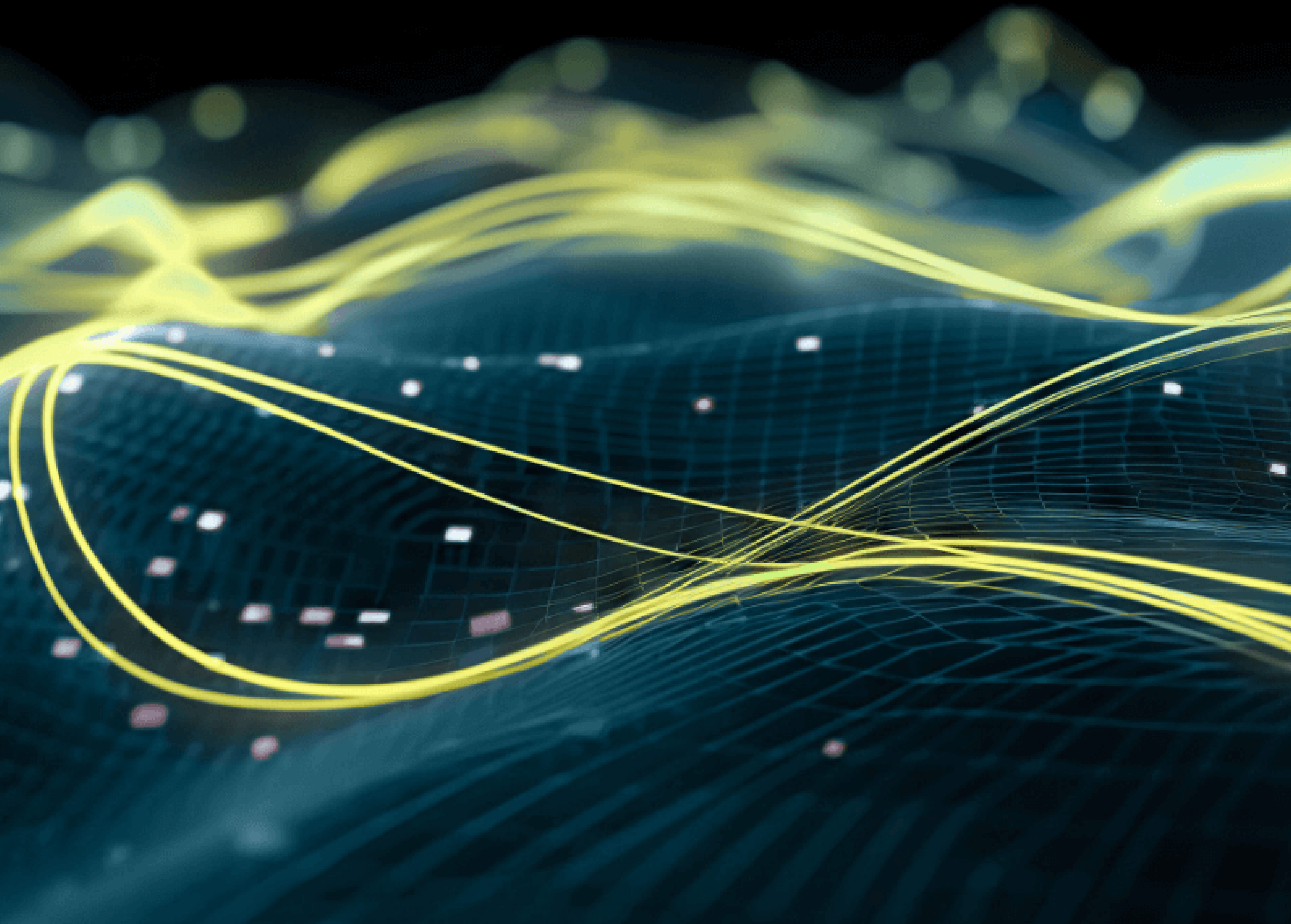
Whitepaper

Veracode Fix: AI Code Remediation Done Right

“With Veracode Fix, developers receive precise guidance, reducing the time and effort needed to secure applications.”

- Sebastian Wilke, Cybersecurity Manager, Banco Galicia



Introduction

Veracode Fix is Veracode's AI security remediation assistant. Veracode Fix takes the flaw findings from Veracode Static Analysis, then creates patches that can be applied directly to your code to fix the flaws, cutting the time taken to remediate flaws down to just a few minutes.

Now, you might be thinking this sounds like every other AI code remediation tool. The difference between tools is what happens between the flaw detection and remediation being applied to the code. **And that matters.**

When developing Veracode Fix over three years ago, we evaluated architectures against criteria for responsibility, reliability, and scalability.

Responsibility:

When you wrote that handy app all those years ago and shared it on GitHub, did you anticipate that it might be used to create AI tools in the future? Do you trust everything out there? What about the solutions that an assistant produces – are there any legal or IP related risks there? Also, what happens to the code you send the AI?

Reliability:

While some use cases for AI thrive on creativity and variability, we're pretty certain that one of the key things you want from a security solution is consistent, reliable results that you can trust. Hallucinations and haikus are not required.

Scalability:

We knew that to be effective, Veracode Fix would need to cover most of the flaws that our customers are finding. That meant covering both multiple flaw types and *different languages*.

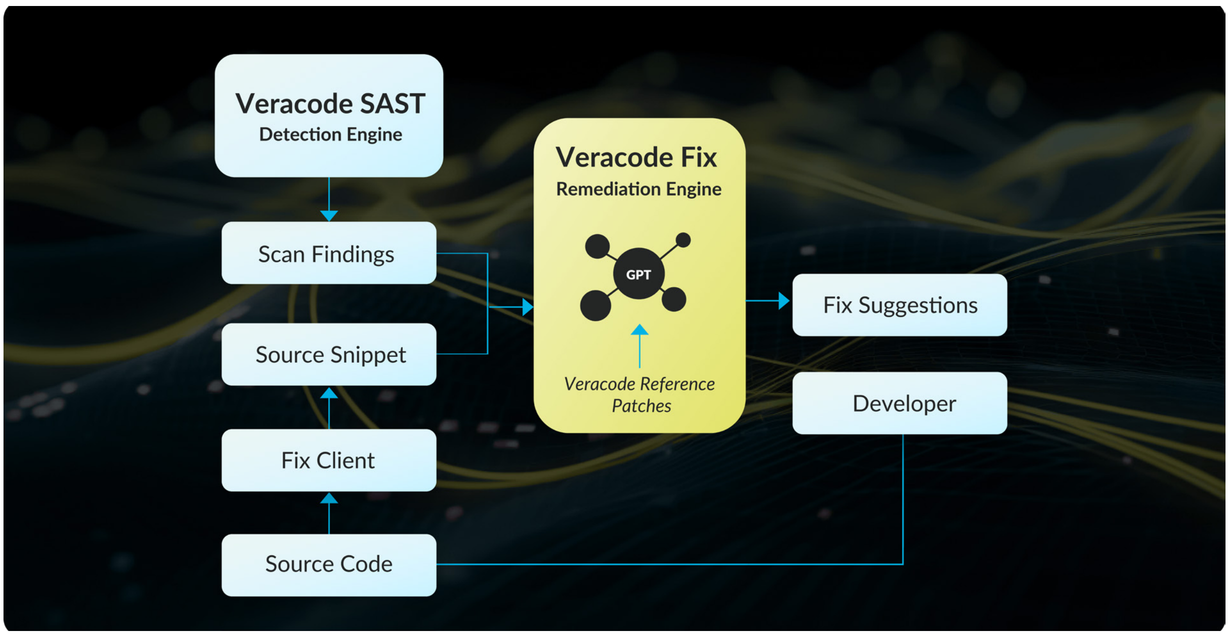
How Does Veracode Fix Address These Requirements?

Responsibility

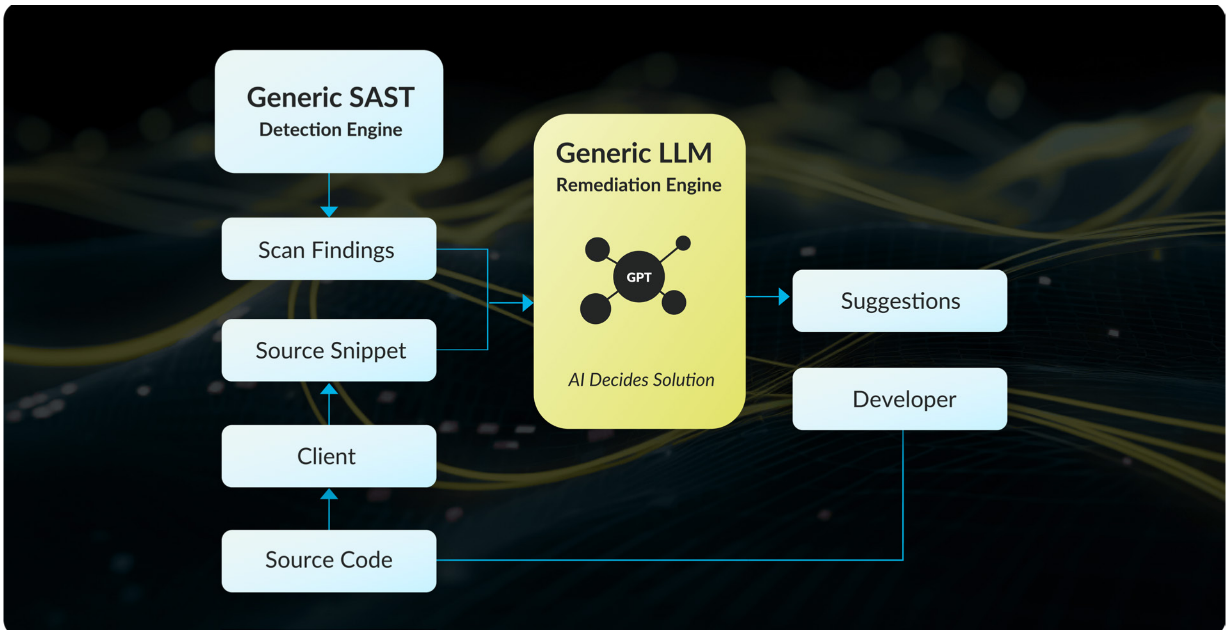
The first and foremost requirement for Veracode Fix was that it be “Responsible-by-Design”. We knew that savvy customers would have legitimate concerns about the source of the model training data, how the model was trained, and how their source code was going to be handled. Mitigating these concerns is achieved by a clever mix of architectural design and responsible data stewardship. As a bonus, many of these design decisions also contribute to the reliability of the results.

The first key design decision was based on the critical function of Fix – how would we use an AI to solve security remediations? While it would be possible to train a model on all the various solutions to common flaws, and simply pass in the code with the flaw along with a flaw description or Common Weakness Enumeration (CWE) ID, this methodology has some drawbacks:

- **Training the model on security best practices would place a lot of ‘responsibility’ on the AI’s ability to correctly interpret code flaws, and solutions.**
- **Adding new flaw type coverage would require altering the underlying model.**
- **The results might have too high a degree of variability.**



Instead, Veracode chose to create a model that was fundamentally unaware of security flaws and best practices but was trained simply to rewrite code to match a given template.



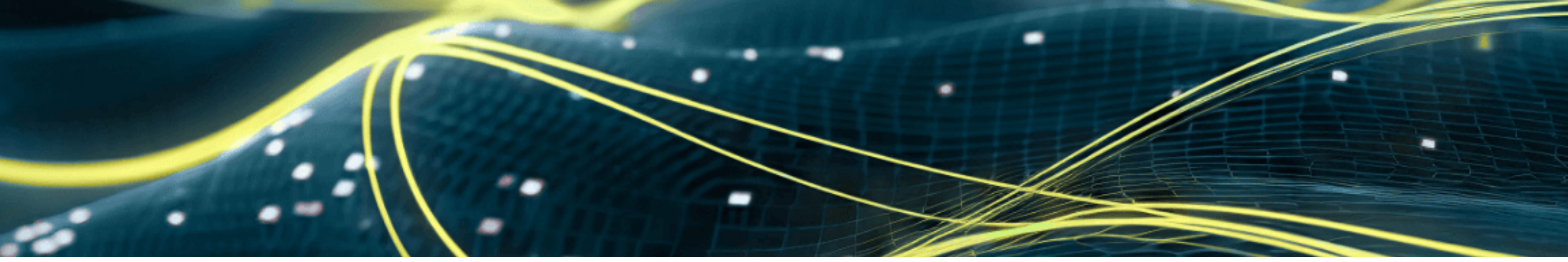
This might seem counter intuitive but has some significant advantages:

- **The solutions to a given CWE are designed by Veracode's expert security researchers, so we can be highly confident that they are correct.**
- **Creating or amending flaw solutions is done outside of the model – making tuning, extending, or correcting remediation separate from the underlying model.**
- **The model can be built with a highly constrained use case – trained to make one piece of code model a given solution that we have named a 'Reference Patch'. The model does not need to be trained to understand how to identify a SQL injection flaw and to devise a solution – just to apply a supplied reference patch to a code snippet.**

Another key area of concern might be the source of training data for a model. How well curated is the data, what is its provenance, and does using it create any intellectual property risk? The exact nature of any liability for models based on scraping open-source code repositories is, at the time of writing, not fully legally tested.

Veracode chose to implement an architecture that insulates our customers from any risks around intellectual property, copyright, or licensing. The design of Veracode Fix ensures that supplied code solutions are free from any claims.

Of course, you also need to worry about your organization's IP, since the solution obviously needs to access it to make changes. Veracode Fix is designed to require only the smallest effective snippet of your code and retain it for the shortest viable window. **Customer code is never used to further train the model**, and the activities of other customers do not affect model behavior.



Reliability and Consistency

Most organizations would prefer their code base to be consistent and follow set standards. Methods for sanitizing input and output, making database queries, etc. should be uniform, and a standardized set of third-party libraires used for routine functions. Any system that adapts code to solve security flaws should, therefore, be similarly consistent.

Obviously, you want security solutions to be consistently correct – to both solve the issue, and to not alter how the code works, or create compilation or other errors.

Veracode's experience in building Fix was that the best way to achieve consistency and reliability was to keep humans in control of the template used to solve flaws. As discussed above, Veracode's architecture of reference patches produces highly reliable and consistent results and is simpler to adapt and tune.

The degree of variability in the results that Large Language Models can produce is essentially eliminated by using a highly constrained model function and controlling the inputs (prompts) to a very high degree.

It's also worth noting that the Static Code Analysis findings that Veracode Fix acts upon are highly accurate. After all, it would be a shame to send results littered with false positives to the AI, even if it does not have the same capacity to get frustrated as a live developer does...yet.

Scalability

Extending an AI model trained on solving security flaws involves retraining the model – which is time consuming and should involve significant regression testing. The same is true for improving results or fixing defects. Where several models are involved in a multi-step process, this problem is obviously potentially magnified.

With Veracode's reference patch architecture, extending or tuning remediations involves adapting reference patches, which are part of the prompt, not the model. Not only does that make extending Fix to cover new vulnerabilities less risky, it makes quick tuning of existing virtual patches relatively easy.

This architecture provides a highly efficient way to scale Veracode's huge wealth of internal experience in solving security problems to all customers.

VERACODE

As AI supercharges your developers, it's going to be important to make sure your security remediation can keep up. Using AI to dramatically speed up remediating flaws can help stop security becoming a bottleneck, but choosing the right tool is going to be critical. There are sure to be plenty of tools designed quickly as a shim to existing GPT models that will appear to answer the problem but might not deliver, or worse, might introduce additional risk.

Make sure that your chosen solution is responsible, reliable, and scalable. We think Veracode Fix fits the bill, but we'd encourage you to try for yourself.

For more information on the Veracode Platform

Get a Demo

For more information on the Veracode Fix

View Now