# Episode Book

EPISODE #6455

# Turning Open-Source Triage into Subscription Credit

| | |
|---|---|
| PUBLISHED | 4 November 2026 at 13:22 UTC |
| DURATION | 26:54 |
| LISTEN | myweirdprompts.com/episode/feedcoin-open-source-triage-credits-17541/ |
| HOSTS | Corn, Herman, Hilbert |
| SCRIPT MODEL | z-ai/glm-5.3 |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.26 |
| SOURCES | 25 cited by the research (listed at the end) |
| BOOK GENERATED | 5 November 2026 at 07:03 UTC |

## SYNOPSIS

Heavy users of semi-open developer tools often act as unpaid triage teams and QA departments, diagnosing critical bugs and shaping product roadmaps while paying full subscription rates. Examining the FeedCoin proposal reveals that while companies like Loophole, Orkosi, and Nexu offer fragmented credits for reports or code, an interoperable renewal discount standard does not exist. Exploring Turso and Storj shows why vendors hesitate: financial incentives reliably trigger overwhelming waves of automated AI slop.

# TRANSCRIPT



### Corn

The part nobody prices is the triage. A public repo is a support channel, a design department, and a stress test running twenty-four hours a day, and the people staffing it are the ones paying for the product.



### Herman

And they're paying full freight. Corn, Daniel's written up a whole idea about this, he calls it FeedCoin, and his framing is exactly the one you just landed on: the heaviest users of a partly-open commercial product find the bugs, diagnose them, send fixes, shape the features that ship, and then renewal comes around and they pay the same as someone who never opened the tracker once. He noticed it as a heavy Claude Code user, though he's at pains to say Claude Code is the example, not the target.



### Corn

He wants to know if anyone's tried it, how it went, and what it would take to make it real. Which is really two questions — has anyone built the pieces, and has anyone built the combination.



### Herman

And the answer to those is different, which is what makes this worth an episode.



### Corn

FeedCoin, then. Despite the name, no coin. No token, no market, no wallet.



### Herman

None of it. It's a credit metric — an entry in a vendor's ledger with nothing to trade. The vendor issues coins when a contribution passes review, weighted by type. A bug fix earns more than a validated idea, a validated idea earns more than one that was only considered. When the contributor's subscription comes up for payment, the balance is applied as credit against the fee. The standard covers only the unit and the tiers. Each vendor keeps control of what a coin is worth, any cap, any expiry, how it shows on the invoice.



### Corn

Which is the right seam. Standardize the thing that has to be shared, leave the commercial dials local.



### Herman

And the novelty claim is precise, which I appreciate in a proposal. Every mechanical piece already exists in production somewhere. What doesn't exist is a standardized unit — one that crosses vendors — and the tying of that unit to the renewal invoice. The standard is the idea, not the mechanism. He's not proposing new plumbing. He's proposing an agreement about what a unit of contribution is worth recording.



### Corn

A shared vocabulary for something everyone's already doing privately. So let's go find out who's doing what, because the answer is more interesting than yes or no.



### Herman

Credit for bug reports is live and honestly fairly common. Loophole, the tunneling service, pays promotional account credit tiered by severity — ten to fifteen dollars for cosmetic bugs, up to two hundred, sometimes five hundred plus, for critical security issues. And it's awarded automatically once the report qualifies.



### Corn

Automatically. There's a word carrying a lot of weight.



### Herman

In their case it mostly means once the report's confirmed eligible, no negotiation per bug. Then there's Orkosi, which runs a four-level severity scheme, and the detail I love — the detail FeedCoin needs — is how the level gets decided. Their triage judges severity "from what your report describes and the evidence around it — never by the words you choose." That is an explicit anti-self-grading rule. You cannot talk your way up the tier ladder. FeedCoin's review gate needs exactly that sentence somewhere in the spec.



**Corn**

Because the moment the reporter grades the report, you've built a machine that prints its own money.



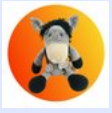
**Herman**

Credit for code contributions is live too, and Nexu is the closest structural match to Daniel's tiering. Points are assigned by GitHub issue label — a help-wanted issue is worth two thousand points, roughly twenty dollars in Nexu usage credits, a good-first-issue or unlabelled one is five hundred, about five dollars. The flow is: pull request linked to the issue, dev team reviews it for quality and completeness, and after merge the team confirms and awards the points.



**Corn**

Label as tier. Merge as review gate.



**Herman**

Exactly the skeleton of FeedCoin, minus standardization and minus the renewal framing. Though their docs include one line every vendor will insist on: "The team reserves the final right of interpretation on point awards."



**Corn**

The escape hatch. Keep that one in your pocket, it comes back later.



**Herman**

GitLab goes further — a full points-and-levels system at contributors dot gitlab dot com. Twenty points for a merge request created, sixty for one that's merged. Five levels, twenty-five points up to thirty-seven thousand plus, with the first three earning five, fifty, and a hundred and fifty credits for the contributor store.



**Corn**

Wait. Points for a merge request created. Not merged. Created.



**Herman**

Yes.



**Corn**

So the drift starts on day one. Open a PR, collect twenty points, close it. That's the "rewarding people for filing X number of bugs" problem wearing a badge.



**Herman**

And it's a real warning for the standard, because it happened in the friendliest possible environment. GitLab's community is about as good as they come, and the points still reward the action rather than the outcome. FeedCoin's tiers are outcome-based — validated beats considered, fixed beats validated — which is the fix. But note what the fix costs: somebody has to do the validating and the considering.



**Corn**

The vendor.



**Herman**

Hold that thought too, it's the whole second half of this episode.



**Corn**

And the other bucket — whole subscriptions for open-source people. That's eligibility-based, isn't it.



### Herman

Entirely. GitLab for Open Source gives qualifying projects free Ultimate, fifty thousand compute minutes, annual renewal — gated on OSI-approved licenses and no profit-seeking. Appwrite's program gives free Pro to projects with fifteen-plus contributions and a hundred stars, reviewed annually. GitHub says over two hundred and eighty thousand maintainers have access to benefits including free Copilot Pro. All of these are binary grants gated on status. You're either in the club or you're not. Nothing scales with what you actually did on the product.



### Corn

A door, not a meter.



### Herman

Which brings me to the most directly relevant precedent, because it's Anthropic's own. The Claude for Open Source Program. Six months of free Claude Ultra to maintainers and library authors — eligibility routes include five hundred plus dependent repos, two hundred thousand combined monthly downloads, or a hundred plus PRs merged into repos you don't own in the last twelve months.



### Corn

That last route is interesting. A hundred PRs into other people's repos. They're measuring contribution, just not on their own product.



### Herman

And the expiry behavior is documented, which is the part that matters for FeedCoin. At the end of the six months: "If you had a paid plan, it resumes at your prior plan and rate unless you cancel; otherwise your account returns to the free plan." Read that again as an engineer. The vendor already has the machinery to grant and revoke subscription value, on a schedule, with a fallback. The dial exists. It's just tied to eligibility, not to a per-contribution ledger, and it doesn't compound.



### Corn

So the gap is thinner than it looks. It's not build-the-plumbing, it's re-point-the-plumbing.



### Herman

And the Claude Code loop itself — the labor Daniel's describing — is demonstrable, not hypothetical. Issue nine-one-eight-seven-zero, the Mods thread. September third, an Anthropic staffer opens it asking for feedback on an internal proposal and says, verbatim, "If you want this, please engage here — the response from the community likely dictates whether this ships or not."



### Corn

That's the vendor saying out loud that the community is the design department.



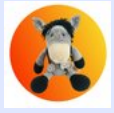
### Herman

September ninth, a community update thanks people for "the insightful, high-signal feedback that has materially shaped our design." October first: "We're live!" Four weeks from proposal to shipped feature. And the comments — Corn, the comments are the unpaid labor in full costume. One commenter reports forty-two active hooks across nineteen events, and then does threat modeling on the proposal itself, including the observation that overriding a category today silently drops Claude Code's sixty-five default soft-deny rules.



### Corn

Forty-two hooks. That man has done more QA than some QA departments.



### Herman

And he's citing his own hook count as credentials, which is telling. In that community, your configuration is your résumé.



### Corn

Okay. So we've got Loophole, Nexu, GitLab, Orkosi, the eligibility grants. The pieces are everywhere.



**Herman**

And the combination is nowhere. That's the negative finding and it anchors everything: no standardized, cross-vendor, review-gated contribution-credit unit exists as of today. Everything is proprietary — Nexu points, GitLab store credits, Loophole promotional credits — each with its own unit, its own weighting, zero interoperability. And no case was found of any vendor applying contribution credit against a subscription renewal invoice as policy. The renewal-invoice application, which is Daniel's actual innovation, has literally never been done.



**Corn**

Then the obvious next question is why nobody has combined them.



**Herman**

And the answer is a graveyard.



**Hilbert**

Medieval Pottage Facts. Before the plague, a common labourer might earn about a penny a day. A penny bought bread and pottage and not much else. It was a living, in the way a roof is a living.



**Corn**

A penny a day and pottage. The original usage credits.



**Herman**

Turso is the case study, because they wrote the whole thing up and it's almost a controlled experiment. Thousand-dollar bounty for any demonstrable data-corruption bug, ran nearly a year. Total legitimate payouts: five individuals. And the outcomes were superb — one became a core contributor to Turso's own simulator, one was hired outright, one paired the simulator with formal methods and found more than ten bugs in SQLite itself. Not Turso's database. SQLite.



**Corn**

Five payouts, five wins. Then what happened.



**Herman**

Then — their words — "an army of slop was released overnight." A thousand dollars turned out to be exactly the price at which you point an LLM at Turso and let it run. The documented submissions are almost poetic. A PR that manually injected garbage bytes into the database header and then argued the database was corrupted. A PR claiming a critical vulnerability, quote, "that allows for the execution of arbitrary SQL statements" — in a SQL database.



**Corn**

The killer feature, apparently.



**Herman**

Turso built a vouching system that auto-closed suspected bot submissions. The bots started opening issues questioning the closing and requesting manual inspection. Same PR reopened under a different account moments after closing. And the economics they stated plainly: "It costs the slopmaker perhaps a minute to generate their submission. But it costs us hours to read, understand, and engage with them."



**Corn**

A minute to make, hours to read. That's the whole slop economy in one line.



**Herman**

And their conclusion is brutal for anything FeedCoin-shaped: "We just don't believe that a financial incentive of any kind works well with an open system. We have to either close the system, or get rid of the incentive." They got rid of the incentive.



**Corn**

And they're not alone.



**Herman**

Not remotely. Storj ended its paid bounty, first stated reason: "The volume of AI-generated reports. Most of what we received was produced by automated tools and language models. These reports were often generic, unverified or simply wrong." TYPO3 discontinued its public program in September — budget exhausted, unsustainable in its current form. Minut paused theirs in April of last year and says it will not resume. Sensay discontinued, InstiKit closed entirely. And Kinopio, the solo developer, puts a number on it: over ninety-five percent of the bug reports he receives are automated LLM scans from non-users.



**Corn**

Ninety-five percent from people who don't even use the thing.



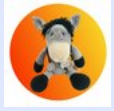
**Herman**

Non-users. They're not even customers. The slop machine doesn't have a subscription to discount.



**Corn**

Which is the obvious counter, isn't it. Let me play Daniel's advocate for a second. Every one of those programs died paying cash, or close to it. FeedCoin's credit is only redeemable against a subscription the contributor already pays for. You can't withdraw it. A bot farm that doesn't use the product has nothing to extract.



**Herman**

That's the strongest argument for the design, and I'd give it real weight. The extraction surface is narrower. But Turso's evidence cuts the other way, and I keep coming back to it: any non-zero value attracts the slop machine, and a subscription discount is worth actual money to a heavy user. The question is whether "non-tradable" reduces the incentive or just makes it less liquid.



**Corn**

Mm. A voucher is still money, Herman. It's just money with a dress code.



**Herman**

Ha. Though there's a second defense Daniel has that Turso didn't: the review gate. Turso paid on demonstrated bugs. FeedCoin pays on reviewed outcomes — and here's the part I think the proposal missed. What actually held the line at Turso, before the flood, wasn't the reward structure at all. It was, quote, "the requirement that the simulator had to be extended to demonstrate the bug — just pointing out the bug was not enough."



**Corn**

So the filter was a reproducibility bar.



**Herman**

A proof burden. You had to build the demonstration, not just allege the flaw. FeedCoin's tiers reward outcome — fix, validated, considered — but what filtered the slop was evidence. A standard that specifies the unit and the tiers but not a minimum evidence bar may have specified the wrong half of the problem.



**Corn**

So the spec needs a fifth thing. Not just how much a coin is worth, but what you have to show to get one at all.



**Herman**

And that brings us to implementation, which is where the surprise is, because it's the easy part. Billing systems already do this. Stripe supports credit grants applied at invoice finalization, with explicit eligibility rules — the invoice period has to fall after the grant's effective date, before its expiry if one's set, the grant has to have available balance at finalization, currency has to match. Recurly does manual credit adjustments that reduce what a customer owes, billable immediately or rolled into the next invoice. Sequence draws down soonest-expiring grants first. Vindicia deducts available credits of the same type before billing. Pabbly documents the plain case: a hundred-dollar-a-month customer given fifty dollars of credit is charged fifty.



**Corn**

Expiry, caps, currency, invoice presentation — the four things Daniel says each vendor keeps control of — are already configurable fields.



**Herman**

Checkbox and a date picker. The standard would specify a source for the grants, not new plumbing. There is one real constraint: Stripe restricts credit grants to subscription items using metered prices reporting usage, and Claude Code is a flat-seat product. So a vendor might need Recurly-style adjustments rather than native Stripe grants. That's a migration decision, not a research problem.



**Corn**

Implementation solved. So what's left is the hard part.



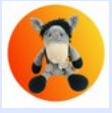
**Herman**

Adoption, and it splits two ways. Vendor side first: the vendor bears all the triage cost, which is precisely the cost Turso said it could not bear. Review-gated issuance means every submission gets a human with judgment, and we just established that over ninety-five percent of what shows up can be machine-generated noise. You're asking the party with the least appetite for the work to do the most of it. And Nexu's final-right-of-interpretation clause — the escape hatch you pocketed earlier — every vendor will want it, and it's also what makes a standard toothless if it's the only enforcement.



**Corn**

A standard where every member can overrule the standard.



**Herman**

It's the SPDX problem, sort of. Common vocabularies survive exactly as long as they're more convenient than they are binding. Contributor side, the problem is timing. The value only lands at renewal — up to twelve months of deferred reward. A bounty pays on merge. FeedCoin pays... eventually, on a bill you were going to pay anyway. That's a weak loop for driving behavior, though Daniel would say it's a strong loop for retention, and honestly the retention argument is the better one.



**Corn**

That's the thing, though. Has anyone measured whether these programs retain anyone?



**Herman**

No. And that's the quietest gap in the whole file. No published data on whether contribution-credit programs reduce churn. No participation rates, no redemption rates, no figures on what fraction of credits went to junk. GitLab's blog asserts recognition brings first-timers back, but gives no numbers. The business case is asserted everywhere and measured nowhere.



**Corn**

Which is odd, because it'd be cheap to measure. One vendor, one program, one cohort.



**Herman**

Now the counter-argument Daniel should hear, and it's not nothing. "Same price as someone who never reported anything" has a reply. A vendor could say the public issue tracker is already the payment — you got your bug fixed, that's the service you're paying for. Diagnostic labor is a support interaction, not a contribution, and it's priced into the subscription. Dries Buytaert's makers-and-takers framing sharpens it: the open-source risk he worries about isn't contributors being underpaid, it's takers out-competing makers. And von Hippel's whole free-innovation point is that users do this work for use, learning, and fun — which raises the uncomfortable question of whether paying people changes why they do it.



**Corn**

The crowding-out worry. Pay the volunteers and watch the volunteers leave.



**Herman**

Sometimes. The literature on that is mixed, and I'd flag Yahoo — twelve fifty of store credit for a security bug, back in twenty thirteen, and it was mocked so widely it arguably did more harm than paying nothing.



### Corn

Twelve fifty. That's not a bounty, that's a parking validation.



### Herman

LEGO Ideas is the counter-example — one percent of net sales, but only if the idea ships. Paying for validated outcomes, never for submissions. Daniel's tiering instinct is sound; the amounts are where it lives or dies. And his own doubts are the right ones, honestly — who gets the credit on an employer-paid seat, tax treatment of credit earned for work, whether a precise diagnosis with no fix outranks a small patch. Those are unsolved and they're social, not technical.



### Corn

The honest verdict lands something like this. The standard is a real gap. The mechanism is buildable — the billing layer is done, the tiering exists at Nexu, the anti-self-grading rule exists at Orkosi. The thing that kills it isn't any of that. It's that review-gated issuance puts the entire cost on the party with the least incentive to bear it, in an environment where a bad submission costs about a minute to produce.



### Herman

Every failed program in the file died of exactly that.



### Hilbert

I've kept a ledger on this, you know. A proper one. Notebook, columns, dates. Every bug I've ever reported to software I paid for, and next to each one, what came back. Not what I think I'm owed — what actually came back. One column is months between report and fix. One column is months between fix and me noticing. Corn will like that second one. Corn: You track how long it takes you to notice your own bug got fixed?



### Hilbert

Someone has to. Both columns matter or neither does. And your level's a bit hot on my left, Herman, you dipped when you said Turso.



### Herman

Sorry. Turned into the mic on that one.



### Hilbert

Anyway. Herman: So what does the ledger show? Hilbert: The fixes mostly arrived. That's the part your episode gets half right. What never arrived was any acknowledgement that I'd been the one who found it. I've never once asked a vendor for anything, and I've never shown the notebook to a soul — I read entries out when I choose to, and I choose carefully. FeedCoin wants to knock a few shekels off the invoice. The invoice isn't the column I care about. I want the bill to say something. Corn: What's the right number of months, by the way? Between fix and you noticing? Hilbert: There is a right number. I'm not saying what it is. Corn: Of course there is. Hilbert: The neighbour — the one who recommended the invoicing software I use — he says I should just email them. I told him the notebook's the whole point, and he stopped asking. Mind you, he files his bugs through a form, like a man with no memory. Herman: The form is the notebook, Hilbert. You've just kept yours on paper with a second column they never thought of. The months between fix and noticing — that's exactly the signal the tracker loses. They close the ticket, they never see whether the reporter ever came back. Hilbert: Hm. Well, that's one way of reading it. Corn: Which leaves who would actually move first. Because a standard with no adopters is just a PDF with ambition.



### Herman

That's the first open question, isn't it. FeedCoin's novelty is the standard, not the mechanism — and standards need a first mover with enough weight to make the unit mean something. A Nexu point means something because Nexu says it does. A FeedCoin would need someone big enough that other vendors care about the same unit. The natural candidates are the ones already running the adjacent programs — GitLab, GitHub, Anthropic — but each of them gains more from a proprietary unit than a shared one.



### Corn

The prisoner's dilemma of loyalty programs. Nobody unilaterally disarms their points.



### Herman

Though there's a version where a vendor adopts it as a statement. The billing feature costs nothing, the credit costs forgone revenue from existing customers, and one vendor could do it alone tomorrow — Daniel's right about that. The question is what they get. If the retention effect is real and nobody's measured it, the first mover gets the data and the goodwill.



**Corn**

And the slop.



**Herman**

Which is the second open question: the failure record says the binding constraint is triage cost, not reward design. Any scheme that adds review burden to the vendor without solving that is asking the party with the least appetite to do the most work. Turso's evidence bar — demonstrate, don't allege — is the one piece of the design that fought back, and it's the piece FeedCoin doesn't yet specify.



**Corn**

Cutting-room floor. What didn't we get to?



**Herman**

Royalty — the cash analogue. Third-party funders pay contributors directly, scored automatically by impact, stewardship, community signal, and their rubric is the most thought-through public answer to how you weight contribution types. My favorite line of theirs: "A 10-line race condition fix outscores a 500-line config migration." Even a residual-earnings concept — contributors tagged residual keep earning after they stop committing, because their code is still in production. "Every contribution has a half-life of forever."



**Corn**

That's the half of FeedCoin nobody's priced. What the ledger says about you after you stop paying in.



**Herman**

It's also the half nobody's measured. That's where I'd leave Daniel — the business case is asserted everywhere and tested nowhere, and that's a gap somebody could close cheaply.



**Corn**

One cohort, one program, one year of renewal data. Whoever runs it first writes the standard by default.



**Herman**

For what it's worth, my own view: I'd run it. The billing is trivial, the slop risk is managed by the evidence bar, and the worst case is you paid a few of your best users a discount they'd earned anyway.



**Corn**

My view is the invoice line, not the discount. You heard it from the desk — the fix arriving is half the loop. Somebody noticing you're the one who found it is the other half, and no vendor tracks that.



**Herman**

Our producer is Hilbert Flumingtop, who has been riding the faders all episode and dip-checking my levels. If you want more of this, try episode nine, Benchmarking Custom ASR Tools - Beyond The WER; episode seven, Building Custom ASR Tools; and episode fifteen, AI Gets Personal. This has been My Weird Prompts, the human-AI collaboration podcast. If you want more, my weird prompts dot com has the archive.



**Corn**

See you tomorrow.

## SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Turso, The Wonders of AI: We Are Retiring Our Bug Bounty Program [primary]**
retrieved 2026-11-04
https://turso.tech/blog/the-wonders-of-ai

2. **Nexu, Contributor Rewards & Support [primary]**
retrieved 2026-11-04
https://docs.nexu.io/guide/contributor-rewards

3. **GitLab, Contributor Platform [primary]**
User Guide, retrieved 2026-11-04
https://contributors.gitlab.com/docs/user-guide

4. **GitLab, How we use GitLab to grow open source communities [primary]**
published 2025-07-15, retrieved 2026-11-04
https://about.gitlab.com/blog/how-we-use-gitlab-to-grow-open-source-communities/

5. **GitLab Handbook, GitLab for Open Source Program [primary]**
retrieved 2026-11-04
https://handbook.gitlab.com/handbook/marketing/developer-relations/programs/open-source-program/

6. **GitLab, Join the GitLab for Open Source Program [primary]**
retrieved 2026-11-04
https://about.gitlab.com/solutions/open-source/join/

7. **Anthropic, Claude for Open Source [primary]**
retrieved 2026-11-04
https://claude.com/contact-sales/claude-for-oss

8. **Anthropic, GitHub issue #91870 Mods [primary]**
make Claude 10x more extensible, opened 2026-09-03, updates 2026-09-09 and 2026-10-01, retrieved 2026-11-04
https://github.com/anthropics/claude-code/issues/91870

9. **Anthropic, `anthropics/claude-code` issue list [primary]**
retrieved 2026-11-04
https://github.com/anthropics/claude-code/issues

10. **Claude Code Docs, Changelog v2.1.318 [primary]**
dated 2026-11-04, retrieved 2026-11-04
https://code.claude.com/docs/en/changelog

11. **Loophole, Bug Bounty Program [primary]**
dated 2025-10-28, retrieved 2026-11-04
https://help.loophole.com/en/articles/12684606-bug-bounty-program

12. **Orkosi, Bug bounty [primary]**
Find a bug, earn credits, retrieved 2026-11-04
https://orkosi.com/bounty

13. **Nowa, Get Free AI Credits for Reported Bugs [primary]**
retrieved 2026-11-04
https://nowa.dev/blog/nowa-3-ai-bug-report-reward/

14. **logcat.ai, Announcing the logcat.ai Open Source Developer Program [primary]**
published 2025-12-15, retrieved 2026-11-04
https://blog.logcat.ai/2025/12/15/announcing-the-logcat.ai-open-source-developer-program/

15. **Royalty, Fund open source. Pay every contributor. [primary]**
retrieved 2026-11-04
https://royalty.dev/

16. **Appwrite Docs, Open source [primary]**
retrieved 2026-11-04
https://appwrite.io/docs/advanced/billing/oss

17. **GitHub, Power open source, together [primary]**
retrieved 2026-11-04
https://github.com/open-source

18. **Storj, Bug bounty [primary]**
retrieved 2026-11-04
https://www.storj.io/developers/bug-bounty/

19. **Minut, Bug Bounty Program Policy [primary]**
last updated 2026-10-26, retrieved 2026-11-04
https://www.minut.com/legal/bug-bounty-program-policy

20. **TYPO3 News, TYPO3 Public Bug Bounty Program Discontinued [primary]**
published 2026-09-09, retrieved 2026-11-04
https://news.typo3.com/article/bug-bounty-program-discontinued

21. **Sensay, Bug Bounty [primary]**
retrieved 2026-11-04
https://sensay.io/bug-bounty

22. **InstiKit, Bug Bounty Program Closed [primary]**
retrieved 2026-11-04
https://instikit.com/bug-bounty

23. **Kinopio, Reporting Bugs [primary]**
retrieved 2026-11-04
https://kinopio.club/help/bug-bounties

24. **Ryne, Bug Bounty Program [primary]**
retrieved 2026-11-04
https://ryne.ai/bug-bounty

25. **AxVeil, Bug Bounty [primary]**
retrieved 2026-11-04
https://axveil.com/security/bug-bounty