



Secure your code

The Essential Guide to Managing Security Debt

The rise of security debt

Security debt is stalling DevSecOps, as developers lose trust in security tooling and alerts. Security has always been a delicate subject with development teams. Gone are the days of “move fast and break things” because security is no longer a “nice to have.” Today it’s a non-negotiable for all applications as security breaches can lead to financial losses, reputational damage, and compliance penalties.

But it’s a balancing act. Companies can’t stop innovating or slow the pace of development. The cold reality is that development teams, under pressure to meet business demands, continue to introduce more security vulnerabilities to code than they fix. This results in growing security debt, which is commonly defined as the accumulation of vulnerabilities that remain in your organization’s codebase for more than a year without remediation. Security debt is a significant organizational risk because software vulnerabilities are a [leading source of costly security breaches](#).¹

1: Verizon Business “[2024 Data Breach Investigations Report](#)”

Not addressing security debt can have negative long-term consequences. If it’s exploited, it can ultimately lead to a data breach, financial losses, reputation damage, and regulatory fines. That’s why it’s important to have regular audits, to gather information about the size and scope of your security debt, and to ensure you’re identifying the most high-risk vulnerabilities that may be present.

Fortunately, there is a way out of security debt—and that’s by leveraging developer-centric AppSec tooling and the latest advancements in AI. It’s possible to develop at a fast pace, without sacrificing security or pitting security and development teams against one another. In this guide, we’ll unpack what security debt is, why it’s important to pay down, and how you can transform your security operations.

The cycle of security debt

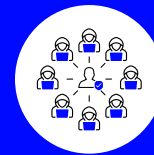
Let's start by looking at the vicious cycle that causes organizations to rack up security debt in the first place.

- Security teams already fight an uphill battle because the number of [security professionals are vastly outnumbered by software developers](#).²
- Developers are measured on output and often lack security training and the guidance to find and fix vulnerabilities code.
- When guidance is given, it's often littered with false positives; E.g., alerts that aren't really security risks.
- As a result, developers often lose trust in security tooling and guidance and accept a subjective level of risk. This leads to shipping insecure code to meet deadlines, which is ultimately what they're measured on. [One survey](#)³ found that 56% of developers struggle to prioritize remediation.
- The end result is high-risk vulnerabilities in production code that are even more time consuming and costly to fix—or even worse, exploited, resulting in a cyber event that can cause long-term financial and reputational damage.

2: Inforsecuity Magazine "[Developers Outnumber Security Pros 100:1 as Breaches Grow](#)", Apr 2018

3: Checkmarx "[Future of Application Security 2024](#)", Annula Report

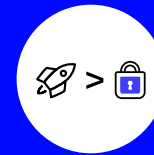
What causes security debt?



Security teams are outnumbered



Developers don't get enough training to be security experts



Developers are incentivized on shipping software more than security



Developers suffer from alert fatigue and lose trust in security tooling



Unclear prioritization of which vulnerabilities to remediate

Why traditional application security tooling falls short

Organizations spend billions of dollars per year on cybersecurity tooling and services. In fact, estimates show that organizations have [more than 70 tools on average](#).⁴ Yet, we still see security and developer teams struggling to keep up with rising security debt.

The major reason is that traditional security tools often fail to meet developer needs. These tools aren't typically designed with the developers in mind, even though developers tasked with remediation are the core users. These tools require developers to leave their familiar environments and workflows to run manual tests, which require context switching. Furthermore, since most developers aren't security experts, security issues require researching vulnerabilities and how to fix them—and that often comes at a great cost to productivity.

Even more worrisome is that when these tools surface issues, the alerts they

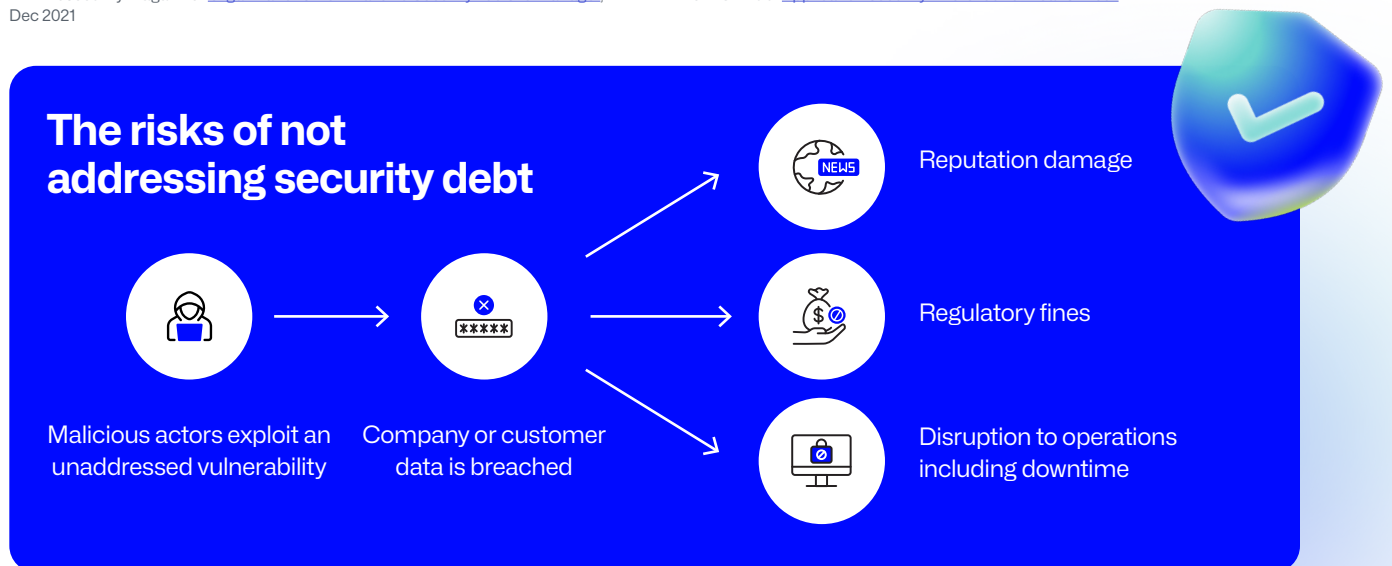
generate often provide little context and aren't actionable. For example, many alerts are false positives. Others are low risk—meaning if a malicious actor exploits them, they won't be able to cause much harm. Overwhelmed by alerts that are questionable or lack context, developers often end up ignoring them.

What DevSecOps teams don't need is more alerts, detection capabilities, or additional security tooling. Instead, they need help prioritizing and scaling remediation efforts to pay down security debt.

At GitHub, the world's largest developer community and the home of open source, we aim to deliver native security tooling in a true developer-first approach. There's no need for clunky integrations or plug-ins, because [GitHub Advanced Security](#)⁵ is embedded within GitHub, which is already a core part of the developer workflow.

4: Infosecurity Magazine "[Organizations Now Have 76 Security Tools to Manage](#)", Dec 2021

5: GitHub "[Application security where found means fixed](#)"



Escape the security debt cycle with DevSecOps

Just as with financial debt, the first step to getting out of security debt is realizing and measuring the scope of the problem. To take action, you must have the capability to address your application security (AppSec) debt in large swaths. By leveraging modern tools and AI, you can navigate and prioritize this debt in chunks from high to low priority, just like you would with a credit card by looking at the highest interest rate ones first. Even more importantly, approaching your AppSec vulnerabilities in a 'debt reduction' mentality will help develop a sense of teamwork and camaraderie between AppSec and development teams, which traditionally have misaligned goals and priorities.

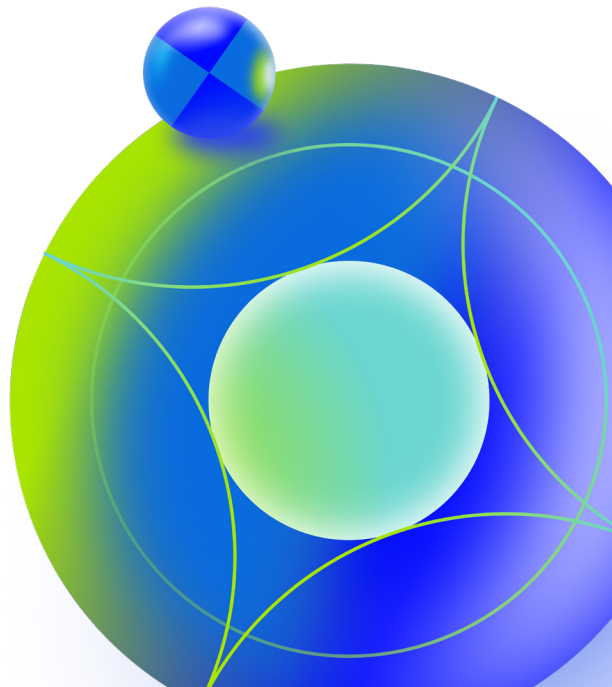
DevOps has transformed how many organizations build and ship software, but one aspect of the software development lifecycle (SDLC) was left outside the DevOps model: security. [DevSecOps⁶](#) seeks to correct that by incorporating security into the SDLC by weaving automated security testing into every aspect of the DevOps culture, tooling, and processes in the same way that quality, speed, and collaboration are already integrated. Put simply, DevSecOps is the natural evolution of DevOps, with security baked into the process.

6: GitHub "[What is DevSecOps?](#)"

DevSecOps is critical to getting application security right. [A Harris poll⁷](#) found that nearly three in four developers responded that software supply chain security tools were debilitating to their productivity, while fewer than half of CISOs responded that they believed that developers were "very familiar" with the security risks associated with their tools and workflows. This disconnect between teams can create friction and helps no one. Developers need security tools embedded into their workflows to enable them to fix vulnerabilities without hindering creativity and productivity.

At GitHub, we saw this firsthand, and that's why we began offering native security capabilities for our platform for free on all public repositories, and offer licensed features for private repositories.

7: Chainguard "[CISO & Developer Trends in Software Supply Chain Security](#)", Nov 2023

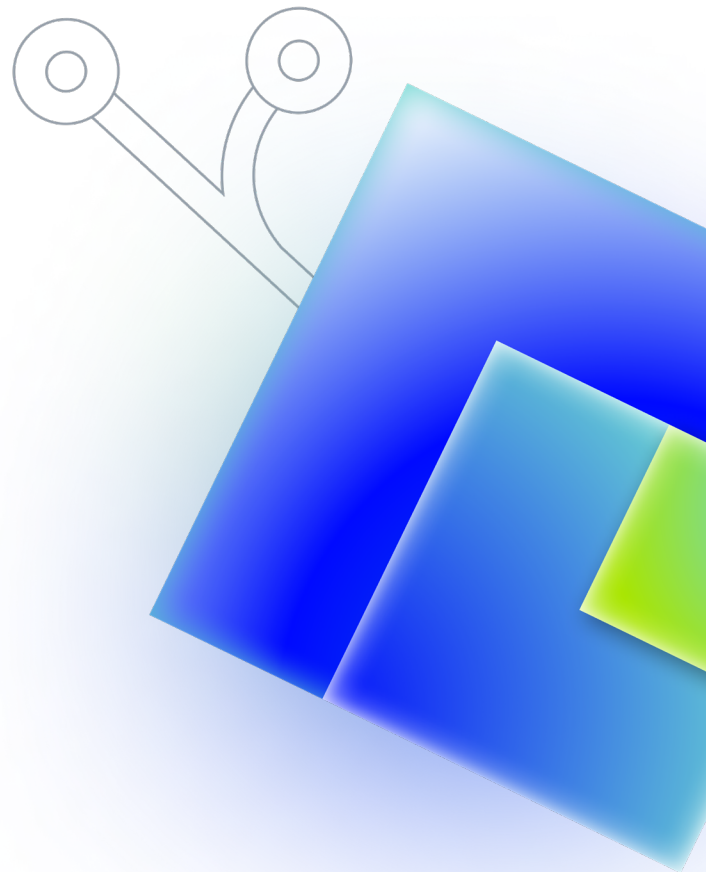


Found means fixed, a new approach to AppSec

Thanks to advances in AI, organizations now can scale and tackle their security debt. We no longer have to assume developers are security experts. Instead, we can leverage AI almost in real time to help fix security issues so developers can get back to doing what they do best—coding. At GitHub, we aim to augment, accelerate, and empower teams with the power of AI—and specifically with the power of [GitHub Copilot⁸](#), the world’s most widely adopted AI developer tool. We started by successfully automating mundane tasks for developers. Now, we’re bringing that to life in application security.

The basis of our vision for security is: “Found means fixed.” This means that developers can fix issues as they find them with the help of AI, almost instantly and with just a few clicks. More specifically, Copilot Autofix goes beyond notifications with automatically-generated vulnerability analysis, which includes detailed alert explanations, and, most importantly, suggested fixes that help developers remediate vulnerabilities as fast as they are found. We support C#, Java, JavaScript, Python, and [more⁹](#).

Experienced security talent is in short supply, but with Code Scanning Autofix, every developer benefits from the security expertise of Copilot whenever they need it. This frees up security teams to focus on strategies to protect the business instead of triaging the deluge of everyday vulnerabilities. Copilot allows developers to automate fixes with oversight, accelerating remediation, but humans stay in the pilot’s chair, reviewing proposed fixes.



8,9: GitHub [“Responsible use of Copilot Autofix for code scanning”](#)

How GitHub Copilot Autofix helps teams dig their way out of security debt

With Copilot Autofix, teams can pay down years' worth of security debt—even those low- and moderate-severity alerts that can be hard to prioritize—with speed and efficiency through targeted campaigns.

To understand how transformative this process is for security and development teams, let's take a look at how the tool works:

- In the pull request: When a new vulnerability is found, Copilot Autofix generates an explanation and code suggestion to help fix the problem
- For existing alerts: Open an alert and press the "Generate fix" button







Copilot Autofix provides explanations and code suggestions for around 90% of alert types so development and security teams can rapidly address the majority of alerts that comprise their security debt.

We've found that developers using Copilot Autofix are able to address code vulnerabilities more than [three times faster](#)¹⁰ than those who do so manually. It's a powerful example of how AI-powered security solutions – especially those built into the developer workflow – can radically simplify and accelerate secure software development.

10: GitHub, "Found means fixed: Secure code more than three times faster with Copilot Autofix", Aug 2023

- Customers **fixed code vulnerabilities more than three times faster** than those who did so manually, reducing time to fix for a pull request-time alert from 1.5 hours to 28 minutes.
- **They were able to fix cross-site scripting vulnerabilities seven times faster**, reducing time to fix to 22 minutes, compared to almost three hours.
- **They fixed SQL injection vulnerabilities twelve times faster**, cutting time to fix to just 18 minutes, compared to 3.7 hours.

Reducing remediation time with Copilot Autofix

	Manual Fix	Copilot Autofix
Fix within pull request	1.5 hours 	28 mins 
Fix cross-site scripting (XSS) vulnerability	3 hours 	28 mins 
Fix SQL injection vulnerability	3.7 hours 	18 mins 

A 3-step action plan for reducing security debt

Step 1 – Stop adding to your debt:

The first step to getting out of credit card debt is to stop borrowing money. So it is with security debt. GitHub Advanced Security helps surface vulnerabilities at the time of the pull request, before issues are committed to a codebase. Copilot Autofix will offer suggested fixes for most vulnerabilities, so developers can fix issues as they code. Developers remain in full control and can accept, edit, or dismiss a suggested fix. They're able to address issues as they arise and while the code is still fresh in their minds, keeping new vulnerabilities from stacking up.

Step 2 – Educate and empower your developers about security debt:

As consumers work their way out of debt, they learn tactics and strategies to protect themselves moving forward. Autofix empowers developers by educating them about security and encouraging best practices. Over time, developers become more knowledgeable about security practices and can learn to write more secure code, thus improving security moving forward.

Security campaigns with Copilot Autofix

Agentic AI refers to artificial intelligence capable of making decisions, planning, and adapting to new information in real time. [AI agents](#)¹¹ learn and enhance their performance through feedback, utilizing advanced algorithms and sensory inputs to execute tasks and engage with their environments.

Copilot Autofix is an agent that aims to generate high-quality fixes for detected vulnerabilities. Even with the power of Autofix, it can take a long time to remediate large backlogs of vulnerabilities. Security campaigns gives security managers the power to “divide-and-conquer” thousands of alerts into manageable “campaigns” through robust filtering tools to triage competing priorities, plus a dashboard and reporting tools to track and demonstrate progress. For example, you can create a campaign to identify all instances of the log4shell vulnerability in log4j, or create a campaign to clean-up all high-severity vulnerabilities in an important repository. The dashboard and reporting tools keep stakeholders, from junior developers to the C-suite, up to date as the backlog shrinks.

Unlike third-party tools that require integration steps, new logins, or custom configurations, security campaigns are built into GitHub, the DevSecOps platform that developers already trust and love. By pairing the power of Copilot Autofix with the orchestration of security campaigns, organizations can bring development and security teams together to attack vulnerabilities rather than each other, systematically reducing the risk of disruptions and zero-day attacks.

The value of AI agents in software development is clear: they can offload time-consuming activities and enable developers and security professionals to focus on higher-value activities. Our focus at GitHub is on rethinking the developer “inner loop” as collaboration with AI. That means AI agents that can reliably build, test, and debug code. The more we can do that, the less energy required for engineers and the more developers who can learn from and contribute to code bases.¹¹

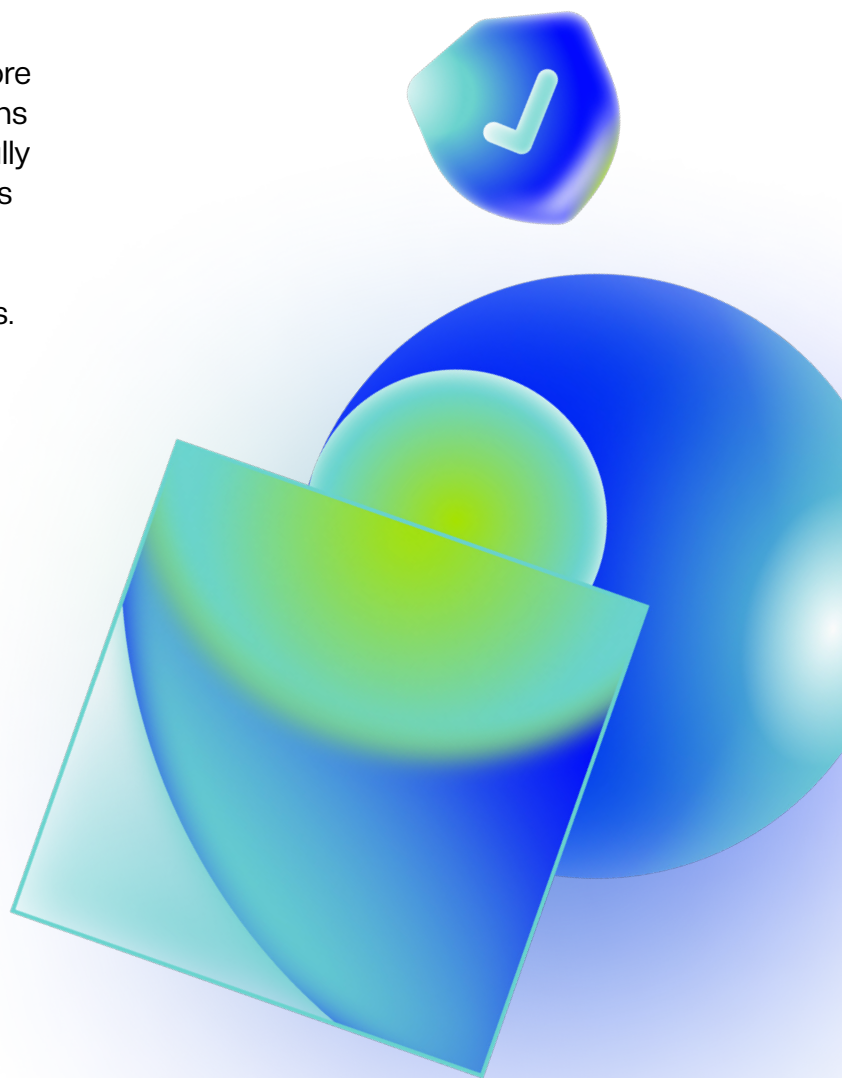
¹¹: <https://github.blog/ai-and-ml/generative-ai/what-are-ai-agents-and-why-do-they-matter/>

Step 3 – Get rid of debt in large,

targeted chunks: One strategy to pay down credit card debt is to chip away at one card at a time, often starting with the ones with the highest interest rates. You can apply a similar strategy to security campaigns, where you can target and generate contextual explanations and code suggestions for up to 1,000 historical alerts at a time. You might want to focus on securing one specific codebase or eliminating all instances of a specific vulnerability like Log4J. Tackle the most impactful vulnerabilities first and move down the list from there. This targeted approach helps developers focus on a more manageable number of vulnerabilities—tens instead of hundreds—at a time. Successfully completing a security campaign also gives teams measurable results that they can report to stakeholders, and builds trust between development and security teams.

Copilot Autofix takes care of cumbersome security tasks, ensuring our existing and new code is always as secure as possible. Vulnerabilities are flagged immediately and code changes are recommended automatically. It helps our teams to free up time so they can focus on more strategic initiatives.

Mario Landgra | Community Manager, Security | Otto (GmbH & Co KG)



Conclusion

There's no such thing as a perfectly secure codebase. But with the right tools and strategy, you can squash known vulnerabilities, keep your applications more secure moving forward, and free your developer and security teams to do higher-value work.

Getting Started with GitHub Advanced Security

GitHub Advanced Security is a developer-first application security testing solution that brings GitHub's world-class security capabilities to public and private repositories. It only takes a few clicks to get started. Right out of the box, you'll benefit from highly curated detection and remediation capabilities crafted by some of the world's best security engineers to help ensure your code and software supply chain are as secure as possible. It's fully automated, so once it's enabled, developers don't have to remember to run tests or wait for a security review before merging.

GitHub Advanced Security features include:

- **Secret scanning:** Protect your developers from accidentally leaking tokens and other secrets by testing for over 200 token types with support from a partner program of approximately 150 service providers that help detect

leaked secrets across common software development tools. With over 500 definable custom patterns, you can help ensure that your unique or proprietary secrets are also detected. Meanwhile, push protection prevents vulnerabilities from being created in the first place by actively warning developers at the commit stage.

- **Code scanning:** GitHub's application security testing interface is home to CodeQL, a semantic static analysis engine that can uncover known vulnerabilities and their unknown variations, potentially unsafe coding practices, and other code quality issues. GitHub provides thousands of queries covering the most critical types of vulnerabilities. For example, the combination of our default CodeQL and Dependabot queries—selected for their high level of accuracy to ensure low false positivity—will help ensure you stay OWASP Top 10 and SANS Top 25 compliant. At the same time, you can write your own queries to cover edge cases unique to your organization.

- **Reporting:** [Security overview](#)¹² provides a high-level view into how application security efforts are performing over time, while also providing granular filtering capabilities to identify and prioritize problematic areas of the codebase that require immediate attention.
- **Supply chain security:** As the home of open source, GitHub offers many supply chain security features for free to honor our commitment to

make open source usage secure for everyone. This includes Dependabot, which identifies vulnerabilities in dependencies and suggests automatic ways to fix, patch, or update them. To extend these capabilities for Enterprise users, GitHub Advanced Security offers supply chain security tailored for the enterprise, like dependency review, a proactive feature that helps prevent insecure dependencies from making it into private repositories.

[Learn more about GitHub Advanced Security](#), or talk to your GitHub representative. We're on standby.

