AI in Application Security

Finding Value in the Hype

qwiet AI

TABLE OF CONTENTS

Overview

There’s no doubt that generative AI is having its time in the sun. Popularized by tools like ChatGPT and Midjourney, AI has made its way up most organizations' priority lists. Gartner predicts that by 2026, more than 80% of enterprises will have used generative AI APIs or models and/or deployed generative AI-enabled applications in production environments. This is up from less than 5% in 2023, so we’re really at the beginning of the expansion of professional-grade generative AI.

Of course, with all of the buzz around AI, it’s hard to tell the actual use cases from the hype. Is there a legitimate use for AI in application security? The quick answer is “yes,” but it’s a much more nuanced question than one might think. In this whitepaper, we’ll look at the ins and outs of generative AI: the types of models, use cases for AI, and how to best use AI to help organizations to ultimately produce more secure code.

All Models are Not Created Equal

Unless you're working directly with AI, the terminology can seem a bit confusing. So let's first look at the most common AI terms you're most likely hearing.

- **Generative A (GenAI or GAI)I:** AI models trained on large volumes of data to generate new data or content independently. The key capability of generative AI is synthesizing novel outputs, such as images, videos, text, or other multimedia, that are realistic but not simply copied from the training data.   Midjourney and DALL-E are great examples of Generative AI used to create new and unique images rather than just pulling from pre-existing content.

- **Foundation Models:** Foundation models are large, pre-trained machine learning models that serve as a foundation for building diverse AI applications. Foundation models like BERT and GPT-3 are trained on vast datasets in an unsupervised manner to capture general patterns about the world, after which they can be adapted and fine-tuned for a wide range of applications.

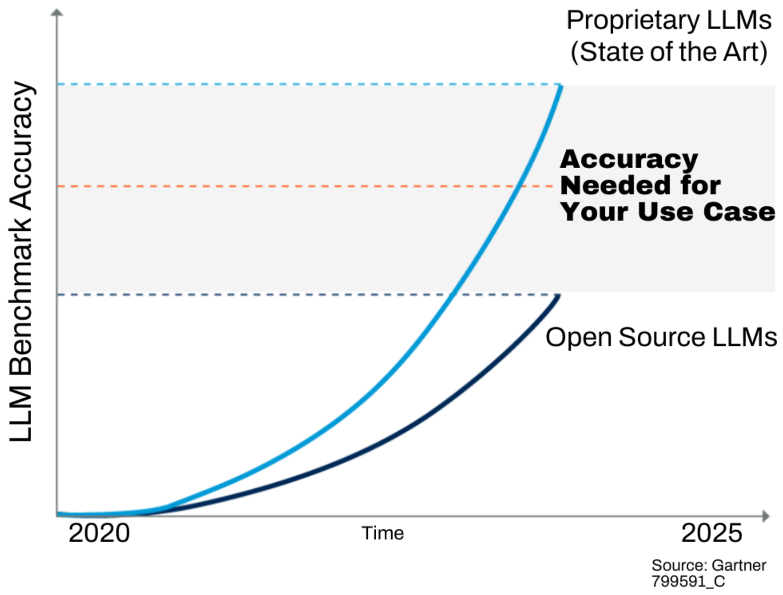
- **Large Language Models (LLM):** AI trained on vast amounts of text, allowing it to interpret and generate humanlike text.

- **ChatGTP:** Undoubtedly the most common term used in relation to AI. ChatGTP is a product from OpenAI that marries a conversational chatbot with an LLM to create content. Trained on a foundational model of billions of words from multiple sources and fine-tuned by learning from human feedback.

Open-source models may seem like a cheap and easy fix, but that is highly dependent on the use case. If you need more focused and specialized datasets, a proprietary LLM trained on data specific to your use case is the better option.

That's not to say that open-source LLMs won't see improvement; in fact, we should expect to see more specialized LLMs slowly emerge, but for the foreseeable future, proprietary LLMs will maintain a significant lead in efficacy.

Open-Source and Proprietary Large Language Model Accuracy Gap



Why Use AI for AppSec?

With artificial intelligence being one of the most hyped technologies of recent years, it's no surprise that companies are rushing to incorporate AI capabilities into their products. However, while in some cases integrating AI leads to truly enhanced functionality and performance, there are certainly instances where its inclusion seems to be primarily a marketing ploy to take advantage of the AI hype cycle. While AI can genuinely enhance products, businesses must ensure they integrate it responsibly and not just for hype.

AI has been an active part of cybersecurity for some years now. A great example is the transformation of the endpoint security space. The dirty secret in the anti-virus industry was that despite behavioral rules and heuristic detection methods, most protection was done by humans sitting in a room writing signatures for each new attack. Then, vendors like Cylance and others introduced an AI-based method of detecting new threats on the endpoint, and the rest of the industry quickly followed suit. Now if you aren't using AI on the endpoint, you're considered outdated and "last-gen". Given that AppSec faces a similar battle against an active cybercriminal community, it's only natural to adopt a similar approach.

Where to Apply AI?

The most logical place to use AI in application security is Static Application Security Testing (SAST). Software Composition Analysis (SCA) tends to be more straightforward from an analysis perspective, as the security issues associated with open-source libraries are fairly well-documented and frequently updated.

Detection

The predominant methods for detecting vulnerabilities in SAST today are very similar in concept to the earlier days of antivirus. While there are technical nuances to how most SAST vendors scan code, ultimately, it comes down to combining heuristics, pattern matching, and a degree of human analysis. While this can do a passable job, the level of accuracy from both a false positive and false negative perspective can suffer dramatically. While not optimal, many organizations accept this lower efficacy, thinking that some results are better than not scanning at all. This can have the unfortunate side effect of developers ignoring results due to high false positives, effectively making some SAST tools a “checkbox” item but not being fully utilized.

AI can help address many of the shortcomings of SAST tools by enhancing the detections they already have in place, but also by utilizing some of the things AI excels at: pattern recognition and learning. Here are some ways AI can improve SAST:

- **Attack pattern recognition:** AI models can be trained to recognize general attack types and find those quickly in closed-source code libraries. Vulnerabilities like buffer overflows, SQL injection, and cross-site scripting have recognizable patterns but can also seem very similar to legitimate code, making it difficult to detect purely by traditional heuristics detection. An AI trained on robust data sets can help avoid false positives and false negatives.

- **Reducing false positives:** Machine learning algorithms can learn to distinguish between real vulnerabilities and false positives based on context. This allows the SAST tool to focus just on the real threats.

- **Zero-day detection:** AI models can be trained to understand secure vs. vulnerable coding patterns by analyzing large code datasets. This knowledge can then assist SAST tools in identifying new types of vulnerabilities, potentially finding new zero-day vulnerabilities before the bad guys know they exist.

- **Adaptive learning over time:** With continuous training on new code, ML models can learn to find new types of vulnerabilities and adapt to changing codebases. This makes the AI-enhanced SAST more robust over time.

The key is to use AI in ways that augment human analysts rather than replace them entirely. The ideal state is having security analysts focus on uncovering the root causes of vulnerabilities and pinpointing classes of attacks ahead of the bad guys instead of scrambling to write signatures each time a new exploit is released. AI helps accelerate SAST, reduce noise, and enable humans to focus on higher-value security issues.

Fixing Code

The next logical step in AI use in SAST is to fix the issues discovered during a SAST scan. AI can be a great tool to assist in providing fixes, but this is definitely an area where human interaction is required. Each individual piece of code has variations and nuances that are particular to that application and implementation, which can be lost on an AI. Users shouldn't expect AI to fix all their issues automatically but should look to AI to help with suggestions and best practices. Here are some areas AI could assist with when it comes to fixing code:

- **Code pattern learning -** AI models can be trained to learn secure coding practices and patterns. They can then suggest fixes when vulnerable patterns are detected.

- **Automated patching -** For common vulnerabilities with established fixes, AI could be trained to make those patches automatically. It is useful for easily fixable bugs, but users may be resistant to automatic changes implemented by AI.

- **Generating fix options -** Instead of just flagging bugs, AI models might suggest possible ways to fix them, such as adding input validation, encoding output, etc.

- **Code hardening -** Machine learning can help add appropriate security controls like rate limiting, input sanitization, etc., based on analyzing app behavior and data flows.

- **Providing code snippets -** AI code completion tools can suggest secure code snippets to developers to help them learn to program more securely.

- **Identifying potential issues associated with a fix -** Before deploying fixes, AI can help assess downstream impacts on functionality and behavior (e.g., the only known fix for a vulnerability is to upgrade a library, but that library would break other dependencies elsewhere in the code)

Challenges to Using AI in AppSec

Many challenges with using AI in AppSec stem from the relative newness of commercially available generative AI tools. With any cutting-edge technology, there are always a few bumps in the road that are eventually smoothed out. We expect to see the severity of these challenges decrease over the next few years as overall adoption increases and tools go through more iterations. Here are a few challenges to look out for:

- Training data requirements - Large, high-quality, representative datasets are needed to train effective AI models. Security data can be noisy, imbalanced, and insufficient, requiring a diligent eye on the AI model's training.

- Explainability vs. accuracy tradeoff - More complex AI models like neural networks increase accuracy but reduce the explainability of results. The end user must be able to accurately understand the returned results, or the results will most likely be ignored.

- Potential for bias - Training data or design choices may unintentionally introduce bias into the AI model and cause it to overlook certain vulnerability patterns.

- AI hallucinations - Similar to the previous point, AI models have been known to be "confidently incorrect" at times if not provided the proper training data. The AI is only as intelligent as its training data, and if fed incorrect information, it will give inaccurate results.

- Evolution of attacks - If the AI is trained on current attack types and behaviors, it may miss new attack techniques that emerge in the future. AI requires constant retraining and updating.

- Integration challenges - Effectively integrating AI detection into existing security tools and workflows can require overcoming technical debt and changing processes built without AI in mind. This is particularly challenging for tools looking to "shoehorn" AI into their existing platform.

How to Do it Right

As we look to the future of application security, AI is poised to transform AppSec much like it did the endpoint protection space. However, the industry must embrace AI thoughtfully and responsibly to realize the full benefits. While we should expect to see varying levels of AI implementation, some common points must be included to provide robust tools for AppSec:

- **AI should be trained on a purpose-built dataset:** While open-source LLMs do a decent job of providing a base level of information, there is no replacement for a finely tuned LLM created with real-world AppSec data. Given the wide variety of applications, languages, and programming styles, any LLM used for AppSec must be carefully curated and constantly trained by professionals with experience in code science and application security.

- **AI should be built in, not bolted on:** AI should be a foundational part of the toolset, not an afterthought. Due to the complexity of AI models, inefficient integration of AI will result in high false positive and false negative rates, ultimately leading to mistrust by the end user. In the development space, if developers can't trust the results from the AppSec tool, they will eventually ignore the tool, negating any potential value.

- **Constant training and tuning:** Cybersecurity is constantly evolving, with new attacks popping up every few seconds. The application space is also a developing area, with languages fluctuating in popularity and usage. Add those two factors together, and any AI tool not constantly going through training cycles will quickly become obsolete.

Conclusion

As we look to the future of application security, AI is poised to transform the AppSec space, just like it did with endpoint and network security. This transformation will allow for faster and more accurate detection of vulnerabilities in code and offer potential solutions. However, the industry must embrace AI thoughtfully and responsibly to realize the full benefits. Bolting on AI to an existing AppSec platform may let vendors tick a checkbox but will ultimately lead to suboptimal results. We are at the tipping point of a colossal shift in code security, but it can be challenging to quiet the noise around the hype. However, if you can cancel out the noise, the benefits of a great AI-enhanced application security platform can change the game for you and help you focus on what's important: releasing secure code.

About Qwiet AI

Qwiet AI is the AI-enhanced application security testing platform that provides SAST, SCA, Container Scanning, and Secrets Detection all in one fast and comprehensive scan. Qwiet AI customers benefit from targeted results with scans that are 10x faster and 12x more accurate than traditional application security tools.