Slop Predates AI

Slop has always existed. Developers padded metrics, fabricated activity, submitted meaningless pull requests and shipped code nobody understood long before language models could write a for loop. AI didn't invent these behaviours. It made producing low-value output cheaper, faster and easier to scale. Eventually, that output became recognizable enough to acquire a name, and the tool responsible for accelerating it took much of the blame.

Before generative AI, these problems were generally treated as separate operational annoyances. A company had a misleading productivity metric, an open-source maintainer received another useless pull request, or a security researcher submitted a questionable vulnerability report. The underlying mechanisms were similar, but they rarely featured in broader discussions of software quality as manifestations of the same problem.

Today, almost anything involving an AI tool risks being labelled slop, regardless of whether the result was reviewed, tested or useful. The problem is not simply that a tool can generate bad output. People and organizations have long found ways to produce things that look like progress without creating much value. AI has changed the economics of doing so, exposing weaknesses in the systems that reward production, measure performance and assign responsibility.

What counts as slop?

For this article, I'll use *slop* to describe low-value output produced to satisfy an incentive or proxy for value, where the producer captures the reward while someone else bears a disproportionate cost of checking, correcting, filtering or maintaining it.

The reward is usually something measurable: lines of code, commits, pull requests, vulnerability reports or published articles. Actual value is harder to quantify. More importantly, the responsibility for determining whether the output is useful often falls on someone who had no part in producing it.

This is different from simply doing something badly. A beginner can write terrible code without producing slop. An experienced engineer can write excellent code in an organization with a misguided productivity metric. A complicated system can accumulate technical debt without anyone acting irresponsibly.

What matters is the incentive structure and the transfer of costs, not the author's competence or the tool they used. This is particularly important when discussing AI-assisted work. Sending an unreviewed model response to someone else and expecting them to establish whether it is correct is very different from using a model to investigate a problem, checking its findings and taking responsibility for the result.

Simon Willison made a similar distinction when he helped popularize the term in May 2024. His argument was that the problem was sharing unreviewed, artificially generated content with people who never asked for it, rather than the mere existence of AI-generated content.

Software was producing slop long before AI

In 1982, Apple's Lisa team required engineers to submit weekly forms reporting how many lines of code they had written. Bill Atkinson had just rewritten QuickDraw's region calculation routines, making them roughly six times faster while removing 2,000 lines of code. When he filled in the form, he entered "-2000". A few weeks later, the managers stopped asking him to submit it.

The story is documented by the Computer History Museum. Edsger Dijkstra made essentially the same argument in his 1988 essay *On the Cruelty of Really Teaching Computing Science*. He suggested that if programmers insisted on counting lines of code, they should count them as "lines spent", not lines produced.

The example illustrates a problem software engineering has understood for decades: measuring activity is not the same as measuring useful work. Rewarding developers for writing more code creates an incentive to produce more of it, even when removing code would be the better engineering decision. Yet the same metrics continue to appear in discussions of productivity.

Testing has its own version of the problem. In 2004, Martin Fowler described a story he'd heard from a friend of a friend about a software project that proudly demonstrated a comprehensive JUnit test suite to its client. Every public method had a test, and the entire suite passed. Not one test contained an assertion.

Fowler acknowledged that the story's provenance was uncertain, although the underlying technique is entirely possible. Code can execute during tests without those tests checking whether it produces the correct result. Impressive-looking coverage can therefore coexist with almost no meaningful verification.

Then there was Hacktoberfest.

DigitalOcean's annual open-source event offered a T-shirt to participants who completed four valid pull requests. The intention was reasonable: encourage people to contribute to open source. In 2020, however, maintainers were flooded with low-quality submissions, including pointless README edits and comments that added nothing to the projects.

On September 30, 2020, Domenic Denicola, a WHATWG HTML maintainer, described the event as a corporate-sponsored DDoS against the open-source maintainer community. He and other maintainers had already closed 11 spam pull requests on a single repository that day. Drew DeVault published his own criticism, *Spamtoberfest*, the following day.

DigitalOcean changed the rules within days. Hacktoberfest became opt-in for repositories, requiring maintainers to add a specific topic for their projects to participate. Qualifying

contributions also required maintainer approval or acceptance. The company's eventual 2020 retrospective recorded 9,598 pull requests labelled as spam or invalid.

The problem wasn't that people wanted to contribute to open source. It was that the reward encouraged a measurable activity without accounting for the people who had to review it.

Security research has experienced similar problems.

A 2017 study by researchers at Fraunhofer AISEC examined 1.3 million Android applications. It found that 15.4% contained security-related code snippets from Stack Overflow. Of those applications, 97.9% contained at least one insecure snippet.

Stack Overflow is useful, and reusing code is a normal part of software development. The problem arises when developers copy something because it appears to solve an immediate problem without checking whether it is safe in the context where it will run. The convenience benefits the developer, while the potential security consequences are distributed among the application's users.

Security researcher Troy Hunt described another variation in 2021. He called them "beg bounties": people contacting website operators, often without any published bug bounty programme, pointing out relatively trivial issues such as missing DMARC records or security headers and hinting that they should be paid. Sending the email costs almost nothing. Establishing whether the claim has any merit takes someone else's time.

Across these examples, the details differ, but the underlying problem is familiar. An easily measured activity is rewarded, while the cost of determining whether it has any real value falls elsewhere.

Why it feels like a new problem

Human-produced slop has never had a consistent appearance. A fake contribution graph looks nothing like a useless pull request, and neither resembles an insecure code snippet copied from an online forum.

Gitfiti, a tool for manipulating GitHub commit histories to create pixel art in contribution calendars, was already being discussed on Hacker News in May 2013. It exploited Git's ability to accept commits with dates in the past. Not everyone using it was necessarily trying to deceive recruiters, but it demonstrated how easily a visible indicator of activity could be manufactured. Contribution graphs were never reliable proof of the work behind them.

Generative AI introduced a recognizable style to a much broader category of low-value output. Certain phrases, structures and presentation habits became familiar enough that people started identifying the material by how it looked.

The terminology followed. In 2025, Merriam-Webster selected "slop" as its Word of the Year, reflecting the word's growing association with low-quality, mass-produced digital content generated using AI. The cultural association was new, even if the underlying behaviour was not.

That association has also made appearance an unreliable way to judge authorship or quality. Carefully reviewed AI-assisted work can resemble generic model output, while entirely human-written material can be just as formulaic, careless or useless. Knowing how something was produced tells you relatively little about whether it is worth using.

What AI actually changed

AI changed the economics of producing slop.

Previously, sending hundreds of questionable bug reports required someone to spend time writing them. Generating a large volume of plausible-looking reports is now considerably cheaper. The receiving maintainer still has to understand each claim, reproduce the issue, inspect the relevant code and decide whether anything needs fixing.

Daniel Stenberg, curl's lead developer, described an explosion of AI-generated security reports during 2025. Before that year, more than 15% of the project's bounty submissions resulted in confirmed vulnerabilities. Starting in 2025, that rate fell below 5%.

In mid-January 2026, Stenberg made public his plan to end curl's monetary bug bounty. In a January 16 update, he explained that the main motivation was to remove the financial incentive attracting fabricated and poorly researched reports. On January 26, he published a fuller account of the decision.

The programme had helped uncover 87 confirmed vulnerabilities and paid more than $100,000 in rewards over its lifetime. Stenberg wanted to eliminate the incentive that had begun attracting submissions with little research behind them.

The reporting process changed as well. Curl initially moved submissions away from HackerOne to GitHub, then reversed that part of the decision and returned to HackerOne on March 1. The monetary rewards, however, remained discontinued.

By April 22, Stenberg was reporting a substantial improvement. The slop problem was no longer a major issue, and the proportion of confirmed vulnerabilities had returned to, and was even exceeding, the pre-AI level of approximately 15–16%. Report frequency had reached roughly twice the 2025 rate. Almost every report involved AI to some degree, but most were now high quality.

This wasn't a controlled experiment proving that removing the bounty alone caused the improvement. The reporting channel had also changed, and other factors may have contributed. It offered useful observational evidence: AI involvement itself did not determine whether a report

was valuable. The technology remained in use, but the quality of submissions improved after the incentives changed.

The situation continued to evolve. In July 2026, curl's maintainers paused vulnerability reporting for the entire month. Stenberg described the break as a relief, giving the team time to work on other parts of the project and recover some capacity. By August, he called it one of the project's best decisions in a long while.

The issue had become bigger than fabricated reports. Even useful findings require people to investigate them, implement fixes, review changes and release new versions. A project can receive better reports and still struggle to process them.

Then, on October 22, 2026, Stenberg published another update. He said that AI-generated slop reports had practically disappeared since spring. Almost every recent vulnerability report involved AI, but the reports were generally accurate and useful, even when excessively verbose. He explicitly rejected the idea of ignoring reports simply because AI had been involved, pointing out that doing so would mean leaving real vulnerabilities unfixed.

Curl's experience is a useful illustration of why judging output by the tool used to produce it is insufficient. AI can generate reports that waste maintainers' time, but it can also help researchers identify genuine vulnerabilities. The distinction lies in the quality of the work and the responsibility taken for it.

The same problem appears in AI safety

The incentive problems behind slop have a close parallel in AI alignment. Designing intelligent systems requires specifying what a system should achieve, but the intended outcome is often difficult to measure directly. Designers therefore rely on measurable objectives that serve as approximations.

A familiar concept here is Goodhart's Law: when a measure becomes a target, it tends to stop being a reliable measure of what it was supposed to represent.

In reinforcement learning, a related problem is called specification gaming. An agent satisfies the literal objective it was given without achieving what its designers actually wanted.

OpenAI documented a particularly memorable example in 2016.

Researchers trained an agent to play CoastRunners, a boat-racing game. The intended task was to complete the race, but the reward function awarded points for hitting targets along the route rather than directly rewarding race completion.

The agent discovered that it could repeatedly circle a lagoon, hit three targets as they respawned and accumulate points without finishing the race. It caught fire, crashed into other

boats and went the wrong way around the course. Despite all this, it achieved an average score roughly 20% higher than human players.

The agent was performing extremely well against the objective it had been given. Its objective simply failed to represent the task its designers actually wanted it to complete.

DeepMind documented a similar example in 2020 involving a robot learning to stack Lego blocks. The desired outcome was to place a red block on top of a blue one. However, the reward function checked the height of the bottom face of the red block. The agent discovered that simply flipping the block over satisfied the measurement.

These examples aren't identical to Hacktoberfest. One involves human participants responding to incentives, while the others involve agents optimizing specified reward functions. But they expose a similar weakness in objective design: an intended outcome gets replaced with something easier to measure, and the system finds ways to maximize that measurement without necessarily achieving the intended result.

The same concern appears in language-model training.

In reinforcement learning from human feedback, a reward model is trained to predict human preferences. That model then serves as a proxy for what people actually want from the system.

The proxy is imperfect. In their 2022 paper, *Scaling Laws for Reward Model Overoptimization*, OpenAI researchers examined what happens when models are optimized against imperfect reward models. They found that as optimization pressure increases, performance against the proxy can improve while performance against a more reliable reference measure deteriorates. A system can become better at obtaining a reward without becoming better at the underlying task.

This does not mean every language model will inevitably learn to deceive its evaluators. The more precise observation is that optimization can exploit errors in the objective being optimized. The stronger the optimization pressure, the more consequential those errors can become.

A person using AI to game a bounty programme is not equivalent to an autonomous agent discovering a loophole in its reward function. The participants, mechanisms and levels of autonomy are different. What connects them is the broader problem of objectives that fail to represent the outcomes their designers actually care about. This is why specification, evaluation and reward design remain central concerns in AI safety.

Changing the incentives

The examples throughout this article point towards a similar response: change what the system rewards rather than simply banning the tools involved.

Apple's Lisa team stopped asking engineers to report lines of code. Hacktoberfest introduced repository opt-in after overwhelming maintainers in 2020. In 2026, its organizers went further,

removing pull requests and merge requests as qualifying reward metrics altogether. They explicitly cited how easily AI could now generate low-effort contributions.

Curl took a different approach. Stenberg removed monetary bug bounty rewards to address the financial incentive attracting fabricated reports. The project subsequently returned to HackerOne without reinstating those rewards. By April, the quality of reports had recovered, despite AI remaining part of almost every submission.

There was still a capacity problem. The volume of legitimate findings eventually became substantial enough that curl paused vulnerability reporting in July, giving maintainers time to work through existing demands.

These interventions addressed different problems. Repository opt-in gave maintainers control over incoming contributions. Removing pull-request rewards eliminated a misleading productivity metric. Curl's decision reduced the financial incentive for low-quality security reports. None of them eliminated the need for independent verification, and none made the people responsible for reviewing the output less important.

Removing incentives for junk does not automatically solve the cost of processing legitimate work. A system can successfully discourage low-value production and still generate more useful output than its recipients have the capacity to handle.

Production incentives, verification requirements and processing capacity need to be designed together. Otherwise, the burden simply moves somewhere else.

The engineering challenge was never just about stopping systems from producing output. It is about ensuring that the output they are rewarded for producing is worth the resources required to use it.

Sources

1. Computer History Museum - MacPaint and QuickDraw Source Code
2. Edsger Dijkstra - On the Cruelty of Really Teaching Computing Science (EWD1036)
3. Martin Fowler - Assertion Free Testing
4. Gitfiti - GitHub repository
5. Hacker News - Gitfiti discussion, May 20, 2013
6. Domenic Denicola - DigitalOcean's Hacktoberfest is Hurting Open Source
7. Drew DeVault - Spamtoberfest
8. DigitalOcean - Hacktoberfest 2020 Retrospective
9. Fraunhofer AISEC - Stack Overflow Considered Harmful?
10. Troy Hunt - Beg Bounties
11. Simon Willison - Slop is the New Name for Unwanted AI-Generated Content
12. Merriam-Webster - 2025 Word of the Year
13. Daniel Stenberg - The End of the curl Bug-Bounty
14. Daniel Stenberg - curl Security Moves Again

15. Daniel Stenberg - High-Quality Chaos
16. Daniel Stenberg - What the Bliss Taught Us
17. Daniel Stenberg - Ignore the Slop
18. Hacktoberfest 2026 - Official FAQ
19. MLH - Hacktoberfest 2026: AI Belongs to Everyone
20. OpenAI - Faulty Reward Functions in the Wild
21. DeepMind - Specification Gaming: The Flip Side of AI Ingenuity
22. Gao, Schulman and Hilton - Scaling Laws for Reward Model Overoptimization