WHITE PAPER

Empowering Developers

to write secure code and quash recurring vulnerabilities once and for all

SECURE CODE WARRIOR

securecodewarrior.com

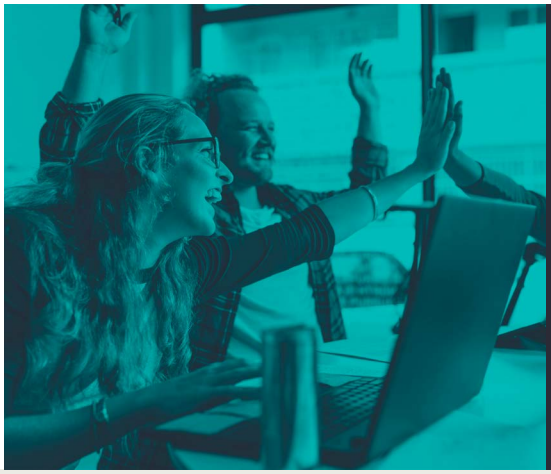
Introduction

Economic growth today is largely based on digital technology, which means that every major business has become a software company in some form. With approx 27 million software developers around the world today[1], they are now the primary architects underpinning the success of many public and private organizations. Within this environment, many development teams have moved from a niche corner of their organization right into the 'hot seat'. They are challenged to rapidly translate business needs into competitive applications that are convenient, trustworthy, and secure.

'Convenient, trustworthy and secure' are easy words to write on a page, but infinitely more challenging to deliver, with the increasing volume of code written every year. In 2021, with more products and services online than ever before, software security continues to be a major challenge, with the threat and reality of breaches growing every day. Agreeing with many concurring reports, Verizon's 2021 Data Breach Investigations Report found that around 25 percent of data breaches today are caused by basic web application attacks, something that has been a consistent finding over the past decade[2].

Despite ever-increasing application security budgets, testing platforms, tools and penetration tests, the number of successful cyber attacks keep rising, while the same recurring software vulnerabilities are routinely detected day after day, year after year. According to a Veracode report, a scan of 130,000 applications over 12 months found 66% had a security flaw that fell into the OWASP Top 10[3]. Astonishingly, SQL injections continue to lead the OWASP Top 10 after more than 20 years.

Security skills and training for developers must now be a constant and positive part of their everyday working routine - writing quality code must mean writing secure code. Improving secure coding skills and outcomes adds a powerful layer of cyber protection for organizations, squashing recurring vulnerabilities and reducing risk. Developers don't need to become security experts, but they must be empowered to be the first line of defense for their organization.



Developers don't need to become security experts, but they must be empowered to be the first line of defense for their organization.

Part 1.

What's wrong with the current approach to software security?

Developers are under pressure to deliver software with as little business risk as possible, but time and resources to help write secure code from the start are limited. With today's pace of development, even the best teams produce the same recurring vulnerabilities, causing delays, cost, and downstream risks that companies can't afford.

At the crux of the problem is the negative mindset in which security is often viewed during software development. It is too hard and disruptive when developers are focused on creating cool functions and features, and is seen predominantly as a compliance and defensive strategy, rather than a positive enabler of growth. Security teams and developers often work against each other and most importantly, the majority of emphasis is on finding flaws in the software, leading to delays and reworks, rather than fixing the problem at the source. Unless security is embedded into developer workflows, software vulnerabilities will continue to pose a huge risk for organizations.

Unless security is embedded into developer workflows, software vulnerabilities will continue to pose a huge risk for organizations.

Insecure software is responsible for a number of successful cyberattacks

Successful attacks are commonplace occurrences, giving victim organizations a lot of bad press. There are regular statistics published about breaches due to flaws in the software, with one US government software assurance program quoting 90% of incidents are caused by defects in the design or code[4].

Some organizations manage to identify serious bugs before they are breached, such as Twitter's warning in April 2018 that an internal defect in their password software had left its 336 million users vulnerable to hackers[5]. The narrative puts the responsibility on the user to change their password, and the company's rectification of the bug, but why is there such easy acceptance that the software being produced and used is vulnerable in the first place?

Security vs. Development is an unfair game

For a long time, balancing the needs of security teams and developers has been a hard slog. Tension begins at the outset, because developers need to create functional code without any security issues, while security's job is to identify as many security problems as possible in order to minimize the risk of exploitation in the future. Few developers are formally taught security principles or are held to secure coding standards. They don't know how to code securely or how to fix the problems that security teams throw at them, as the recommendations provided are often too generic or not usable.

Application security has a tough time too. They can be responsible for security across 50-100 developers, meaning they are often stretched and stressed. Also, although they are experts in security concepts and identifying security problems in code, security professionals are usually not trained in specific languages and frameworks which would have allowed them to modify the developer's code and actually fix the issues they find.

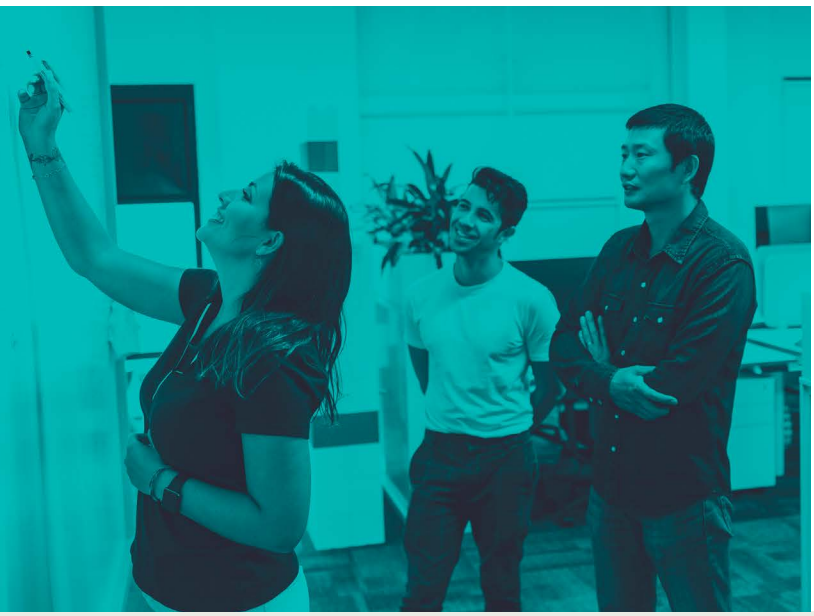
Right-to-left is back to front

Current application security tools focus on moving from right to left in the Software Development Life Cycle (SDLC). This process is good at identifying vulnerabilities in code, but it doesn't teach anyone how to prevent the problems or fix them, so the same recurring vulnerabilities keep occuring. When that happens, organizations are faced with escalating costs and time required to detect and fix vulnerabilities in committed code as opposed to preventing them in the IDE (Integrated Development Environment).

Current security tools and processes can't keep up with DevSecOps

In recent times there has been a real focus on short release cycles, with the emphasis on getting code out quickly and regularly, so a DevSecOps approach, which integrates development, operations and security teams under one umbrella with a common security goal, is becoming increasingly popular. In a DevSecOps world, there is a need to find security bugs within shorter release cycles, but many of the tools being employed are not able to do this. For example, executing a penetration test takes two weeks, and by the time the results come in, the version of the software and the bug are both already outdated.

Most tools only point out problems, but don't offer guidance on how to validate them, correct them, or how to write secure code.



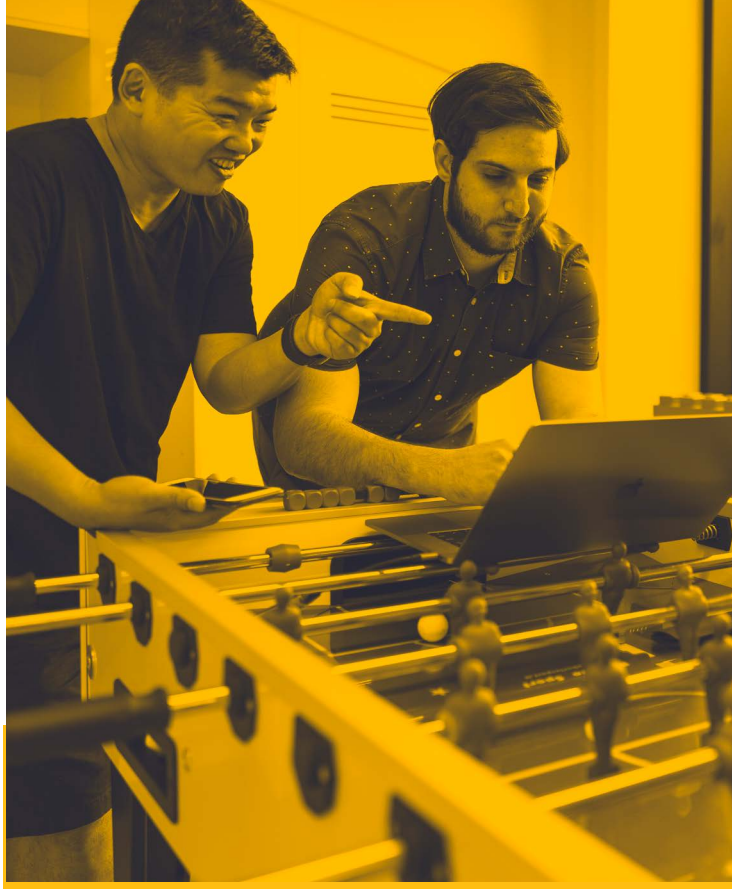
Maturing AppSec programs is a slow process

Apart from large, highly regulated industries such as banking and finance, most formal software security programs begin only after an organization has been breached. At this point, organizations will generally start a program immediately to understand if there are more of the same issues and contain them. Penetration testing and code analysis would be common practice at this stage. Once they start testing, security will normally find dozens or even hundreds of issues in the code. Some companies will try and deal with the OWASP Top 10, but many will only try to deal with the OWASP Top 2 or 3.

As companies mature their AppSec programs, they generally "shift left," closer to the start of the SDLC. They might invest in some white hat hackers, then shift further left and employ a static code analysis solution. Still, they are constantly finding hundreds of potential problems in their completed code, which are both time consuming and expensive to fix. There will also be many false positives and false negatives, and they will keep finding the same problems, which is frustrating for everyone. Even when companies move bug-finding all the way into the developer's IDE, most tools only point out problems, but don't offer guidance on how to validate them, correct them, or how to write secure code from the start.

We know lots about the problem; it's time to focus on the solution

It is important to have tools and processes to look for problems in the code and many companies are improving their ability to do this. However, for every bug found, developers are forced to go back into old code, re-familiarize themselves with it, learn about that problem, work out how to fix it, and then push it back into the system, which then triggers a new quality assurance process.



ADOPTING A POSITIVE SECURITY CULTURE

Apart from the obvious overhead, imagine how a developer feels knowing they are encountering the same issue many times, but they aren't able to find and fix all the other instances until they receive them individually via a bug report. Each time they receive a bug report they must stop their code building and enter the same cycle again, without the ability to share the solution with other developers, who may be figuring out the very same solution by themselves somewhere else.

A new strategy is needed: one that enables developers to embrace a preventative secure coding approach that empowers them to deliver quality, kick-ass code, with confidence. When that happens, development teams can eradicate recurring vulnerabilities and minimize reworks, leading to improved productivity and delivery of secure, quality code.

Part 2.

Empowering developer security requires a positive approach

At the risk of oversimplifying things, the first step is for an organization to decide that instead of perceiving software security as a burden or an impediment to innovation, it will be embraced as an opportunity for development teams and organizations to write better, more secure applications.

Developers as security heroes: The opportunity

Although every developer is unique, it is fair to say that software development is a true craft, and most developers take immense pride in their work. Developers are creators, keen to solve complex, abstract problems and build clever features. They get excited about shipping an excellent product, and although there is enormous pressure to build applications quickly, no one wants to produce a shabby product.

Both security and development teams need a shift in the way they think and operate, enabled by innovative tools and techniques that encourage secure coding from the start.

What does empowering developer security mean?

Empowering developer security moves the focus from reaction to prevention. Rather than dull "compliance-based" training and a negative focus on pointing out every perceived and real flaw, the approach evolves to see security issues from a developer's perspective. This means making it relevant and engaging, while genuinely teaching developers how to identify and fix their own issues, as well as learning how to securely construct their own code.

Can you imagine if developers were taught to write secure code in real time, and to avoid creating many of the bugs in the first place?

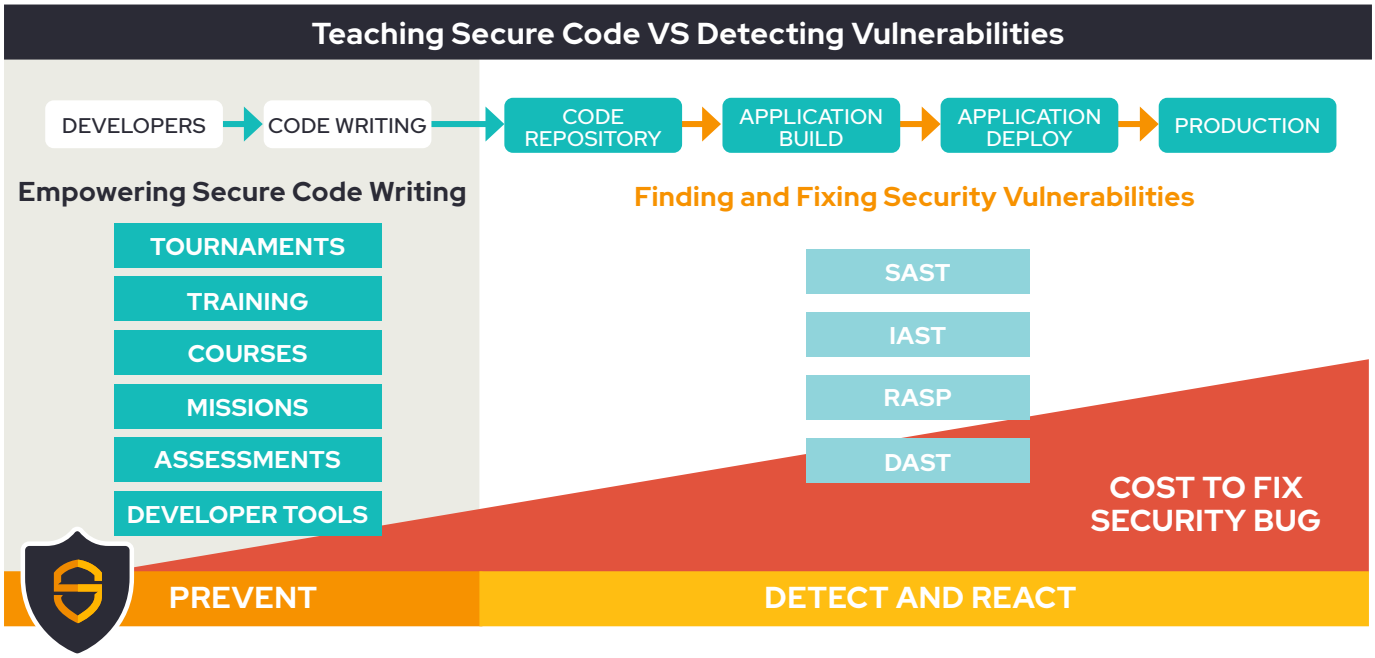
What if they could fix secure coding problems as they arise? And what if your highly skilled security managers and testers could focus on finding and fixing challenging, complex bugs rather than sending developers after nebulous minor issues with limited instructions?

The developer persona is clever, problem-solving, and keen to build their skills. This must be applied to security training and tools, rather than the current model of boring videos and CBT training – tools that focus just on finding faults. This approach will not replace security experts and does not require developers to be security professionals themselves. However, it flips the strategy from tedious training followed by continually scanning and finding errors, to genuinely motivating and aiding developers to make their code secure at the most efficient stage of the development cycle.

Teaching developers to write secure code vs. detecting vulnerabilities

As previously mentioned there are many solutions that find vulnerabilities in code. We need more emphasis on opportunities for developers to learn and understand how to bake security into their code from the start.

Developers should be assisted to write secure code and fix any errors they make as they are writing code, and the technology now exists to achieve this. Just as spelling and grammar correction tools assisted in the real-time writing of this whitepaper, specific to the American English it was written in, developers can be helped in real-time to write securely in line with the relevant language and security policy.



OWASP has an important role to play in empowering developer security

Many organizations rely on the OWASP Top 10 for direction in their AppSec programs. Indeed, SCW supports the OWASP Top 10[6] through our flagship Learning Platform which also allows users to customize their security policy in alignment with this and other industry standards such as PCI-DSS, NIST and ISO 27001. The OWASP Top 10 is useful, but like scanning and testing tools, it focuses on pointing to the vulnerabilities rather than teaching developers how to identify and fix problems as they code, and how to follow secure coding guidelines so they can prevent bugs in the first place. But with the SCW Learning Platform, developers can learn how to find and fix vulnerabilities highlighted in the OWASP Top 10, plus the Mobile Top 10, API Security Top 10 and CWE/Sans Top 25 Most Dangerous Software Errors 2020[7].

OWASP also plays a role in helping companies to empower developer security. For two decades OWASP has created a global community of like-minded people helping each other. It provides free assistance to coders worldwide, including cheat sheets, scanning tools, secure coding frameworks, and testing environments. Its numerous consensus projects are powerful too, because the community decides what is good and bad practice. OWASP's Application Security Verification Standard (ASVS) Project[8] and the OWASP Top Ten Proactive Controls[9] are both making a tangible and valuable contribution to evolving developer secure coding education and skill development.

OWASP ASVS[8]

Use as a metric	Provide application developers and owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications
Use as guidance	Provide guidance to security control developers as to what to build into security controls to satisfy application security requirements
Use during procurement	Provide a basis for specifying application security verification requirements in contracts

OWASP'S APPLICATION SECURITY VERIFICATION STANDARD (ASVS) provides a basis for testing web application technical security controls and provides developers with a list of requirements for secure development.

The Top 10 Proactive Controls[9]

- **C1:** Define Security Requirements
- **C2:** Leverage Security Frameworks and Libraries
- **C3:** Secure Database Access
- **C4:** Encode and Escape Data
- **C5:** Validate All Inputs
- **C6:** Implement Digital Identity
- **C7:** Enforce Access Controls
- **C8:** Protect Data Everywhere
- **C9:** Implement Security Logging and Monitoring
- **C10:** Handle All Errors and Exceptions

The OWASP TOP TEN PROACTIVE CONTROLS 2018 is a list of security techniques that should be considered for every software development project. One of the main goals is to provide concrete practical guidance that helps developers build secure software. These techniques should be applied proactively at the early stages of software development to ensure maximum effectiveness.

Part 3.

How to make secure coding training and tools simple, relevant, and engaging for developers

Developers can be empowered to be the first line of defense in their organization by making security highly visible and providing them with the skills and tools to write secure code at the most efficient stage of the SDLC. This section describes a tested approach to improving secure coding skills and outcomes that is simple, scalable and positive, for development, security teams and operations.

TOURNAMENTS: Building a positive learning experience that drives engagement

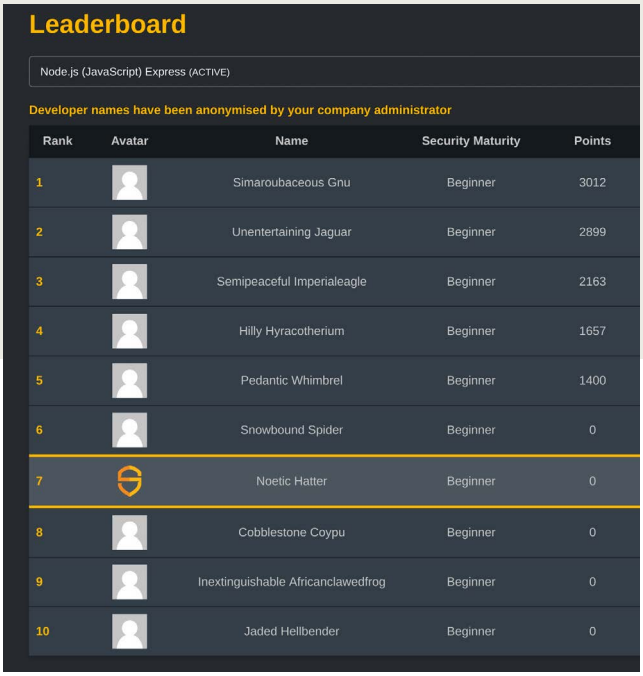
Secure coding tournaments are a fantastic way for companies to build awareness and enthusiasm for secure coding. They can be highly engaging face-to-face or virtual events that get the whole developer community involved and started on a secure coding journey. Tournaments can be short – only a few hours – or run for days, and include a series of hands-on coding challenges and immersive missions where developers identify problems, locate insecure code, and fix the vulnerabilities.

Challenges should generally be based upon common vulnerabilities in your code base and enable developers to compete in the software language:framework they are currently working in to ensure it is relevant. Throughout the tournament, developers earn points and compete to climb to the top of the leaderboard. There are many exciting examples of companies holding innovative tournaments that attract developer attention and drive participation. It also helps companies to find security champions in each scrum team, which is important to ongoing skill building.

They can be highly engaging face-to-face or virtual events that get the whole developer community involved and started on a secure coding journey.

Secure Coding Tournaments

- Makes it fun!
- Keeps teams focused
- Builds awareness
- Identifies security champions



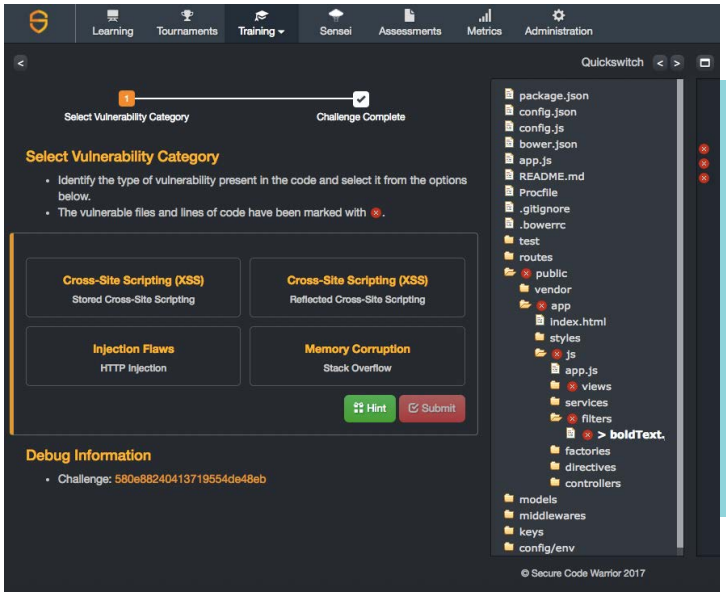
TRAINING: Make it relevant, hands-on and fun, and developer security skills will grow fast

Given security is not the developer's priority, any security training must be fun, competitive, and engaging to motivate them to 'play.' The best learning platforms are gamified and allow developers to earn points and collect badges, with leaderboards for teams.

Training platforms must focus on building specific skills, actively engaging developers to learn, and build their secure coding skills. Developers should be able to work in real code and in their own language:frameworks, to locate, identify and fix known vulnerabilities in code.

Challenges should be short and cover all the common vulnerabilities. They must be constantly expanded and updated so developers can continue to build their skills regularly over time, with a range of increased difficulty for senior developers and the opportunity for assistance.

Enabling developers to view their progress throughout their journey is also important. Both developers and their managers should be able to see which challenges they have completed, their strengths and weaknesses, time spent on training and their accuracy.



Training platforms must focus on building specific skills, actively engaging developers to learn

EXAMPLE:

60% increase in secure development capability in 2 months

One of Secure Code Warrior's customers required their developers to play a single challenge (5 minutes) every day for two months. It tested their skills before and after the training period and observed a 60% increase in secure development capability over a group of hundreds of developers. This meant less resources spent on finding and fixing security bugs later in the lifecycle, and significant long-term savings.

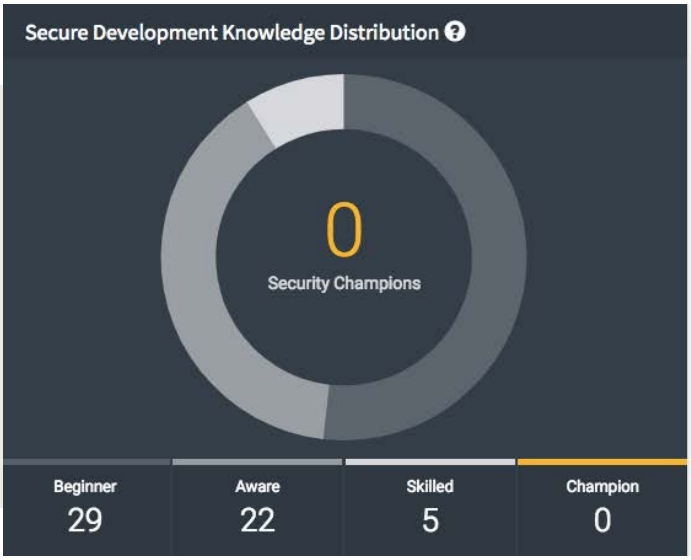
COURSES: Skills-based pathways that lead to intrinsic security

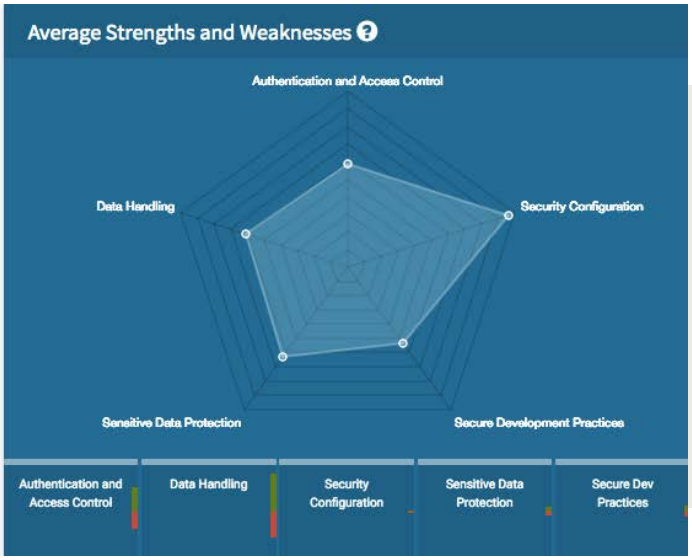
Curated courses not only eradicate vulnerabilities most relevant to your business, but they also introduce a pathway to achieve compliance and audit requirements. Courses should be relevant and timely, and empower developers to level-up their secure coding skills through adaptive, interactive learning modules and benchmarking.

ASSESSMENTS: Identify gaps and create strong evidence that secure coding skills are improving

Managers need to be confident that their developers have a base level of competency when it comes to securing their code. Assessments allows organizations to qualify and baseline the secure coding skills of their existing developers, offshore developers, new hires, and graduates. How easy would it be if you could identify developers who you were confident could write securely in your new project, before they have even started working on it?

Assessments also demonstrates to auditors that developers are learning the necessary secure coding skills that regulations like PCI-DSS (Payment Card Industry - Data Security Standard) outlines. Some companies are now starting to create an internal badge, belting or certification program that aligns with demonstrating certain security achievements.





DEVELOPER TOOLS: Real-time, scalable and productive

As developers code, they focus on staying in the zone while tuning out distractions.

Bringing validated, contextual corrections and learning to the developer while they code, improves developer productivity. Meaning less time is wasted searching Google for fixes and more time focused on creating quality code.

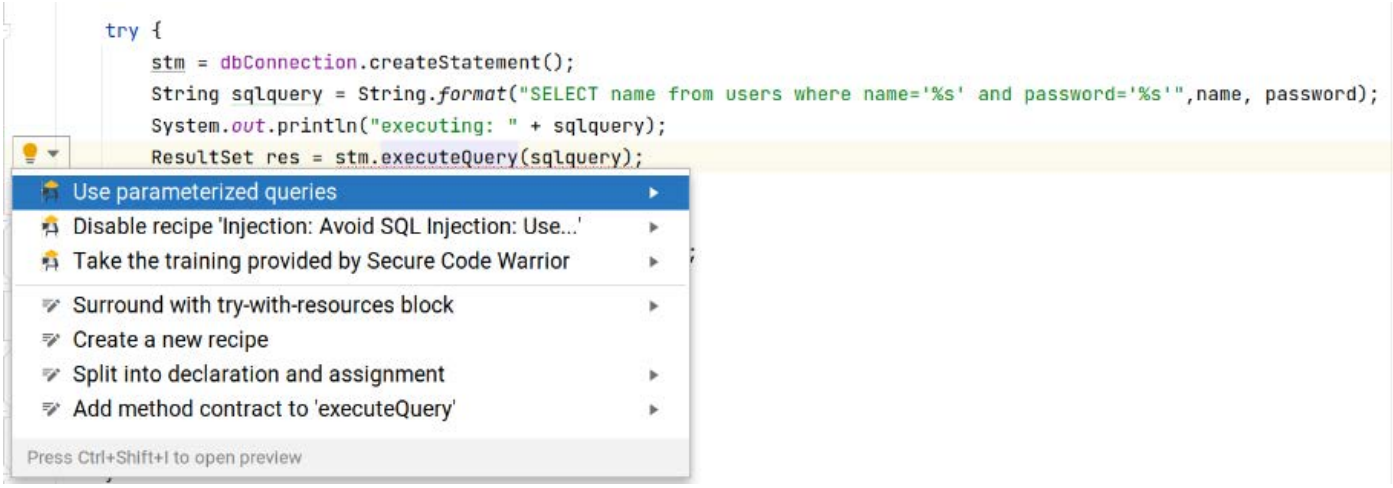
Development teams benefit from Sensei, as it observes and suggests fixes that adhere to coding guidelines directly in the IDE.

The more development teams use a real-time code improvement tools, the better they will become at secure coding and spend less time and money on fixing recurring bugs and reviewing code.

SENSEI

Sensei

Sensei is real-time code analysis, review and remediation tool that enables developers to fix unacceptable coding patterns as they type. Instantly transforming 'bad code' to 'good code', Sensie increases developer productivity by breaking the cycle of poor coding practices and helping teams ship quality, secure, code faster.



Conclusion

Taking Responsibility at the Source

As stated at the outset, the time has come to evolve developer security skills, so they become intrinsic to their workflows. Writing quality code means it must also be secure.

Developers need to take more responsibility for security, and a significant opportunity exists for companies to improve productivity and decrease risk by enabling this kind of approach.

The solution not only involves building skills, but also having the right toolset to help every step of the process, from the first line of code until the last and beyond.

The powerful combination of Secure Code Warrior's Learning Platform and the SCW Sensei will assist security and development teams to ship quality code with confidence.

References

1 https://www.daxx.com/blog/development-trends/number-software-developers-world
2 Verizon 2021 Data Breach Investigations Report
3 Veracode, State of Software Security v11
4 https://www.us-cert.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf
5 2018 Verizon DBIR report, https://www.verizonenterprise.com/resources/reports/rp_DBIR_2018_Report_execsummary_en_xg.pdf
6 OWASP Top 10, www.owasp.org
7 https://www.sans.org/top25-software-errors/
8 OWASP ASVS, www.owasp.org
9 OWASP Top 10 Proactive Controls, www.owasp.org

ABOUT SECURE CODE WARRIOR

Secure Code Warrior makes secure coding a positive and engaging experience for developers as they increase their software security skills. Our flagship Learning Platform delivers relevant skills based pathways for developers to write secure code at speed; whilst intelligent and contextual developer tools fix security flaws in real-time.

Our vision is to inspire a global community of security-conscious developers to embrace a preventative, secure coding practice that enables them to ship quality code faster - so they can create kick-ass software whilst benefiting from improved productivity, reduced costs, lower risk and easier compliance. Established in 2015, our customers include major financial institutions, telcos, retail, governments and global technology companies across Europe, North America and Asia-Pacific.

info@securecodewarrior.com