Shifting from reaction to prevention: The changing face of software security

SECURE CODE WARRIOR

securecodewarrior.com

Shifting from reaction to prevention: The changing face of software security

Abstract

Organizations continue to see the same software vulnerabilities arise in their code time and again, many of which have been known for decades. This continues to be an issue despite development teams deploying scanning tools to check for security issues, managers tasked with hiring security aware developers, and developers believing they are skilled in secure coding.

This paper explores the reasons why this problem continues to persist, drawing on an in-depth survey of developers' and development managers' attitudes towards secure coding, and analyzes how organizations can stop repeat vulnerabilities from happening once and for all.

In order to accomplish this, we engaged with Evans Data Corporation to conduct a study on how developers and development managers understand how application security practices are changing.

Contents

Summary

Humans are at the core of software development. Even with the rise of technologies like AI and machine learning, behind it all is the human - the creative mind and empathetic heart, designing and implementing these applications from inception.

Despite the many ways in which organizations are battling insecure code through methods such as tooling and pentesting, secure coding skills remain in short supply, current training methods are inadequate and there is a lack of organizational alignment as to who is ultimately responsible for security in the Software Development Lifecycle (SDLC).

Addressing the root of the problem requires a human-led approach. In order to be proactive with implementing secure coding from the start of the SDLC, a human must have the necessary skills to write secure code and keep it front of mind from design to development to deployment. But both developers and development managers find this difficult to do. The difficulty, however, does not lie in implementing security but in learning how to do so when training programs and methods are not relevant to a developer's daily work, are unstructured, and non-engaging.

The survey this white paper is based on was conducted with Evans Data Corp. in August 2020. Participation in the study was limited to respondents who self-identified as software developers or decision-makers who managed software developers. Both were required to be involved in software development in a professional manner full-time. Participants from all over the globe were included in the survey

Because the study was designed to measure the awareness of secure coding practices among the general developer population, experience with secure coding practices was not a requirement for inclusion in the survey sample. This allows analysts to measure general awareness of concepts associated with secure coding.[1]

[1] Secure Code Warrior (2021): *Shifting from reaction to prevention: The changing face of application security.*

Reactive tools vs. proactive humans

The use of scanning tools is still the most common method for implementing secure code.

Despite the fact that code is written by humans and a growing understanding that implementing secure code is a human issue, reactive tools such as scanners still top the list for the most common method of implementing secure code. According to the survey, the five most common secure coding practices are:

1. Scanning applications for irregularities or vulnerabilities after they are deployed

2. Scrutinizing written code to inspect for irregularities or vulnerabilities

3. The reuse of pre-approved code that is known to be secure

4. Ongoing training, education, and awareness of secure coding concepts and common coding vulnerabilities

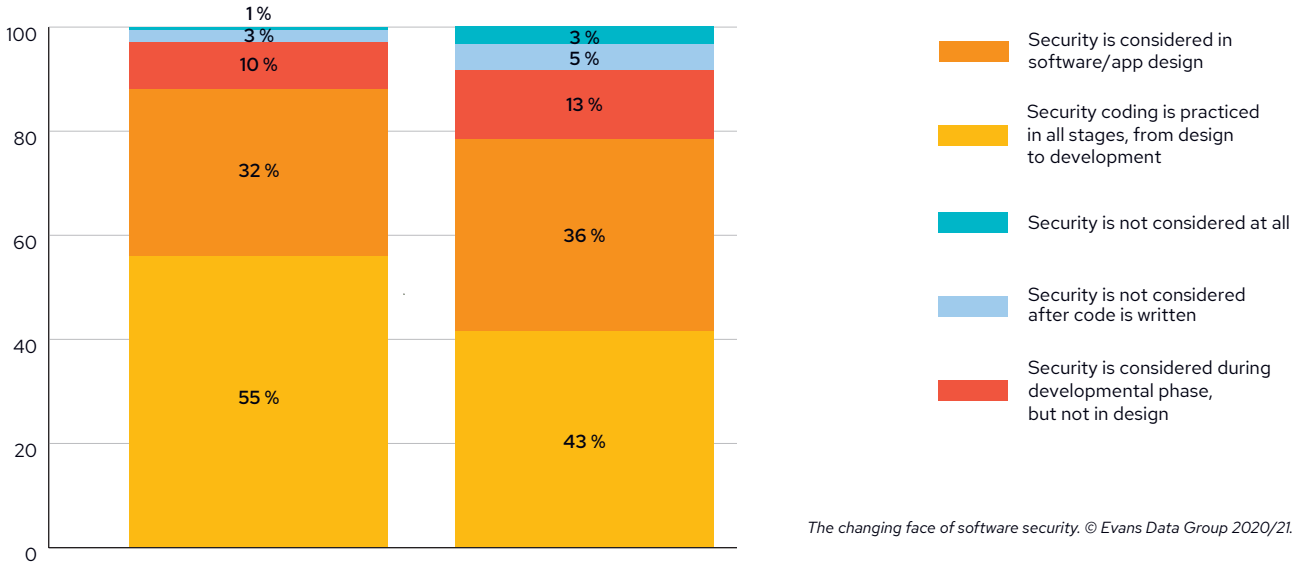
5. The active practice of writing software that is protected from vulnerabilities

The top two practices in this list are both *reactive*. 56% percent of developers and managers believe that secure coding is associated with using scanning tools on deployed applications and another 56% with manually reviewing code for vulnerabilities. Despite the fact that the most common current methods take place after code is written, an understanding is emerging that it is necessary to be *proactive* when writing secure code. Over half responded that they associate secure coding with the active and ongoing practice of writing software that is protected from vulnerabilities in the first place, and that that is done through ongoing training, education, and awareness of secure coding concepts and common vulnerabilities. This understanding is important. It indicates that a shift is coming as a solution to writing more quality, secure code.

The implementation of secure coding

The mindset has not, however, completely shifted. Secure code is not yet always implemented from the start of the SDLC. When developers and development managers were asked when secure coding practices are implemented in the SDLC, their answers did not completely align. 55% of development managers said that security is integrated into all stages of the SDLC (from design through deployment) but only 43% of developers agreed. At the same time more developers believe that security is integrated as early as the design process (see image).

How integrated is application security and secure coding practices into your software development lifecycle?



The changing face of software security. © Evans Data Group 2020/21.

For those organizations that are implementing secure code in all stages of development, the question remains to what extent. To ensure that secure coding is effective in the early stages of the SDLC, developers must be adequately skilled in the first place.

Developers believe they are skilled in secure coding, but still find it hard

Developers believe they are adequately trained in secure coding in general. When questioned, the vast majority (97%) said that they have had adequate and sufficient training and 95% of them said that that training was valuable.

But this self-assessment contradicts their views on how easy they find implementing secure code to be. More than 91% of development managers say that implementing secure code practices is tough and more than 88% of developers find that coding securely is challenging.

So if they receive adequate training, why is this hard? This suggests a need for ongoing clarity of what secure coding actually is, and how to fix vulnerabilities. Developers may also overestimate their current knowledge and skill level. This would also partially explain why more managers say security is implemented across their company's SDLC than developers. Perhaps their confidence in developers influences how well they believe security is implemented.

A more concrete explanation is that both roles believe security to be inherently difficult, and although they believe they have received adequate training, that training has not been sufficient (even if they are unaware of that fact). This opens up new opportunities to compel developers and managers to seek out better training. Implementing secure code should never be difficult when properly equipped with the right knowledge and skills to prevent vulnerabilities.

Developers are highly motivated to improve their secure coding skills

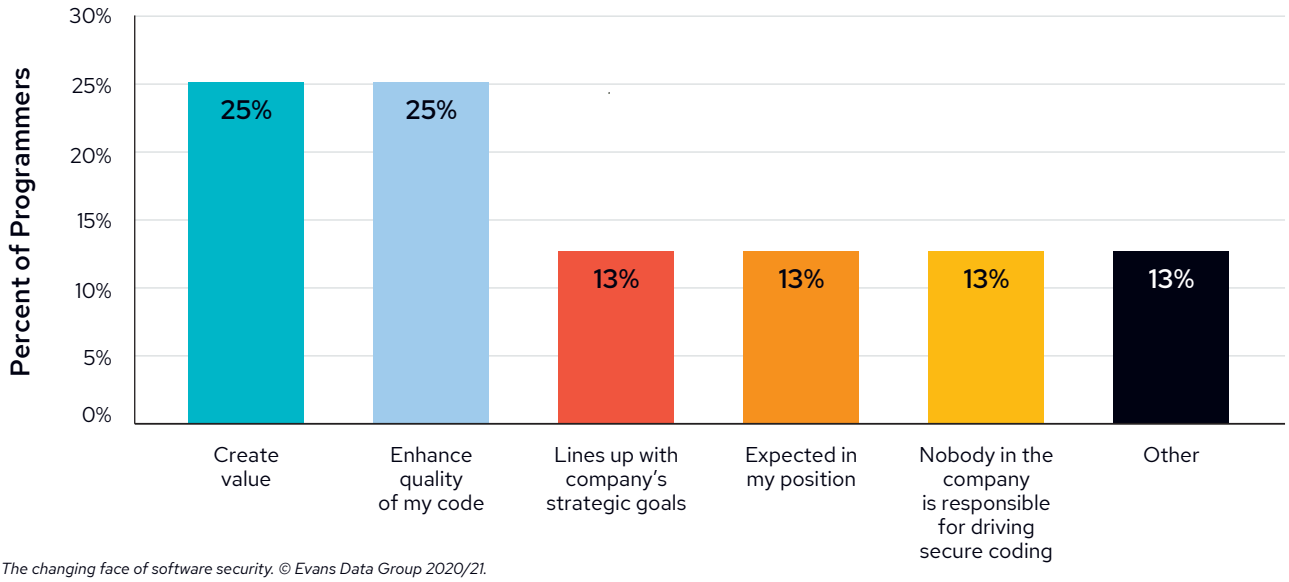
Developers see value in improving their skills and are motivated to learn more. Both developers and their managers understand the benefits of further training both for their current roles at their company and also for their future careers.

The top five motivations to learn secure coding are as follows (see image):

Personal vs. external motivations

25% of developers surveyed in the study say that they would learn secure coding practices simply because they want to enhance the quality of their code. This means that a large number of developers are open to learning purely for self improvement and to be more effective in their roles. And another 25% say they would do so to create value for their companies (see image).

What circumstances would persuade you to study secure coding practices?



The changing face of software security. © Evans Data Group 2020/21.

This reveals a strong sense of pride within the development community and that there is a strive to constantly improve.

While three of the top five motivations for implementing secure code practices are personal drivers, we cannot ignore the influence of corporate mandates or an individual's goal for career progression within their current organization.

81% of participants revealed that they are held accountable for writing vulnerable code and 70% say that they are recognized by their company for writing secure code. Development managers also reveal that developers are incentivized by the company to learn secure coding practices through the following means:

1. The potential for greater responsibility (66.7%)
2. The potential for higher pay (47%)
3. Special events to recognize good secure coding practices (46.1%)
4. Regular skills assessments (44.3%)
5. Offering more relevant or contextual training solutions (35.6%)
6. Coding tournaments (35.6%)
7. Penalties for non-implementation of secure coding practices (14.6%)

Unsurprisingly, over 80% of development managers say they actively seek out and are more likely to hire developers who have secure coding skills.

The good news is that developers have the appetite to improve their skills and should therefore be easy to engage in training or upskilling offered by their employers. In fact, developers often seek out secure coding training on their own. About half of respondents say they learned secure coding through their employers and 42% learned on their own time. However security skilled and minded developers remain hard to find[2], which raises questions over the quality of training solutions being utilized.

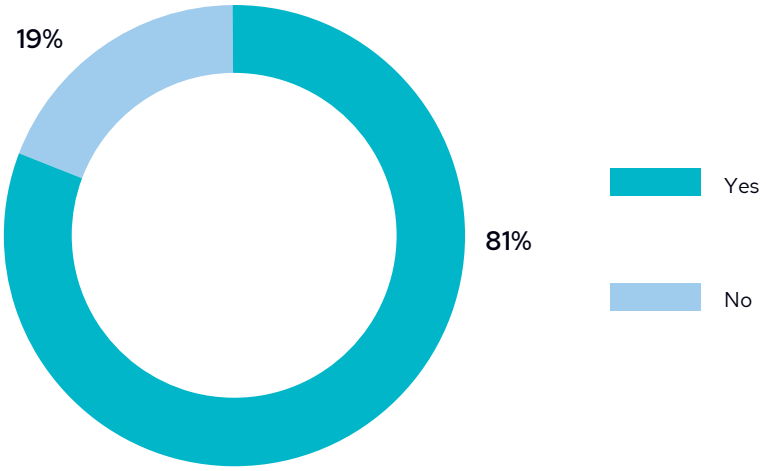
[2] Laurence Bradford. (2018). *Why Developers Will Be More Marketable If They Know This Skill,* Forbes.

Barriers to adoption

Security-aware developers are in short supply

This shortage of developers with secure coding skills remains one of the main barriers for companies to implement secure code. As stated above, development managers actively incentivize their team to improve their secure coding skills and look for developers with those skills when hiring.

As a manager are you more likely to hire developers who have secure coding skills?



The changing face of software security. © Evans Data Group 2020/21.

The issue, however, remains meaning that there is a lack of skill on the market to begin with. Not only that, but it's hard to verify the security knowledge of those potential hires.

When hiring new developers, the managers surveyed reveal that they use several metrics to measure secure coding knowledge. 56% look at past projects, 55% ask appropriate interview questions, and 44% examine whether the applicants are members of professional associations. Lower on this list are: looking at completed classes or certifications, checking references, or a written test. While they are keen to hire developers with secure coding skills, the methods they often use to assess that ability lacks substance.

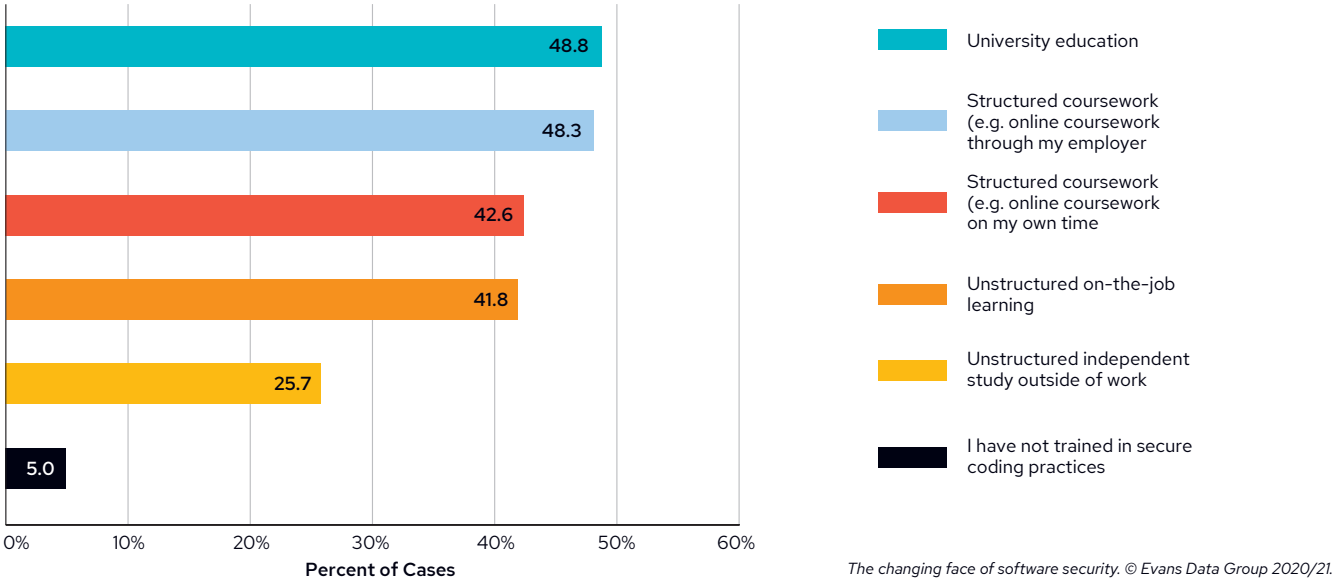
Current secure code training is letting developers down

Not only is training not standardized within organizations, but the training developers receive is also lacking in relevance and engagement. 28% of developers find the learning process challenging and 24% of them say it's simply boring. These top concerns are similar in nature demonstrating a need for more engaging and relevant training that makes the learning process more interesting, leading to better skills development and outcomes.

Where and how do developers learn software security?

Most developers say they learned software security in University, with over 86% of developers earning a bachelor's degree and 42% have a master's or doctoral degree.[3] Computer sciences curriculums, however, especially at the bachelor level are often focused on the fundamentals of development. And most university computer science courses do not teach secure coding at all.[4]

How did you acquire your familiarity with secure coding practices?



The changing face of software security. © Evans Data Group 2020/21.

Again this highlights inconsistencies between what developers claim to know about secure coding and their actual skill level. It also suggests that developers should seek out secure coding training beyond their university studies either through their employers or on their own time, as many do.

Regardless of training at University, the fact that both developers and managers acknowledge the benefits of continuous secure coding education means there is a need for ongoing training outside of academia. It is also important to continuously train because software vulnerabilities constantly evolve, so training must as well.

Companies are providing structured training, but unfortunately that usually comes in the form of online tutorials or coursework, as we can see from the image above. Informal training or 'unstructured training' is also often provided by more experienced employees, takes on the form of advice, and is often ad-hoc in nature.

Whilst 41% of developers turn to their peers to ask for help and 99% of them find this type of learning valuable to their careers, they also found coding challenges, external trainers and classes, and practical exercises like penetration of virtual sandboxes of even higher value.

Despite these various methods of training, developers still believe that secure coding is hard and software vulnerabilities continue to be a regular occurrence, indicating that current training is inadequate.

[3] Evans Data Corporation. (2020) Global Development Report: Volume 2.
[4] Amy DeMartine et al (2019). Show, Don't Tell, Your Developers How To Write Secure Code. Use Application Security Testing To Educate Your Developers. Forbes.

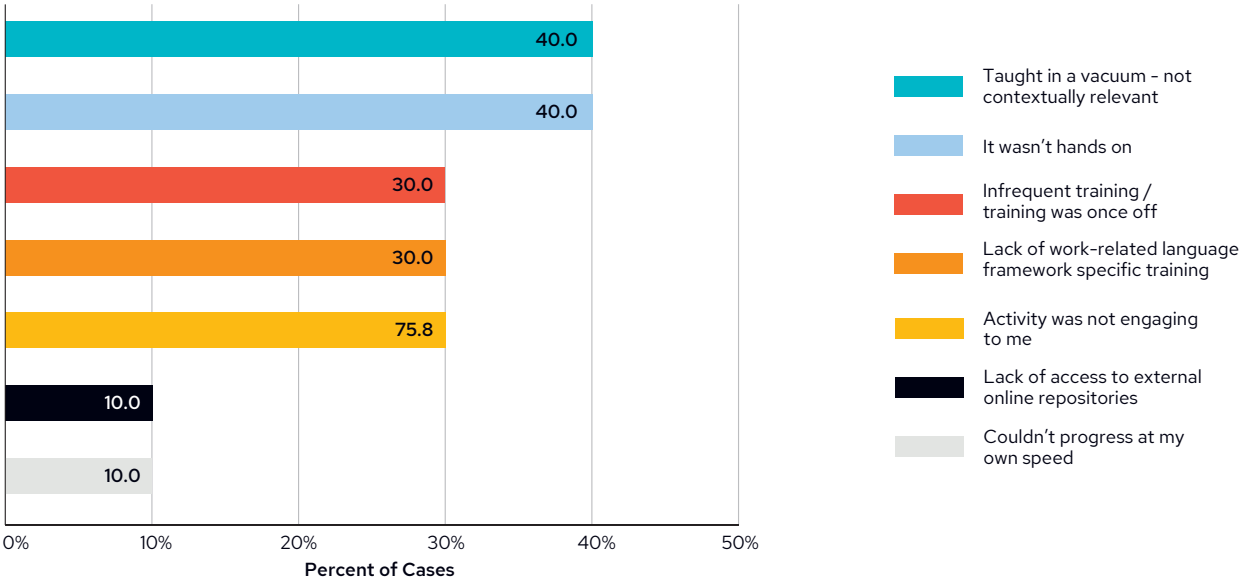
Developers want more structured, relevant, and hands-on training

It's clear that developers and managers see the value to be gained by robust secure coding training programs, but they also say that the current training methods available fall short.

When asked what kind of learning developers want, 75% of them responded that they prefer structured coursework, rating it as excellent or satisfying as compared to unstructured training. In fact beyond university, 69% of developers look for structured training programs for certification.

In response to what could be improved, 40% of respondents said that it was taught in a vacuum and another 40% said it wasn't hands-on enough (see image).

Since you feel your secure code training could be improved what was the main problem with it?



The changing face of software security. © Evans Data Group 2020/21.

This demonstrates that developers want engaging programs for up-skilling that are both hands-on and relevant to the work they do.

When questioned further, there are three main approaches to training that developers would like to see more of. These are:

1. Internal peer coaching / guidance (ie. 'Brown Bag' sessions, Sponsorship Meetups, conferences/ seminars) - 78%
2. Structured training (ie. in-house instructor led, external instructor led, formal coaching/ mentoring) - 73%
3. Hands-on (self paced) (ie. Sandbox practice environment, hackathons) - 62%

Developers tend to seek out informal training through their peers, but there is a need and desire to have more structured training available to them. This echoes other findings that show that developers learn more from structured coursework than unstructured learning.

In addition to providing structured methods for learning, companies must also meet the need for more engaging learning experiences that cover topics that are relevant to the work developers do.

The top five characteristics of a successful training program according to the survey are:

1. More practical training showing real work scenarios
2. Guided training focused on specific code vulnerabilities
3. Inclusion of more examples or use cases
4. Provides some concrete advantage to taking the training
5. Incorporates more team building exercises

66% say that more training in language-specific vulnerabilities is required, demonstrating the need for more relevant tutorials and coursework that specifically focus on them.

Additionally, developers require training on various compliance frameworks. The top four compliance frameworks that managers believe developers need more training in are:

1. Owasp Top 10 (65%)
2. NIST Security Framework (58%)
3. CIS Security Framework (52%)
4. PCI DSS (50%)

It's clear that organizations need to drive training programs entailing more structured training that is relevant to a developer's everyday work, including compliance frameworks. So how can companies ensure that training is engaging?

As we've already seen, developers tend to find secure code training boring, meaning that companies must assess the types of training activities they offer in order to increase engagement and therefore skills. Developers also benefit from peer to peer coaching and team building exercises. When considering both training methods and engagement, the preference from developers is to be practical and hands-on.

When asked what kinds of training drive the most engagement, the answers were:

1. Hands-on boot camps (96%)
2. Virtual sandboxes to infiltrate (93%)
3. In-house seminars (93%)

Lack of communication between stakeholders and management

When surveyed, management identifies the top three barriers to adoption as:

1. Lack of communication between stakeholders and management
2. Lack of a secure coding skill set among new hires
3. Inadequate training time/ resources

We've discussed the second two barriers, but the first still remains to be explored. Organizations recognize the importance of secure coding, but communication remains a huge roadblocker to adoption with 45% of respondents agreeing it's an issue.

Why communication is such a huge barrier is partially explained by the fact that development managers feel pressure from stakeholders (58%) to ensure their developers have adequate training to write secure code. But as we will learn later, they don't necessarily have responsibility for how that is done.

Developer motivators and concerns

Not only are development teams recognized for writing secure code, they are also held accountable for when vulnerabilities occur. When we look at the five main concerns in terms of secure coding from developers and managers we learn that their concerns are similar.

Developers are concerned about:

1. Including code that replicates previous vulnerabilities
2. Dealing with vulnerabilities introduced by co-workers
3. Meeting deadlines
4. Being accountable for code
5. Learning process is challenging

Development managers are concerned about:

1. Being accountable for code
2. Including code that replicates previous vulnerabilities
3. Learning process is challenging
4. Dealing with vulnerabilities introduced by co-workers
5. Engaging with developers on my team

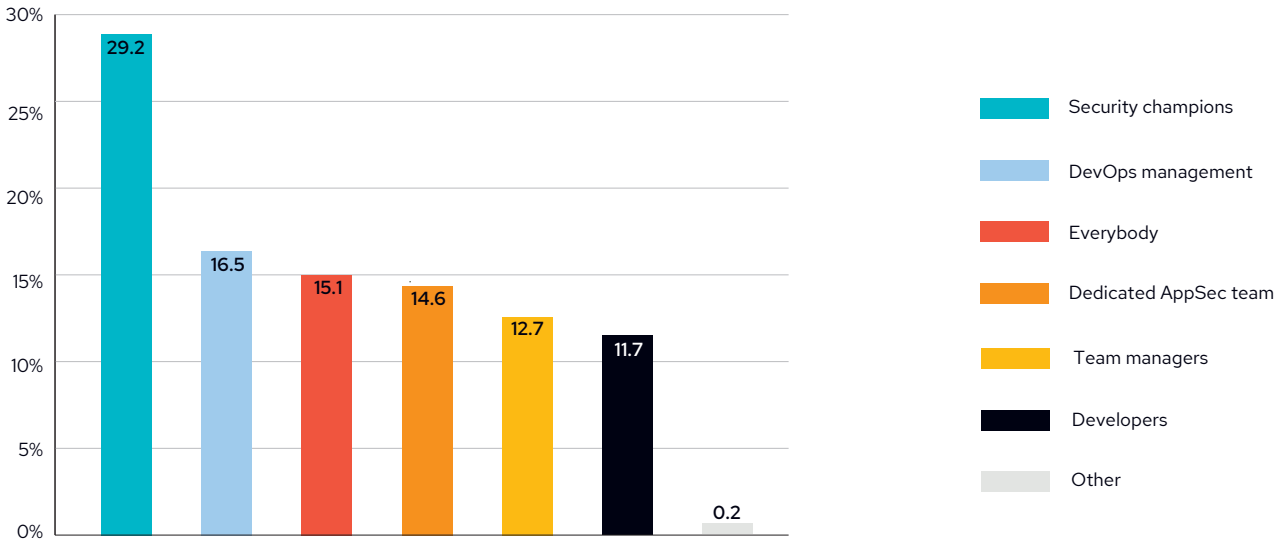
Both developers and managers are concerned about the presence of vulnerabilities whilst being accountable for code. This worry is understandable given the pressure to deliver secure code, under deadline, and potentially without the adequate skills to do so.

A shift in responsibility

This concern highlights the need for a shift in recognized responsibility when it comes to software security. Everyone in the engineering team should be responsible for secure code on some level, but as practices are being adopted, it's often the case that a few individuals become the "go-to-people".

The survey data showed that 29% of developers believe that individual security champions should be in charge of practicing or implementing security, whereas only 15% of them believe that dedicated application security teams should be in charge.

In your current role, who should be in charge of practicing or implementing application security and coding practices?



The changing face of software security. © Evans Data Group 2020/21.

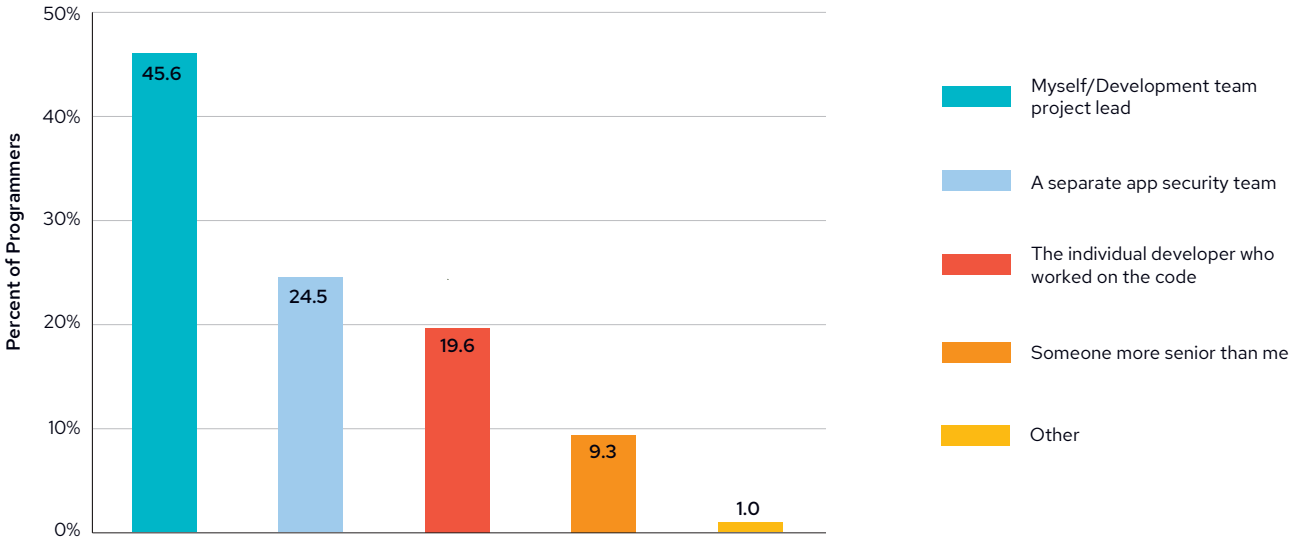
This points to a positive shift in mindset where developers and development teams themselves no longer believe that security is "someone else's problem" and that it should be implemented earlier on. It also reflects the fact that developers are increasingly held accountable for the security integrity of the code they deliver.

Despite there being a drive for developers to also become security champions the effort should still be led by management, to ensure consistency of training across developer teams and ensure that secure coding practices are implemented at the start of the SDLC.

For now the responsibility for security typically falls on AppSec although individual security champions take responsibility for security on their projects. But who holds the ultimate burden of responsibility and who should be responsible?

46% of development managers say that they themselves or their development leads should be the people who are ultimately responsible for application security.

While application security may be something everyone on the team wants, as a manager who do you think SHOULD ultimately be responsible for secure code?



The changing face of software security. © Evans Data Group 2020/21.

This suggests that although being accountable for secure coding is a major concern for development managers and developers alike, that concern is not stemming from the fact that they do not want the responsibility. Instead it points to the possibility that one of the reasons they find the accountability concerning is they do not have control over the training and implementation of security in the first place, resulting in security programs that do not suit their needs.

That said, development managers and developers agree that with the rise of Secure DevOps or (DevSecOps) cooperation between stakeholders is improving and increasing communication flows. When asked how employing secure code practices impacts developers, most agree that implementing secure code has increased communication within the team and within management. Managers also agree that when implementing secure code practices, they are able to spend more time managing people and that it helps increase the velocity of releases. As we learned earlier, communication between stakeholders is the main barrier for managers to implement secure code in the first place. And the root of this poor communication seems to be organizational misalignment and the uncertainty of where the true responsibility for implementing secure code lies.

Shipping quality code with confidence

To ship quality code with confidence, the process of implementing secure code practices needs to be transformed. We know from the research that better training leads to overall better quality code. 55% of developers revealed that good training increased their confidence in their coding techniques, while 53% said that good training allowed them to be more careful when debugging and testing their own code. And perhaps most importantly, 53% say they have become more mindful of security while they write their code.

From a management perspective, 47% of managers revealed that good training has allowed them to be more selective with tooling choices that provide more security, whilst 44% say that secure code training and techniques have helped to save time and speed up software releases.

Top three ways training has changed coding for developers

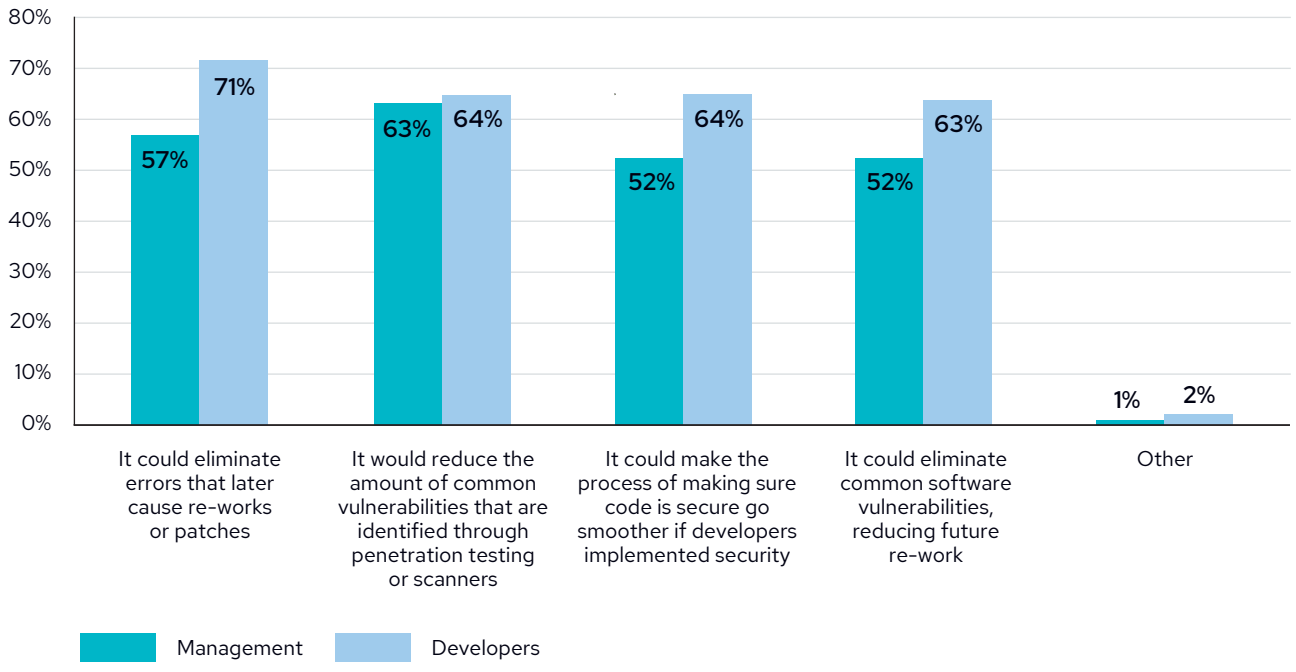
1. More confidence in coding techniques
2. More careful when debugging and testing their own code
3. More mindful of security as they write code

Top three ways training has changed coding for managers

1. Can select tools that provide more security in coding
2. Secure coding techniques save time and speed up software releases
3. More careful when debugging and testing their own code

This demonstrates that better secure code training can increase productivity in general. It has helped developers eliminate efforts that require re-work by minimizing recurring vulnerabilities and it helps speed up software releases (see image). Perhaps we can even be so bold to say that it has made secure coding less difficult.

In what way would it MOST improve productivity?



The changing face of software security. © Evans Data Group 2020/21.

Implementing secure coding into the SDLC and the rise of Devsecops

As developers and their managers integrate secure coding practices into their development lifecycles, the real-world impacts of their training can be measured. While all stages of the SDLC are positively impacted by the integration of secure coding, the most benefits are reaped during debug and test, coding, and application design. When it comes to deployment, 44% of developers say that deployment has become more productive since implementing secure coding into the SDLC.

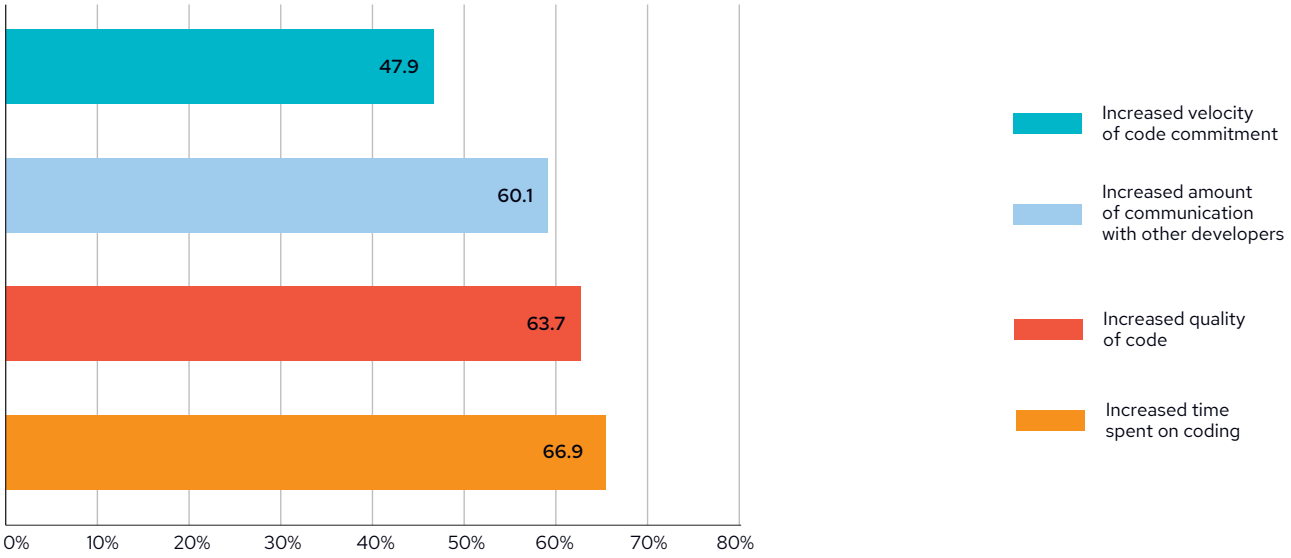
Implementing secure coding from the beginning of the SDLC also has a positive impact on the teams themselves. Clear communication is a key part of an effective security culture and when developers have a good understanding of what their managers expect in terms of secure code, and there is a trust between team members that they are collectively responsible for the quality of their code as a team, the quality of code ultimately increases.

When asked how employing secure code practices impacts developers, most of them agree that implementing secure code has increased communication within the team and within management. Managers also agree that when implementing secure code practices, they spend more time managing people which helps increase the velocity of releases. As we learned earlier, communication between stakeholders is the main barrier for managers to implement secure code in the first place.

Secure code is quality code

Better communication is not the only byproduct of implementing secure coding earlier in the SDLC. It also improves the overall code quality. Although developers responded that they spend more time coding after the implementation, 63% of them also agree that the quality of that code is increased (see image).

Developers: In your role as a developer, how has employing secure code practices impacted the following?



The changing face of software security. © Evans Data Group 2020/21.

Not only that, but we are seeing a shift in organizations to use different means to measure the quality of their work. In terms of how the success of security is measured within an organization there are again both reactive and proactive approaches.

Reactive approaches include:

- Measuring the seriousness of defects
- The elapsed time since the discovery of defects
- Scanner metrics
- Defect counts
- Time to resolve defects

Proactive approaches include:

- Measuring a developer's awareness of the OWASP Top 10 security risks
- Training developers for competency around application security
- The use of whitelisted code
- Ensuring compliance with regulatory requirements.

What we're seeing is that a growing proportion of development teams now rely on those preventative metrics and 90% of developers say they pay attention to those metrics. Not only that, but 67% of organizations now measure developer awareness of the OWASP Top 10.

This doesn't mean that development teams will stop using reactive measures of security success nor that they should. It is, however, a sign that development teams are beginning to think of security earlier in the development lifecycle and equate writing secure code with writing quality code. And when they do, the overall quality of code improves with it.

Conclusion

Common vulnerabilities continue to persist within the SDLC because reactive measures (such as scanners, tooling and pentesting) only find the problem, often after the application is in production.

Engineering teams must embrace a human-led approach to coding. That is, starting left to ensure that security is considered and implemented at the beginning of the SDLC by developers who are accountable for producing quality code.

In order to be successful, developers need to have the right skills. And whilst many developers believe they have been adequately trained, the fact that vulnerabilities continue to occur, and that developers and managers admit they find security to be difficult, tells us they are not. Supporting that, we know that both managers and developers are concerned about vulnerabilities in code and being accountable for that code.

Current training methods are deemed to be irrelevant to their daily work, and/or unengaging, leading developers to seek out unstructured, adhoc learning methods. Even though Managers do not possess the authority to have control over secure coding programs and training for their teams, they're held accountable for security and lack the jurisdiction and support to implement it properly.

But there's good news. The mindset around who should be responsible for secure coding is changing. The rise of DevSecOps combines security and development and makes way for secure coding to take its rightful place amongst dev teams meaning that there is a growing understanding that *quality code is synonymous with secure code*. If development teams are responsible for the implementation and training of secure coding, they can choose training that is relevant, engaging, and effective. Once that happens, writing secure code will be easy to implement at the start of the SDLC.

Recommendations

Start left

Rather than shift left, *start left.* Empower your developers with the knowledge and skills to bake security into the products they write, when they are writing them. This human-led defence will shift your organization's approach to secure coding from being reactive and uncertain; to preventative, productive and positive. Not only will you eliminate vulnerabilities to make software more secure, you'll spend less time and money on fixing bugs and ship quality code faster, with confidence.

Focus on skills

Don't confuse CBT and scanning tools with skills development. Developers need to improve their s*ecure coding skills,* in the work that they do and the languages they use, so that it is contextually relevant to the job they do.

Make it relevant and engaging

When it comes to extracurricular courses or on-the-job training, it is often overlooked that adults bring with them a certain level of experience and existing knowledge. When we tie this approach back to secure coding skills, it comes as no surprise that the dynamic, learn-by-doing method has long been preferred over the drudgery that is theory-based static learning.

A scaffolded learning approach can be structured so the content is hyper-relevant to the work your developers do every day. Consider learning pathways that provide the opportunity to reduce recurring vulnerabilities faster by upskilling developers specific to your organization's needs. It should also be hands-on and interactive to ensure the highest engagement possible. If developers have fun learning they'll learn more and start writing better quality code instinctively.

Assign security champions

Software security is slowly becoming an important measure for project success, however there are concerns and boundaries that impact alignment and adoption. Security champions through every stage of the development process can help to ensure that security requirements stay front of mind from inception to delivery and improve alignment and communication across teams. The benefits for teams wanting to ship quality code faster, with fewer bugs, minimal rework and lower costs are there to be gained.

About the study

The study titled *Shifting from reaction to prevention: The changing face of application security 2021* was conducted by Secure Code Warrior with Evans Data Group in August of 2020. The study sampled 400 software developers and decision-makers who managed software developers globally. The sampling margin of error at the aggregate level was 4.9%, measured at 95% confidence.

Sources

Amy DeMartine and Trevor Lyness with Stephanie Balaouras, John Rymer, Kate Pesa, Peggy Dostie (2019). *Show, Don't Tell, Your Developers How To Write Secure Code. Use Application Security Testing To Educate Your Developers.* Forbes.
https://www.forrester.com/report/Show+Dont+Tell+Your+Developers+How+To+Write+Secure+Code/-/E-RES144174

Evans Data Corporation. (2020). *Global Development Report: Volume 2.* Evans Data Corporation.
https://evansdata.com/reports/viewRelease.php?reportID=40

Laurence Bradford. (2018). *Why Developers Will Be More Marketable If They Know This Skill,* Forbes.
https://www.forbes.com/sites/laurencebradford/2018/02/28/why-developers-will-be-more-marketable-if-they-know-this-skill/?sh=24da9c5d57a3

Secure Code Warrior (2021). *Shifting from reaction to prevention: The changing face of application security.* 2020. Secure Code Warrior with Evans Data Corporation.

Synopsys (2016. *"Why isn't cybersecurity taught in schools?".* Synopsys Software Integrity Blog.
https://www.synopsys.com/blogs/software-security/cyber-security-education/

Secure Code Warrior

Smarter, faster secure coding

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, telco's, retail, governments and global technology companies across Europe, North America and Asia-Pacific. Learn more at www.securecodewarrior.com.


SECURE
CODE
WARRIOR