project--before it has been fixed and paid".
- Prohibited: "'Beg bounty' behavior, i.e. begging for a bounty reward that is not owed".
- Prohibited: submitting with no proof of concept or an incomplete one when the programme requires one.
- Prohibited: failing to provide KYC information in reasonable time when it is needed for payment, which "may result in forfeiture of payment", and providing inauthentic KYC information.

Violations can mean "a temporary suspension or permanent ban from the Immunefi platform", and for whitehats "1) forfeiture and loss of access to bug reports, and 2) zero payout".

The honest part

Record payouts are not typical. The numbers above are the very top of the field.

Payment timing varies. At Polygon, the whole fix, payout included, was done within a week, per Immunefi. Wormhole's $10,000,000 was

announced on 20 May 2022, about three months after the 24 February report.

Payment can come in the project's own tokens, whose price moves. pwning.eth was paid in AURORA tokens. Alexander Schlindwein's earlier bounties came in Armor tokens and TRIBE tokens.

It needs real skill. Gerhard Wagner had worked on Plasma and fraud-proof systems before. Alexander Schlindwein was a repeat hunter who had received earlier bounties.

There is no promise you will earn anything. You might study for months and find nothing.

Treat any reward as income and check your local tax rules.

Your first week

1. Start CryptoZombies and complete the first few lessons to see what Solidity looks like.
2. Read the Solidity documentation introduction to understand what smart contracts do.
3. Bookmark the Cyfrin Updraft course so you know where to go when you are ready to learn auditing.

This is one of the real stories we cover. Follow for the next one.

Sources

Every number and quote in this guide comes from the pages linked in the text and below. The four hunters' pages were checked on 28 October 2026; the live programmes and the AI rules and cases on 1 November 2026.

- satya0x and Wormhole: Immunefi's Wormhole review, The Block, 20 May 2022
- pwning.eth and Aurora: Immunefi's Aurora review, Aurora's own post
- pwning.eth and Moonbeam, Astar, Acala: Immunefi's Frontier review
- Gerhard Wagner and Polygon: his own post, Immunefi's Polygon review
- Alexander Schlindwein and Belt Finance: BNB Chain blog, 30 September 2021
- The rules: Immunefi rules
- Live programmes: each programme's own page or its platform's listing, checked on 1 November 2026 (links in the tables above).
- AI rules: Immunefi's rules, HackerOne's Code of Conduct, Bugcrowd, 10 March 2026, checked on 1 November 2026.