Whitepaper

PCI DSS 4.0 Unraveled

Seize the opportunity to nail security upskilling for developers

SECURE CODE WARRIOR

Executive summary

If you're an AppSec or development manager, you have likely noticed by now that most developers aren't exactly filled with joy at the prospect of compliance training. Even the word "compliance" has most people stifling a yawn. However, despite seeming at odds with any sort of creative environment or innovation, it is a vital exercise, and we need to do better to capture the hearts and minds of the development cohort.

Let's face facts: A typical developer experience with compliance exercises is usually a (vertical) read through some guidelines, or watching a presentation, before going back to coding features and focusing on creating software that customers will utilize and love. Everyone retains what they can in the moment (and generally, operates with best intentions), but compliance guidelines - especially those surrounding security best practices - are not typically written with developers as the target audience, and any required actions or key takeaways can be unclear. In that scenario, it's all too easy to just stay on-task with current objectives and deprioritize any new initiatives to meet compliance goals.

The thing is, secure software development is no longer a "nice to have" in any company; it should be front-of-mind in every organization. And if that organization is holding vast amounts of sensitive customer information, then it is ripe for the picking when it comes to costly cyberattacks.

> **Developers are the first to get hands-on with code, and as such, should be just as involved as the rest of the team in any security compliance measures.**

Now, fear not - this doesn't mean everyone has to become a slave to rigid, creativity-free development and software outcomes. **It is an opportunity for the developers and AppSec professionals to band together in pursuit of a higher standard of code.** Ever so slowly, the world is catching up to the fact that, to date, developers haven't exactly had the right tools at their disposal to make security a priority (and siloed security specialists cannot shoulder the responsibility alone). However, as the industry moves towards an AI-augmented, DevSecOps future with security as a shared responsibility, they can build the skill set needed to help stem the flow of recurring vulnerabilities.

The 2025 deadline to comply with PCI DSS 4.0 is the industry's biggest opportunity yet to elevate developers with the skills and tech stack required to positively impact software security from the ground up. These latest guidelines are the most operations-flexible to date, and the time is now to create a potent, custom security program that places developers in the driver's seat of meaningful change.

PCI DSS 4.0 Compliance: The regulations that finally consider developers and their challenges?

The PCI DSS guidelines cover online security compliance for card payment gateways - a service most of us use on a regular basis. These guidelines are globally applicable, and - in 2017, at least - they *have* detailed what developers must do to maintain standards in line with their mandates, alongside several compliance documents across aspects of eCommerce business. This does demonstrate historical attempts to include developers as part of the security compliance conversation, however, the success of the advice is entirely dependent on a commitment to right-fit tools, upskilling, and an internal security culture that nurtures developer-driven security outcomes.

> Data breaches are expensive, reputation-destroying risks that businesses can ill-afford, and with Cybersecurity Ventures predicting that global cybercrime costs will grow by 15 percent per year over the next three years, reaching $10.5 trillion USD annually by 2025, this is something that everyone in the software delivery process can help fight. These costs are up from $3 trillion USD in 2015, and have the potential to represent the greatest transfer of wealth in history.

The question is, even with guidelines for developers, do PCI DSS Security Standards move the needle in terms of vulnerability accountability and reduction?

Recently, the PCI Security Standards Council released version 4.0 of their software security guidelines as part of their PCI Software Security Framework. This update aims to bring software security best practices in line with modern software development. It's a fantastic initiative that acknowledges how this process has changed over time, requiring a rethink of the security standards that were set well before the majority of our lives became rapidly digitized.

This is clear evidence of our industry more closely engaging with the idea of adaptable guidelines - ones that evolve with our changing needs - as well as with the demands of a cybersecurity landscape that could very quickly spiral out of control if we continue to be lax in our secure development processes. Naturally, with the PCI Security Standards Council acting as a governing body within the banking and finance industry (as in, setting the security standards for the software we trust to protect all of our money, credit cards, online transactions, and at point-of-sale), they carry a lot of risk and have huge motivation to reduce it.

While these standards improve upon the previous version - especially with regard to championing a customized approach for more mature organizations - and go some way to plug the hole we have in rapid, innovative feature development that also prioritizes security as part of its overall quality assessment, it's going to take a serious commitment to developer education, verification of those new skills, and a security culture that places development teams at the heart of code-level vulnerability reduction.

Make no mistake: this is not a "bah, humbug!" to this initiative.

The fact is, these new security guidelines still require more prescriptive guidance if we want developers to meaningfully share responsibility for security.

We're still captivated by testing (and it's still happening too late)

One glaring issue with the PCI Security Standards Framework is its apparent dependence on testing. Of course, software must still be tested (and the scanning tool process still has its crucial place), but in many ways, we're still falling into the same trap and expecting a different result.

Who writes line after line of code to create the software we know, love, and trust? Software developers. Who has the unenviable position of testing the lion's share of this code at the finish line, either with scanning tools or manual code review? AppSec specialists.

What do these specialists continue to discover? The same bugs that have plagued us for decades. Relatively straightforward issues that we've known how to fix for years: SQL injection, cross-site scripting, session management weaknesses... it's like Groundhog Day for many security professionals. They spent their time finding and fixing code violations that developers themselves have had the power to fix for years, except that security has not been made a priority in their process – especially now, in the age of agile development where feature delivery at speed is king, and security is the elephant in the room that could halt release progress altogether.

There are a lot of guidelines relating to testing requirements, but some represent key changes to the established norm for many organizations:

- **PCI DSS Requirement 11.4.4** stipulates that every single vulnerability located in a pentest must be remediated, regardless of severity, followed by a second pentest to confirm successful fixes.
- The all-new **Requirement 11.6.1** details that entities must implement a manual or automated change-detection mechanism to check payment page content and HTTP headers at least weekly for evidence of any unauthorized modifications.
- These initiatives run parallel to a **requirement for developers to prioritize reducing the number of vulnerabilities and bugs in deployment**, by way of continually testing in-scope applications during the software development life cycle (SDLC) using static application security testing (SAST) and dynamic application security testing (DAST) tools.

Developers and AppSec professionals both have extremely important jobs to do, but they continue to get in each other's way. This situation only perpetuates a flawed SDLC, where developers with little security awareness operate in a negative security culture, producing insecure code, which then has to be scanned, assessed and fixed well after it was initially written. AppSec barely has time to fix the truly complex issues, because they're so caught up with the little recurring problems that could still spell disaster for a company if left unchecked.

> **We are wasting time, money and resources with an over-reliance on testing to be the catch-all for security weaknesses in code. And with massive data breaches every other day, this method is obviously not working optimally, if at all.**

We absolutely must work from the ground up: by getting the development team engaged with security best practices, empowering them with the knowledge to efficiently code securely, in addition to creating and maintaining a positive security culture in *every* workplace.

Is it a learning curve? Indeed, it is. Is it impossible? Definitely not. And it doesn't have to be boring drudgery. Agile, customizable training methods that appeal directly to the developers' problem-solving traits have already had immense success in forward-thinking organizations that are willing to try less conventional approaches to developer education. Some reduced key vulnerabilities by over 50%.

Are your developers prepared to deliver compliant software?

Developers sit as an integral - yet frequently underutilized - part of reaching a state of software security excellence, and this is especially relevant to more than just token PCI DSS compliance. It is crucial that developers understand the broader picture of PCI DSS 4.0 in terms of what they can control, and integrate as part of their default approach to a software build.

There are a lot of guidelines relating to testing requirements, but some represent key changes to the established norm for many organizations:

Authentication

A viable plan for access control has always been a key part of PCI compliance, however, version 4.0 kicks this up a notch in a way that will require careful implementation both internally and externally. Multi-Factor Authentication (MFA) will become standard, as will bolstered rules around password complexity and timeouts.

Encryption & Key Management

We operate in a world where we can access some of our most sensitive information via multiple access points, such as our online banking. With this high-value data at risk, encryption and strong cryptography practices are a must. Developers have to ensure they carry up-to-date knowledge on where information is transmitted, how users can access it, and even in the event of it ending up in the wrong hands, ensuring it is unreadable by threat actors.

Malicious Software

In the previous guidelines, security controls around software protection from malicious code were referred to as “antivirus software”, but this oversimplifies what should be a multi-layered approach shielding against far more than viruses alone. Anti-malware solutions must be applied wherever necessary, and continuous logging and monitoring are mandatory.

It is also vital that developers have learning pathways that cover identifying vulnerable components, especially with most codebases relying at least in part on third-party code.

Of course, you can always take the initiative to squeeze the most juice out of these guidelines as you can, and take them a step further in your organization. With the right (less generic) approach, you can not only stay compliant, but operate with an enviable level of security, quality, and hygiene as a part of your day-to-day development.

It all starts with adapting these guidelines for each role, and how it can be a meaningful exercise for everyone involved.

Hey, AppSec and Compliance teams: All developer training is not created equal!

Developers in large (or even small) organizations rarely have a whole lot of input into the training they receive on the job. In order to retain star employees, some companies do offer comprehensive programs, but these are still less common than they should be. A need for security training presents a great opportunity to not only kick-off organizational compliance without boring everyone to tears, but also start to warm up any frosty relationships with the development team.

In terms of PCI compliance, you can see that their guidelines are specific as to the outcomes they would expect from any software running payment gateways; among other objectives, they want applications hardened (vital for thwarting malicious code injection or tampering), least-privilege access controls and full-scale awareness of common vulnerabilities, at a minimum.

> All personnel working with cardholder data must be adequately trained, and for developers, that training can make or break their success in writing secure code from the start of their process.

The official PCI DSS 4.0 Guidelines document details some direct concepts around security awareness, developer needs, and appropriate training, like:

Guideline	Requirement	SCW's Recommended Action
PCI DSS 4.0 Requirement 6.2.1	Processes and mechanisms for developing and maintaining secure systems and software are defined and understood.	Developers must have a solid, consistent understanding of how they are keeping PCI data safe. E.g. Have authentication/access control parameters been defined? We can assist to address gaps in knowledge by providing precision training in the languages and frameworks that are actively in use. See more on our Learning Platform.
PCI DSS 4.0 Requirement 6.2.2	Software development personnel working on bespoke and custom software are trained at least once every 12 months as follows: • On software security relevant to their job function and development languages. • Including secure software design and secure coding techniques. • Including, if security testing tools are used, how to use the tools for detecting vulnerabilities in software.	Once every twelve months is nowhere near frequent enough to address core security issues and break the cycle of poor coding patterns. Training must be continuous, measured, and ideally, an established skills verification process should be in place to ensure training has been absorbed and put into practice. Learn more about our ready-made secure code training pathways for developers.
PCI DSS 4.0 Requirement 6.2.3	Bespoke and custom software is reviewed prior to being released into production or to customers, to identify and correct potential coding vulnerabilities, as follows: • Code reviews ensure code is developed according to secure coding guidelines. • Code reviews look for both existing and emerging software vulnerabilities. • Appropriate corrections are implemented prior to release.	A rigorous code review process must be put in place, and peers should review each other's code in accordance with established secure coding guidelines and compliance needs.
PCI DSS 4.0 Requirement 6.2.4	Software engineering techniques or other methods are defined and in use by software development personnel to prevent or mitigate common software attacks and related vulnerabilities in bespoke and custom software, including but not limited to the following: • Injection attacks, including SQL, LDAP, XPath, or other command, parameter, object, fault, or injection-type flaws. • Attacks on data and data structures, including attempts to manipulate buffers, pointers, input data, or shared data. • Attacks on cryptography usage, including attempts to exploit weak, insecure, or inappropriate cryptographic implementations, algorithms, cipher suites, or modes of operation. • Attacks on business logic, including attempts to abuse or bypass application features and functionalities through the manipulation of APIs, communication protocols and channels, clientside functionality, or other system/application functions and resources. This includes cross-site scripting (XSS) and cross-site request forgery (CSRF). • Attacks on access control mechanisms, including attempts to bypass or abuse identification, authentication, or authorization mechanisms, or attempts to exploit weaknesses in the implementation of such mechanisms. • Attacks via any "high-risk" vulnerabilities identified in the vulnerability identification process, as defined in Requirement 6.3.1.	Generic, infrequent training is no longer viable, and it won't have the desired impact on vulnerability reduction. Developers will perform best if they get familiar with navigating relevant vulnerabilities, and this should be conducted via agile learning methods that provide just-in-time, contextul microbursts of training. Learn more about our supported vulnerabilities.

Guideline	Requirement	SCW's Recommended Action
PCI DSS 4.0 Requirement 6.3.1	Security vulnerabilities are identified and managed as follows: • New security vulnerabilities are identified using industry-recognized sources for security vulnerability information, including alerts from international and national computer emergency response teams (CERTs). • Vulnerabilities are assigned a risk ranking based on industry best practices and consideration of potential impact. • Risk rankings identify, at a minimum, all vulnerabilities considered to be a high-risk or critical to the environment. • Vulnerabilities for bespoke and custom, and third-party software (for example operating systems and databases) are covered.	Many security programs suffer from a lack of ownership over certain tasks, especially in organizations with low security maturity. Ensure roles and responsibilities are defined, and that a process for categorizing, documenting and responding to each vulnerability is in place.
PCI DSS 4.0 Requirement 6.3.2	An inventory of bespoke and custom software, and third-party software components incorporated into bespoke and custom software is maintained to facilitate vulnerability and patch management.	Third-party libraries, dependencies and components are in frequent use in most organizations, and as a key source of exploitable vulnerabilities, these must be documented and continuously updated. Ensure you have a dedicated Software Bill of Materials (SBOM) program, with personnel identified to manage it closely.
PCI DSS 4.0 Requirement 6.3.3	All system components are protected from known vulnerabilities by installing applicable security patches/updates as follows: • Critical or high-security patches/updates (identified according to the risk ranking process at Requirement 6.3.1) are installed within one month of release. • All other applicable security patches/updates are installed within an appropriate time frame as determined by the entity (for example, within three months of release).	Ownership over patches and critical updates must be defined and prioritized, and those responsible must patch in a timely manner, with the most high-risk incidents given immediate priority.
PCI DSS 4.0 Requirement 6.4.2	For public-facing web applications, an automated technical solution is deployed that continually detects and prevents web-based attacks, with at least the following: • Is installed in front of public-facing web applications and is configured to detect and prevent web-based attacks. • Actively running and up to date as applicable. • Generating audit logs. • Configured to either block web-based attacks or generate an alert that is immediately investigated.	This requirement is likely to replace 6.4.1. eventually. It focuses on automated tools like Web Application Firewall. Ensure that it is configured correctly, up to date, and that someone is appointed to constantly review the logs. Any alerts should be investigated without delay.
PCI DSS 4.0 Requirement 6.4.3	All payment page scripts that are loaded and executed in the consumer's browser are managed as follows: • A method is implemented to confirm that each script is authorized. • A method is implemented to assure the integrity of each script. • An inventory of all scripts is maintained with written justification as to why each is necessary.	Successful implementation requires developers to employ established secure coding techniques to mitigate malicious scripts. We can help you seamlessly implement this recommendation with templates detailing the right secure code training pathways for developers. Learn more about secure code training pathways for developers.

This gives insight into the specific, in-depth training required to arm developers with the necessary skills, but still doesn't point to any particular type of educational solution as more effective than another, and still dictates that "at least once every twelve months" is sufficient, when training this infrequent is unlikely to have any meaningful impact.

> **As many of us have experienced from various workplace training and compliance initiatives, they can vary wildly in quality and future success. And when it comes to developers, so many secure coding training programs don't seem to cater to their needs, ways of working, or even their day-to-day jobs in any tangible capacity. In this scenario, not all training is created equal, and not all training is going to result in software that is more secure.**

This is, of course, the exact opposite of any organization's desired outcome. If you want to reduce the burden and stop common vulnerabilities from causing a potential disaster, you'll need to build a bridge.

Developer education should start with a foundational education in the OWASP Top 10, as well as any other vulnerabilities that are language-relevant and business-critical. This should be part of an ongoing program, with the view to continue building upon those skills and embedding security not just into software development from the start, but also into their mindset and approach to their role. In addition, roles and responsibilities must be abundantly clear to the development cohort and their managers. Security should be a shared responsibility, but it's only fair to document expectations and ensure they can be met properly.

Bridging the security skills gap with engineering managers on your side

Working directly with engineering managers to find a solution that is fit for purpose, while still being embraced by the people who actually have to do it, is a fast track to positive security outcomes. They will have more immediate insights into their own team, and can speak to any blockers that have previously made compliance training difficult (other than it not being a great experience, most of the time).

Classroom-based training, video-on-demand, and any once-off, tick-the-box exercises make staying current very difficult; these are static solutions that cannot keep pace with the ever-changing landscape that is the cybersecurity industry. Ultimately, the engineering manager will want to see value for the time commitment, and it's important to work with them and provide viable options that work to meet everyone's shared goal: code that is more secure and compliant.

> **So, what does good training look like? There is very little point in learning how to code securely via big, once-off hits of information. A bite-sized, gradual learning process using agile upskilling methods makes it far easier to remember and apply in context, and it absolutely must be in the languages and frameworks used every day.**

Generally, developers want to be challenged, and want to understand a purpose for their efforts; this starts with right-fit tools and training that encourages curiosity and learning by doing.

To comply with the PCI DSS best practices outlined above, depending on the solution chosen, managers might find themselves having to stitch together a patchwork of different courses to cover all languages and frameworks used across the business, and that's when things get very messy, not to mention difficult for AppSec and compliance teams to assess as making an impact on security and vulnerability reduction in software. Work together to find the right fit, instead of rushing into the wrong option chasing a fast result.

PCI DSS 4.0 also embraces the concept that there are many ways to achieve the same goal of airtight security best practices. This is true, but it seems best suited to organizations with advanced security maturity and leaves a lot of room for error, especially for those who have not been realistic in their assessment of their true internal security maturity. Ultimately, companies must be prepared to engage in security as a continuous, evolving process, not a once-off "set and forget" exercise. A strong security culture is a must, with an organization-wide commitment to security awareness.

Everyone should care about trust issues in the customer/ organization relationship

Apart from the stress and calamity the IT, development, and security departments face after a breach, the trust factor is a major element in the long-term success of a newer company, or the continued growth of an established one. The obvious thing you stand to lose is your job, should the company be faced with an economic downturn as a result of lost faith.

The PCI DSS regulations hold businesses accountable - and as above, ignoring these well-laid plans has enormous implications - but they're only as good as the security program that is put in place, and the people working within it. If you take them seriously, stay aware, and set an example for others, then you are setting yourself apart in a very positive way.

Awareness is everything

A failing security awareness program is going to make most attempts to stay PCI-compliant close to useless. Organization-wide security awareness forms the most critical part of the best practice guidelines; they even offer their own training modules on how this can be implemented in cross-functional roles, and what this looks like in businesses that are doing it right.

As we move towards DevSecOps as the current gold standard in secure software development - in which security as a shared responsibility is fundamental - businesses must spend the time, money, and effort to ensure everyone, including vendors and contractors, is security-aware and following best practices.

A security-aware developer is a compliant developer (and getting there doesn't have to be boring)

When it comes to becoming a "certified" compliant PCI DSS developer, there aren't a whole lot of obvious options. Why? Probably because it cannot be a "one-and-done" exercise.

The OWASP organization is one of the best on the planet when it comes to learning how to thwart common vulnerabilities, and their Top 10 is formally listed in the PCI DSS guidelines for developers. However, keeping security front of mind and honing skills takes time and continued effort. And nobody wants this to be uninspiring and a waste of effort.

A positive security culture is not a "nice to have" in an organization; if they're taking security seriously, then it needs to be part of the everyday running of the company.

Developers are on the front lines of battle when it comes to stopping vulnerabilities. Are they receiving the support, tools, and training to hold up their part of the security deal in PCI DSS compliance?

The truth is, the right training is more seamless; it shouldn't feel like a lecture, and it should be highly relevant to the work being done every day. And this kind of hands-on training is an upskilling opportunity - a career move that only has positives for developers who are serious about stopping vulnerabilities and working with the rest of the team to produce a higher standard of code.

About Secure Code Warrior

Secure code learning for today's developers

Secure Code Warrior gives your developers the skills to write secure code. Our learning platform is the most effective secure coding solution because it uses agile learning methods for developers to learn, apply, and retain software security principles. Over 600 enterprises trust Secure Code Warrior to implement agile learning security programs, deliver secure software rapidly, and create a culture of developer-driven security.

Request a demo

Try Secure Code Warrior for free

Find us on social: