Table of Contents

04 Understanding the Developer’s Perspective

05 Deciphering Developer Secure Coding Education

06 Gamification

11 Contextual Learning

12 Summing Up
Foreword

Following a paper released by the US Department of Homeland Security, software security is fundamentally a software engineering issue that must be addressed systematically throughout the software development lifecycle.

It is commonly agreed upon that security starts with software engineers, yet, according to recent research conducted by Node.js® and Sqreen, 60% of software engineers are not confident in the security of their own applications. This goes hand in hand with the findings of SANS’s 2016 State of Application Security report stating that the lack of application security (AppSec) skills, tools and methods is one of the top three challenges to implementing AppSec.

Where does the lack in AppSec skills originate? On one hand, a study by cloudpassage found that only one of the top 36 US computer science programs requires a security course for graduation. On the other hand, according to StackOverflow, 69.1% of software engineers are self-taught. This brings us to an obvious conclusion: organizations that wish to ensure their software engineers deliver secure code need to provide them with top-notch secure coding education (SCE).

On a positive note, organizations understand this. According to SANS, businesses see developer training as the most useful AppSec process, even ahead of vulnerability scanning. Unfortunately, despite organization’s wide adoption of various developer secure education programs and methods, the skill gap is still more like an abyss.

This guide aims to bridge the gap between an organization’s need for secure code and a software engineer’s lack of training adoption. It will walk you through everything you need to know in order to ensure your software engineers get the most effective SCE - the kind they will actually use.
Understanding the Developer’s Perspective

In today’s fast-paced development environments, which require rapid delivery with minimum bugs for continuous integration and continuous delivery (CI/CD), a developer’s most valued resource is time. Secure coding solutions that slow developers down will be considered a nuisance and will end up sitting on a shelf, understandably so. It is for this reason that organizations must consider Source Code Analysis (SCA) solutions that fit in seamlessly with a DevOps environment.

Another important factor to remember when addressing the developer’s security skills gap is that developers are primarily hired to write code. You probably won't find “secure code delivery” in most job requirement ads, developer contracts or onboarding processes. Secure coding is often looked upon as “nice to have if you have time” and as mentioned above - time is not an abundant resource. Software engineers are measured by speed and amount of bugs in their code, not the amount of security vulnerabilities.

Despite the above mentioned and with empathy towards the developer’s perspective in mind, we still need our developers to deliver secure code. In order for that to actually happen, the first thing that needs to change is that development team leaders need to start treating security vulnerabilities as they do coding bugs. Developers are well aware of the need to deliver bug-free code and the majority of them will put in an effort to prevent bugs. Once the importance of secure code is established, proper SCE must be implemented.
Deciphering Developer Secure Coding Education

Video tutorials, periodic classroom training and mandatory online courses are common, yet they fail to achieve developer SCE. They are mostly treated as a box that needs to be checked on the to-do list, not as an important tool for securing an application. As demonstrated above, developers have other things on their minds and agendas. Therefore, out-of-context and mundane training is understandably treated with aversion.

In light of the above, how can organizations wishing to invest in developer SCE go about doing so? Gamification and Contextual Learning.
Gamification

Gamification is the application of game-design elements and game principles in non-game contexts. The benefits of gamification in e-training is nothing new, yet most SCE solutions fall behind in implementing it. We learn better when we enjoy doing so, when we take an active role and don’t tire from it. Developers spend most of their day in front of a screen and lines of code. Our experience with thousands of developers taught us that training, which spices things up rather than bores or requires deep concentration, is highly welcomed.
Consider the four following steps when implementing SCE in your organization to ensure proper gamification is achieved:

1. Make it Interactive

Chief Learning Officer could not have phrased it better: “Just because a learner clicks frequently does not necessarily mean learning content is engaging. The learner may be trying to speed through the course to reach completion.” We see this ever so often with mandatory security courses across organizations - Workers are obliged to complete online training on a subject they don’t fully understand the importance of. They have plenty on their desk so they want to go through the mandatory training ASAP and get back to work. This leads to ineffective online courses and quizzes which workers will quickly click through just to get it over with, without really taking in the important content.

“There are several reasons why learning interactivity is desirable. The stories and examples contained in the lessons are an integral part of the overall engagement with the learning. These stories create a situation where the learner feels directly and emotionally involved with the content, which can improve retention” - CLO. Plus, it’s harder to simply click through an interactive training session. Ensuring interactivity will result in developers paying more attention to the content which will then result in a higher chance of actually learning from it. Furthermore, many people learn better by doing rather than just hearing or seeing. Interactivity caters to their needs.
2. Tell a Story

Characters, role-play and a narrative go a long way when it comes to taking in information. There is ample research on how stories can stimulate our brains and spark a reaction. SCE presented in the form of bullet points, questions and answers or mundane text is bound to bore developers. A narrative, characters and a problem that needs to be solved - that is the stuff effective education is made of. Offering your developers characters whose shoes they can step into and a storyline they can follow (in our case - a vulnerability that needs to be solved) are extremely helpful in ensuring they remember what they learned. Storytelling is also the essence of many games and often a fun-catalyst, hence the connection to gamification.
3. Keep it Short

One can argue the claim saying that the human attention span is getting shorter and shorter is not too accurate or scientifically backed. That being said, it is still best practice to keep things short when presenting information. This is as true of a training session as it is of a powerpoint presentation. Shorter content means a higher chance of that content being precise and to the point, it eliminates the chance of irrelevant information sneaking in and raises the chances people will engage with that content to begin with. To top that, add the mantra we have been chanting throughout this paper of time being a scarce resource for developers and the conclusion seems clear - the shorter your training sessions are, the better.
4. Ensure They Win

According to Dr. Ian Robertson’s famous research “The Winner Effect,” the most underestimated brain enhancing agent is empowerment. Dr. Robertson has written on how winning can be better adapted for educational purposes as it releases lots of “good stuff” in our body like testosterone which in turn increases dopamine activity in the brain which makes us feel really good.

In conclusion...

We have concluded that a short, interactive storyline is important to establishing a SCE training program that works. However, don’t neglect the importance of “winning” the training session. End it with a win, such as success in solving the vulnerability. Ensuring your software engineers training is rewarding will in turn ensure they feel good and will voluntarily turn to the solution to find undiscovered vulnerability challenges.
Contextual Learning

Preparing an amazing, gamified SCE program is critical to ensuring developers actually use it and learn from it, but neglecting to make it contextual may ruin all your hard work. We started out by saying how periodic classroom training doesn’t cut it. It’s not just the lack of gamification that fails periodic training, it’s the lack of context. Once you remove the developers from their screens, you remove them from their day-to-day coding routine and make it that much harder to recall a problematic line of code.

SCE should appear at the exact moment when it is needed - while coding. Integrating your SCE solution into the developer’s work routine so that it is just a click away is the secret to successful training. As awesome and as gamified as your SCE content may be, there is no need for your developers to go through it all at once. We found that offering developers ongoing access to our training sessions while they code, fully integrated in their IDE, is what encouraged them to seek it out when encountering a security vulnerability.

Our last tip to you is to keep it contextual in terms of being up to date with the latest coding languages. Secure coding best practices differ between coding languages, there isn’t a one-size-fit-all type of solution. Check out Checkmarx’s secure coding guides such as The Javascript Guide: Web Application Secure Coding Practices or The Go Language Guide: Web Application Secure Coding Practices to learn more.
Summing Up

We hope you reach this summary with a clear vision as to what your secure coding education should look and act like in order to ensure developer’s adoption and skills improvement. Based on the many research examples presented in this guide as well as on Checkmarx’s vast experience with developers, gamified contextual education is key.

If you wish to see all the best practices above successfully implemented, try Codebashing’s online free lessons for developers secure coding education. They combine all the necessary gamification and in-context tips presented in this guide and ensure your developers will not only enjoy their SCE sessions, they will remember and implement them.

Let the SECURE CODING GAMES BEGIN!
About Checkmarx

Checkmarx is an Application Security company, whose mission is to provide enterprise organizations with products and services that empower developers to deliver secure application faster. The combination of Checkmarx technology, methodology and expertise is the most cost effective way to produce high fidelity results, drive developer adoption, streamline remediation, and lead to secure application delivery. Amongst the company's 1,400+ customers are 5 of the world's top 10 software vendors and many Fortune 500 and government organizations, including SAP, Samsung, and Salesforce.com.

For more information about Checkmarx, visit https://www.checkmarx.com

© 2018 Checkmarx Ltd. All Rights Reserved.