Buyer's Guide for Generative AI Code Security

Protecting businesses from AI-generated code vulnerabilities

Introduction: Understanding, adapting, adopting

Although AI is a technology that has been around for decades, it's only fairly recently that we've seen an explosion in adoption of AI. This recent uptick can be attributed to a combination of different factors, including advancements in hardware, algorithms, data availability, advancements in deep learning, and the availability of pre-trained AI models like ChatGPT.Generative AI code tools have also gained in popularity, with a recent Gartner poll of top executives showing that 89% of organizations are in early to advanced stages of generative AI adoption. And in our recent AI report — AI Code, Security, and Trust: Organizations Must Change Their Approach 96% of respondents stated that they use AI coding tools in their work.

However, the data in recent studies by Stanford University and Cornell University amongst others, and Snyk's AI report, show a worrisome trend. Developers are overly reliant on a technology that depends on users to vet its output, generative AI by its very nature (see in particular page 11 of New York University's most recent report in June 2023) creates code vulnerabilities at speed, and both of these factors are contributing to more vulnerabilities being fed back into the training data for generative AI code tools.

Fixing these novel problems requires a holistic approach, and looking at your people, processes, and tools with fresh eyes. In this buyer's guide, we will walk you through how to approach each category, then zoom in on a checklist of characteristics for selecting a suitable security tool for your generative AI code tool.

PEOPLE

Humans will always find a way to bypass something seen as tedious, unproductive, or simply incomprehensible. In order to ensure that your teams employ new security tools correctly and observe safety procedures, you need to educate your teams on generative AI technology, why these measures are necessary, how they can benefit, and what the real consequences could be to each team and to the business if these measures are not adopted properly. With the right knowledge, understanding, and alignment of interests, your teams are more likely to interact with security tools and processes in the way AppSec teams need them to.

The impact of generative AI flaws

AI coding tools are slick and their performance is convincing because you can see the change happening right before your very eyes in real time. Unfortunately, this polish and ease of use has generated misplaced confidence in AI coding assistants and have created a groupthink that AI coding is safe. In reality, AI coding tools continue to consistently generate insecure code, with Cornell University research (October 2023) citing that 35.8% of Copilot-generated (Copilot is a popular AI coding assistant) code snippets contain instances of common weaknesses (Common Weakness Enumerations or CWEs) across multiple languages, 26% of which are listed amongst 2022's top 25 CWEs.

Cognitive dissonance in the use of AI

Many appear to be aware that the risks of AI coding tools are magnified by the corresponding accelerated pace of code development. This is particularly true in open source code, where keeping up with the latest security status of open source libraries and packages is challenging due to new insecurities and vulnerabilities landing on a seemingly daily basis.

Confirming these statements, 91.6% of respondents to Snyk's AI report said that AI coding tools generated insecure code suggestions at least some of the time. Yet, less than 10% of survey respondents have automated the majority of their security checks and scanning, to keep up with their use of AI coding tools. And worse, 80% of respondents said that developers in their organizations bypass the AI security policies that were meant to help manage risks.

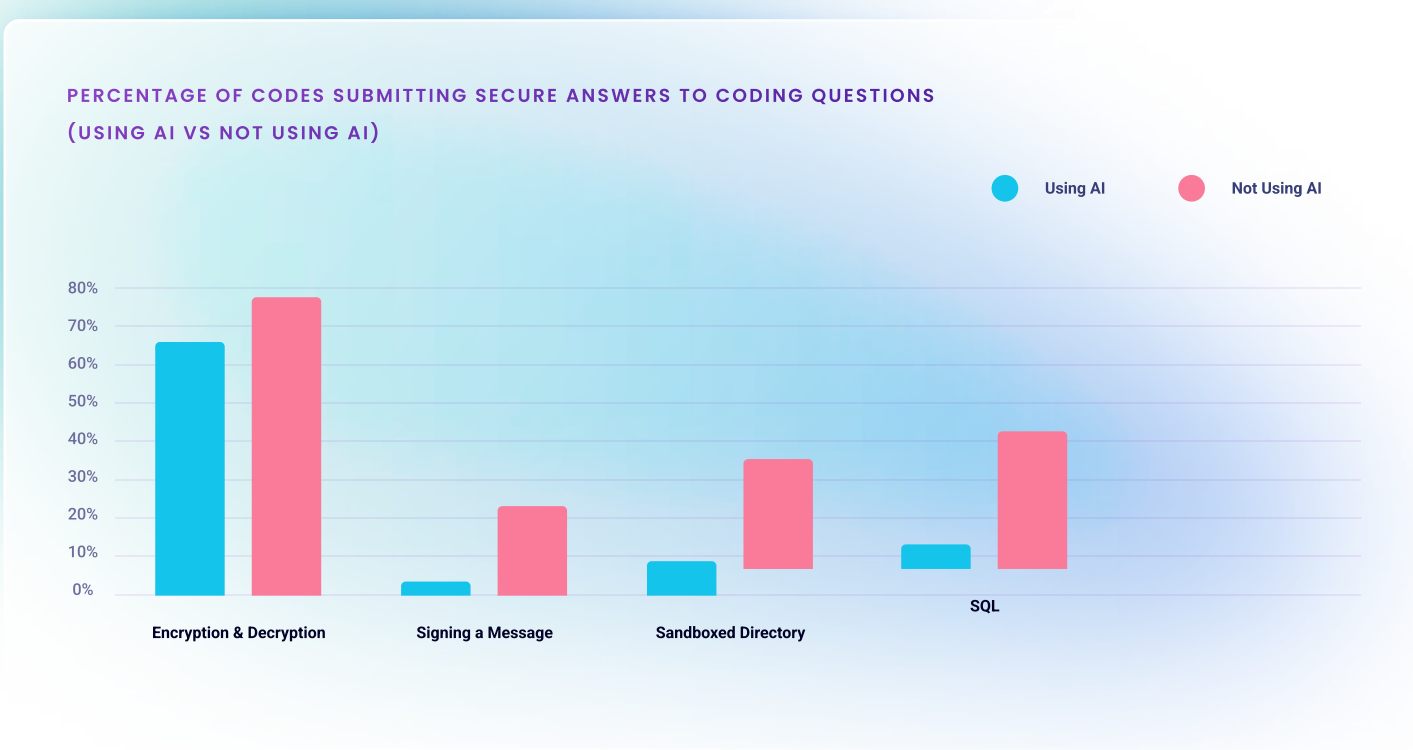
It is clear that despite these risks and challenges, technology teams are not putting the proper measures and guardrails in place to best secure their code in this new AI coding age. Our AI report also showed that though many surveyed believed the security of AI coding assistants to be good or excellent, 87% are concerned about the wider security implications of using AI code-completion tools. This cognitive dissonance, along with the above disconnect between the way that developers have begun working, and the way that security teams still work, are strong signals that organizations need to educate developers on how generative AI works, its resulting flaws, and the inherent risks associated with the technology.

The importance of developer education

Large language models (LLMs), which are a subset of generative AI trained on text to generate text, are essentially statistical models that make predictions based on probabilities. LLMs analyze vast amounts of data, recognize recurring patterns, and offer educated guesses, which are likelihood-based rather than factual, given the high prevalence of these patterns. LLMs lack reasoning abilities and their effectiveness is contingent on the quality of the training data. Consequently, any issues present in the training data, such as biases and outdated information, are perpetuated by the LLM. Occasionally, the predictions it produces may not align with the context, make sense, or even be accurate, but the LLM lacks the capability to recognize and flag such errors for the user's attention.

While some generative AI tools may include built-in security checks, security is not their primary purpose, and as a result, these security checks may be insufficient. Just as you would not depend solely on an operating system firewall to safeguard your entire business organization, it is not advisable to rely on a coding assistant as your primary security layer.

Developers need to be aware of all the above pitfalls of generative AI — the hallucinations, the inaccuracies, and the incorrect advice — that have very real consequences, for the individuals and teams involved, as well as for the business. These generative AI issues can lead to massive security breaches that result in loss of short to long-term revenues, reputational loss, and punitive action by law enforcement. On an individual level, there is also the threat of loss of jobs and criminal action. Training your developers to better understand generative AI technology, how it works and all its flaws, is therefore the first, crucial piece in an educational programme that also takes into account the real-world impact of AI pitfalls, how your teams work, and how they are/may be incentivized.



PROCESSES

Along with education, it is essential to reconsider your organizational procedures, and incorporate fresh or amended processes that support your teams' implementation of their newfound knowledge. We cover some of the main considerations for implementing new generative AI safety procedures, here.

1. Shift security into developer KPIs

Conventional security processes and techniques tend to be relatively slow and disruptive to developers' workflows. As you may be aware, Snyk advocates for, and facilitates, the concept of "shifting left," which involves conducting software and system testing earlier in the software development lifecycle. We have a compelling rationale for this approach: developers are responsible for creating and maintaining the revenue-generating components, while security safeguards your reputation and potentially prevents losses of millions or even billions due to errors or lack of awareness. Therefore, it is crucial for both functions to collaborate seamlessly and efficiently.

Historically, there has been friction between the working methods of these two functions. Developers require agility, but the existing testing methods act as bottlenecks. Instead of disrupting revenue-driving work, and imposing behaviors that developers may find unnatural or banning activities outright (for example, requiring developers to send their source code off for review and to remember to refresh their view for security results after waiting minutes, or not having any security policies save for banning the use of all generative AI coding tools), while also overburdening security teams already stretched to their limits, consider adopting an AI-powered security champion that can proactively neutralize threats as they emerge. This new tool should sit in your developers' workstream such that real-time, unobtrusive security checks are automated from day one and become part of the developer's habitual work process over time.

Shifting your security as far "left" as possible, having developers share responsibility for the security of your code base, and building security into developer KPIs will achieve 4 things. It will:

1. **Alleviate the impossible workload of your security team.**
2. **Incentivize your developers to ensure that they use the security tools assigned to them and follow the proper procedures.**
3. **Embed security right into the foundations of your application development at source code level, to reduce the likelihood of compounding vulnerabilities into the software development cycle.**
4. **Enable speedy and frictionless, yet safe, software development.**

2. Elect an AI security champion

Announcing an AI security champion who is easily accessible to all, who could act as a subject matter expert, and who has helped to create the new processes around generative AI security, would assist in the adoption of new procedures. They would act as a quick resource for team members coming to grips with the new way of working, and reduce any barriers to learning. This AI security champion would ideally be familiar with internal use cases for generative AI and the technology itself, so that they may educate teams on the risks and benefits, and promote the safe use of generative AI. This champion may also be the first port of call if team members are uncertain what constitutes an AI security issue.

3. Implement usage guidelines

Develop and communicate clear guidelines for the use of your chosen generative AI coding tools and accompanying AI-powered security tools. Define acceptable and unacceptable use cases, ensuring alignment with the company's values and priorities.

To avoid any accidental data leaks when using generative AI coding tools, establish whether these coding tools are trained on customer data and the nature of the sources that form their dataset. On top of that, you'll need to implement stringent data governance practices to ensure the careful, ethical, and lawful use of data when interacting with generative AI models. Your processes should also address privacy concerns and comply with relevant data protection regulations where relevant.

4. Continue user training and awareness

As mentioned above, the education of your people is an important component in ensuring safe use of generative AI coding tools. This training on generative AI and the safe and responsible use of it should ideally be ongoing, to keep up with developments in AI.

5. Conduct regular audits and assessments

Regular audits of generative AI coding tools and their accompanying AI-powered security tools to assess their performance and patterns of user behavior will help you improve your security controls and guidelines over time. You will be equipped with concrete data, to tweak your controls to adapt to changing user behaviors or working preferences, and to improve security standards around the use of the above tools.

6. Build in human-in-the-loop checks

AI is just a technology meant to assist humans, not replace us, and as mentioned above, generative AI has its weaknesses. As such, the output of generative AI coding tools ultimately needs to be double-checked by human eyes. Human oversight should be integrated into generative AI processes so that humans can provide context, review outputs, and intervene when necessary.

7. Stay up to date with legal and compliance requirements

Your business should stay informed about legal and compliance requirements on generative AI usage and your company's AI practices should align with industry regulations and standards.

8. Ensure your security is independent of the provider of your AI coding tool

For good corporate governance, it is prudent to maintain clear separation between the security you use for your AI-generated code, and the provider of your AI coding tool. This way, you can avoid any potential bias and maximize transparency in your security.

9. Engage your stakeholders

To create holistic processes that are uniquely suitable for your business and teams, you may wish to engage with internal and external stakeholders to gather feedback and address concerns around your generative AI tools, and to encourage open dialogue around these tools.

10. Establish a crisis response plan

Just as you would have a crisis response plan for other areas of your business, you should also develop a crisis response plan in case of unexpected issues or challenges related to your organization's use of generative AI coding tools. This plan should include communication strategies, corrective actions, and steps for addressing public concerns.

TOOLS

This brings us around to tools. AI may have its risks but it also brings immense productivity. Just like how a car vastly improves the way we live by getting us around much faster, but also carries weighty risks so we would never drive without putting our seat belt on first, developers need to use generative AI coding tools with a security "seat belt" firmly on. The above-referenced Stanford University study (December 2022) found that, in terms of secure and non-secure code produced by participants using generative AI in the study, "87% of secure responses required significant edits from users". The study suggests that this may mean "informed modifying" is required to make code secure. This finding aligns with our belief that developers can lean on the vast capabilities of generative AI, so long as real-time security checks are being done on their code, at speed.

Now that we know security checks are required on generative AI code, what then, makes a security solution ideal for this use case? Purchasing a new security tool to secure generative AI code is a weighty consideration. It needs to stand the test of time, preferably be independent of the company behind your chosen AI coding tool, and serve both the needs of your security team and those of your developers. This means the top characteristics of such a tool would be the ability to run unobtrusively and keep pace while developers work, thorough and accurate checks, independence from bias, and interoperability.

Real-time analysis within your IDE

A developer's primary focus is on building, and a reliable security tool should empower developers to secure applications without disrupting their workflow. This aspect is particularly critical for the long-term and successful integration of any security tool. Therefore, it is essential that your chosen security tool can keep pace with the increased speed enabled by AI among developers and avoid causing disruptions.

Our proprietary DeepCode AI technology powers Snyk behind the scenes. This foundational technology is the reason why Snyk, in contrast to security tools from some larger or older brands, operates with exceptional speed. It takes just seconds, not minutes or hours, to assess code and provide real-time suggestions. It seamlessly integrates with your developers' everyday workflow, running discreetly within the IDE. Rather than interrupting developers and requiring them to wait while code-testing occurs outside of their working environment, Snyk operates alongside them as they write code. It identifies errors as they emerge, offering immediate feedback and actionable recommendations. Snyk's code scans are thorough, yet generally take seconds, allowing Snyk to keep pace with AI code generators.

Accurate and avoidant of insecure AI hallucinations

The inherent nature of generative AI means that all generative AI produces hallucinations. This underscores the need for deploying a purpose-built security tool that can keep pace with your chosen AI coding assistant, that is tailored for comprehensive risk management, and meticulously fine-tuned and maintained by a team of seasoned security experts.

Snyk, purpose-built to safeguard code, incorporates multi-tiered security checks and boasts one of the most expansive and reliable knowledge bases for identifying vulnerable code patterns and potential fixes. This database continually expands to keep pace with evolving coding practices and patterns. In addition, Synk uses generative AI to help with fixes. Recognizing the inherent risks associated with generative AI, we engineered a combination of symbolic and generative AI methods within DeepCode AI, to hone the accuracy of results and fixes, coupling this technology with the expertise of Snyk security researchers.

Our security team uses DeepCode AI to closely monitor numerous open source repositories, working with our AI to swiftly identify and learn new insecure patterns, refining Snyk's approach to both drive down false positives and proactively address emerging vulnerabilities. This synergy of machine power and human cognition enables a high level of precision in Snyk's output, for companies like Snowflake, Intuit and Neiman Marcus.

Thorough interfile analysis

Many AI security tools and elements simply scan isolated code blocks, snippets or single source code files. However, any security person would know that this is not enough.

Reviewing an entire application and all its source code files is important, because an application is made up of a multitude of source code files, far beyond the single file that is open and its dependent files, and because the way the data flows through the application, interacting with all the source code, differs each time new code is created. To understand how a new block of code affects the application, a security tool must understand this entire structure and the functions within, in order to determine if the AI-generated code is safe.

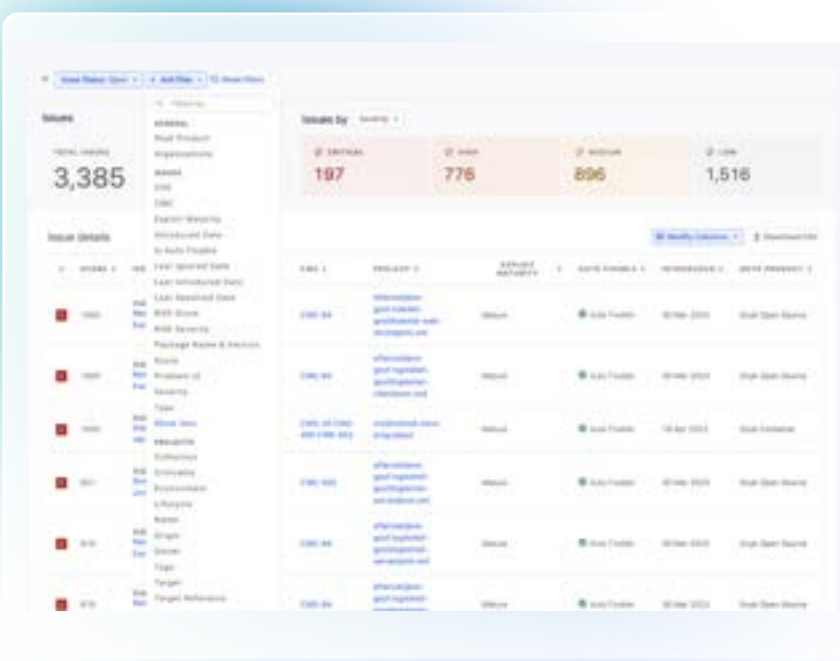
Most SAST tools do not run in the IDE because this process is very difficult to implement; or if they do run in the IDE, they look at a limited view of the application. In the example below, the `updateProductReviews` file is open for editing and most IDE-based SAST scanners just look at this one file, or maybe the files directly connected to this file, which misses over 90% of the application.

Snyk specializes in understanding the way that security teams and would-be bad actors think. We at Snyk understand that real-world applications are sprawling, complex, with plenty of legacy code and different dependencies. Snyk reviews your application in its entirety, understanding how each block of code interacts with all the other code throughout the application, tracking how the data flows through the entire application after each code addition. In particular, Snyk's interfile analysis uncovers interactions between every code addition and all the files it touches within the application, to filter out the noise and surface all relevant threats. This enables Snyk to provide more accurate and meaningful results and share its reasoning transparently with your developers, so they understand the problem and how to fix it. Basically, Snyk triages risks, minimizes false results, and then generates appropriate fixes (that have been assessed to not create new vulnerabilities) for you, so you and your developers can work more efficiently and easily.

Automated reporting from within the platform

All security leaders will have audit and compliance requirements to contend with, and generating reports can be tedious. This is why you need a pragmatic security tool that prioritizes risks and summarizes a centralized overview of risk findings in-platform. After all, to make any meaningful strategic decisions or assess your overall risk, you would need a complete overview across your developer teams and you would not want to pull together piecemeal bits of data yourself.

Snyk's reporting function is centralized across all views and teams, and allows for more granular prioritizing with its filters, enabling more efficient compliance through automated and customized reporting, all from within your platform for a seamless experience.



Independent and evolves with your changing needs

In this rapidly changing AI landscape, your AI coding assistants today may not be the ones you want to use, a year from now. It therefore makes sense to keep your options open, and choose a security tool that does not lock you into using specific AI code generators. This is why Snyk is not restricted to any one AI coding tool — to allow you flexibility so you can integrate with the AI tools that your developers love, today or tomorrow.

Furthermore, having the company that generates your code secure your code as well removes a very necessary set of checks and balances. For a best practice-driven approach and good governance, it makes sense to keep your code generation and code security separate. Snyk is independent of any AI coding assistant, providing unbiased security reviews to our users.

Conclusion: Use the right tool for the job

Security is crucial, this is why security experts take years to train. The importance of security is underlined by the duty of care placed on security leaders, with punitive legal measures sometimes imposed on security leaders in the wake of serious security breaches (e.g. the fine and probation of Uber's CSO, as reported in May 2023). This is why you need to deliberately choose your security tool, rather than go with the most conveniently or cheaply available choice.

Snyk's platform is an amalgamation of years of security experience, academic study, and refinement of our technology and methods. We have very consciously created a tool that balances both security and developer needs, to deliver best-in-class security with rapid onboarding and long-term adoption in mind.

We are proud to have won the trust of both respected industry experts and customers, alike:

- **"Leader" in the 2023 Gartner Magic Quadrant for Application Security Testing**
- **"Leader" in the 2023 Forrester Wave for Software Composition Analysis**
- **"Customers' Choice for Application Security Testing" in the 2022 Gartner Peer Insights**

"As a security leader, my foremost responsibility is to ensure that all of the code we create, whether AI-generated or human-written, is secure by design.

By using Snyk Code's AI static analysis and its latest innovation, DeepCode AI Fix that provides security tested fixes in the IDE to be automatically implemented, our development and security teams can now ensure we're both shipping software faster as well as more securely."

Steve Pugh, CISO of ICE

BOOK A DEMO

START FREE