BRIEF

Defining secure code

> The developers who create the software, applications and programs that drive digital business have become the lifeblood of many organizations. Most modern businesses would not be able to (profitably) function, without competitive applications and programs, or without 24-hour access to their websites and other infrastructure.

And yet, these very same touchpoints are also often the gateway that hackers and other nefarious users employ in order to steal information, launch attacks and springboard to other criminal activities such as fraud and ransomware. The latest Verizon Data Breach Investigations Report highlights that the threats arrayed against businesses and organizations are more dangerous and expensive today than at any other point in history.

Successful attacks remain prevalent, even though spending on cybersecurity in most organizations is way up, and even though movements like DevSecOps are shifting security towards those developers who are the lifeblood of business today.

Developers understand the importance of security, overwhelmingly want to deploy secure and quality code, but software vulnerabilities continue to be exploited.

Why?

> For the second year, Secure Code Warrior conducted *The state of developer-driven security survey, 2022* in partnership with Evans Data Corp in December 2021. We surveyed 1,200 developers globally to understand the skills, perceptions, and behaviors when it comes to secure coding practices, and their impact and perceived relevancy in the software development lifecycle (SDLC).

The survey identified an absence of a clear definition or an understanding as to what constitutes secure code. It turns out that there is a big discrepancy between what developers *think* is secure code, and what secure code *actually* is.

It was not surprising that writing quality code was a top priority for the development community. But when asked specifically about secure code, **only 29% said that active practice of writing code that was free of vulnerabilities was prioritized**. Instead, developers associated less safe and far less reliable practices with the creation of secure code. For example, scrutinizing existing code (37%), and relying on externally sourced libraries for safe code (37%) were the top practices that developers associated with secure coding. Reusing code that had already been deemed to be secure (32%) was another popular choice. The active practice of writing code that is free from vulnerabilities came in 6th with 29% stating this was a top practice in the creation of secure code. When questioned further, **a lack of time and a lack of a cohesive approach from management** were stated as the top barriers to create secure code.

A reliance on existing code is one of the factors that increases the risk of software being shipped with exploitable vulnerabilities. Addressing this disconnect of what constitutes secure code is necessary for developers to create quality code that is also secure.

It turns out that there is a big discrepancy between what developers *think* is secure code, and what secure code *actually* is.

What can organizations do to fix the situation?

> One of the overriding messages from the survey was that the developer community as a whole is filled with professional people who care about what they do. Writing top quality code was overwhelmingly important to them as a group. The problem is that in many cases, the organizations they work for have not identified what best practices are required to produce secure code, and have not put enough resources into training or enabled their developers to meet those goals.

In fact, most developers stated that their organizations did not even have a clear definition of what constitutes secure code. One of the most worrying examples of this was that 28% of the survey respondents said that their organization considered code to be secure if no breach was reported once an application or program was deployed into a production environment or made available to the public.

It probably goes without saying, but in today's complex threat landscape, simply hoping for good results without actually working toward them will likely produce predictable results: even more security breaches.

Thankfully, this is a situation where it's relatively easy to at least get started with fixing the problem, and then to begin to work towards the goal of secure code. The first and arguably most important step is for organizations to define what they consider to be secure code. And everything that is outside of that definition needs to be deemed as not secure.

Secure coding should be defined as the practice of skilled developers writing code that is free from vulnerabilities, from the start of the SDLC. Only once this practice is defined can the developer community work towards that goal.

Making the goal of secure code a reality

> Once the definition of secure code is established, organizations need to be ready to support those efforts and their developers who will be carrying out the goal of implementing total secure code practices. That support is critical. Without it, the definition of secure code within your organization, while important, will be little more than a paper tiger. Secure coding practices must be endorsed by management and given the proper consideration, authority and budget in order to succeed.

This may require new benchmarking goals for developers, who have traditionally been measured on the speed of their coding. In fact, 37% of developers in the survey reported leaving known vulnerabilities within their code because tight deadlines would not allow for the time needed to fix them, or to code properly from the start.

At first, this may mean increasing deadlines to give developers more time to properly code, although that expenditure in time at the beginning of the coding process will likely be made up later because of less of a need for program revisions, patches and post-deployment work. And eliminating the possibility of a breach one deployed can end up saving hundreds of hours and possibility millions in lost revenue, fines and cleanup costs.

Developers will also require relevant, hands-on training, especially as it relates to specific vulnerabilities that they are likely encoutner, and help with learning how to identify and fix code vulnerabilities. This is especially true in light of 36% of survey respondents who said they wanted to remove vulnerabilities from their code, but didn't have the skills or the knowledge to do so.

37% **of developers in the survey reported leaving known vulnerabilities within their code because tight deadlines would not allow for the time needed to fix them, or to code properly from the start.**

For further reading

Whitepaper

The challenges (and opportunities) to improve software security

Report

The state of developer-driven security 2022

Visit our case studies to find out how we're helping to empower development teams in their quest to write secure code from the start of the SDLC.

About Secure Code Warrior

Smarter, faster secure coding

Secure Code Warrior builds a culture of security-driven developers by giving them the skills to code securely. Our flagship Learning Platform delivers relevant skills-based pathways, hands-on missions, and contextual tools for developers to rapidly learn, build, and apply their skills to write secure code at speed.

Established in 2015, Secure Code Warrior has become a critical component for over 450 enterprises including leading financial services, retail and global technology companies across the world.