The Six Pillars of DevSecOps: Collective Responsibility

Key Considerations for Developing a Security-Supportive Culture

CSA cloud security alliance®

SAFECode Software Assurance Forum for Excellence in Code Driving Security and Integrity

The permanent and official location for the DevSecOps Working Group is
https://cloudsecurityalliance.org/research/working-groups/devsecops

Acknowledgments

Lead Authors:

John Martin
Stacy Simpson
Ashleigh Buckingham
Altaz Valani
Melissa Riley

Contributors:

Dennis Bush
Glenn Leifheit
Steve Lipner
Mathew Lyon
Sam Sehgal
Souheil Moghnie
Xiping Song
Sean Heide

Industry Partners

The Cloud Security Alliance (CSA) is the world's leading organization dedicated to defining and raising awareness of best practices to help ensure a secure cloud computing environment. CSA harnesses the subject matter expertise of industry practitioners, associations, governments, and its corporate and individual members to offer cloud security-specific research, education, training, certification, events, and products.
https://cloudsecurityalliance.org/

The Software Assurance Forum for Excellence in Code (SAFECode) is a non-profit organization exclusively dedicated to increasing trust in information and communications technology products and services through the advancement of effective software assurance methods. SAFECode is a global, industry-led effort to identify and promote best practices for developing and delivering more secure and reliable software, hardware and services.
https://safecode.org/

Table of Contents

Introduction 6
Overview 6
Executive Support and Engagement 8
Program Design and Implementation 8
Bringing Champions to the Challenge 10
Reinforcing the Program through Security Awareness and Training 11
Program Sustainment and Measurement 13
Summary 14
Appendix I: Healthy Questions and Discussion Points 15
Appendix II: Further Reading 15

Introduction

The Cloud Security Alliance and SAFECode are both deeply committed to improving software security outcomes. The paper Six Pillars of DevSecOps, published August 2019, provides a high-level set of methods and successfully implemented solutions used by its authors to build software at speed and with minimal security-related bugs. Those six pillars are:

Pillar 1: Collective Responsibility
Pillar 2: Training and Process Integration
Pillar 3: Pragmatic Implementation
Pillar 4: Bridging Compliance and Development
Pillar 5: Automation
Pillar 6: Measure, Monitor, Report and Action

The successful solutions that underpin each of these pillars are the subjects of a much more detailed set of joint publications by the Cloud Security Alliance and SAFECode. This paper is the first of those follow-on publications.

Overview

DevSecOps integrates security principles, processes, and technologies into a continuous software development and IT operations culture that influences its practices and workflows. It brings together the traditionally siloed areas of development, infrastructure operations and information security under a common set of processes, procedures, and tooling automation that facilitates the development of secure software.

The rise in interest in DevSecOps is, in-part, driven by a cloud-first software development environment that features increasingly shortened product life cycles, more iterative development methods, and the growing integration of development and IT operations. Legacy secure development approaches often struggle to meet the demands of continuous development methods that are rapidly being adopted by those working in cloud environments. Integrating secure development practices into DevOps requires a more agile approach to security that encompasses increased automation of security processes, more thoughtful and pragmatic security implementation planning, clearer enumeration of risk and compliance requirements, and more actionable monitoring and measurement methods. Further, getting all of these elements to work together as part of a holistic DevSecOps approach requires a sense of collective security responsibility from everyone who touches the process.

> *A Security culture is the result of deliberate action aimed at instituting sustainable organizational change. It should not be viewed as an intangible attribute achieved through an ad hoc set of diverse activities.*

Given this increased interest in DevSecOps, the Cloud Security Alliance (CSA) and Software Assurance Forum for Excellence in Code (SAFECode) brought together a DevSecOps Working Group to identify and share best practices for the development and sustainment of a DevSecOps program. As the initial step, the Working Group identified and defined six focus areas critical to integrating DevSecOps into an organization, in accordance with the six pillars described in CSA's Reflexive Security Framework[1]. More detailed research and guidance across each of the six pillars of DevSecOps will be revisited and established over time in order to maintain industry specific standards. This paper is part of a planned series and will focus on the area that is arguably the foundation for all others – collective responsibility.

Fostering a sense of collective security responsibility is not only an essential element of driving security into a DevOps environment, but it is also one of the most challenging. It requires cultivating a change to the organization's mindset, its ideas and its customs and behaviors regarding software security. In this paper, we refer to this effort as building a security-supportive culture. Some hallmarks of that culture include the following characteristics:

- A Security-by-Design mindset is one where security is considered and addressed at every point in the software development and operations lifecycles. Security is not treated as an afterthought or simply as something to be dealt with by audit findings.
- A sense that everyone – from development to IT operations to senior management – has a role to play in ensuring software security and that they serve as the organization's first line of defense in navigating today's complex threat environment.
- A focus on individual accountability and trust. Such an approach enables greater autonomy and agility, making available the necessary security information and tools to allow for informed decentralised actions and decisions. This is a departure from traditionally restrictive, heavily manual and siloed security processes.
- A clear recognition that security is not separate and distinct from business objectives, and as a result, that there exists a recognizable and clear alignment between positive security behaviors and incentives, and a belief that team members will be supported in their security efforts.

While much has been written on the need to nurture a security-supportive culture, it remains one of the most consistently cited challenges of DevSecOps execution. Culture is something most often described as a critical but intangible element of an organization. Unfortunately, this may lead to a rather ad hoc approach to fostering cultural change. In software security, this often takes the form of the occasional hackathon or bug bash, or perhaps an annual training session on the value of software security practices. This is not to say these activities are not valuable, but rather that their impact is limited if they are not presented in the context of a team's objective, not reinforced, or do not include the right audience. Introducing security at the beginning of a cycle with the proper knowledge and training can help avoid the need for these on-the-spot sessions.

> *Company X recently focused on a developer training initiative to help create a culture of security. The developers were excited initially at the opportunity to attend the security courses and bring that new knowledge back to their work. However, without regular reinforcement and measurement the focus quickly returned to dealing with more immediate pressures. Despite trying to do the right thing, the investment in learning had little impact in the long term.*

[1] The Six Pillars of DevSecOps: Achieving Reflexive Security through the Integration of Security, Development and Operations https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/

A Security culture is the result of deliberate action aimed at instituting sustainable organizational change. It should not be viewed as an intangible attribute achieved through an ad hoc set of diverse activities. This is not to say that there is one right way to foster a security-supportive culture. In truth, most organizations use a variety of methods to create the environment they need for their software security program to succeed. However, the most successful organizations view security culture development as a comprehensive business program with a clear vision, strategy and set of tactics with clear business value that can be measured.

Building a security-supportive culture is a complex task; for our purpose we will focus on describing the common practices shared by organizations that have taken a program-level approach to security culture development. We break these practices down to three key areas:

- Executive Support and Engagement
- Program Design and Implementation
- Program Sustainment and Measurement

Executive Support and Engagement

The methods used to create and sustain a security-supportive culture vary widely, but one fundamental building block almost all successful organizations share is a supportive and engaged leadership team. Management buy-in is required not only to ensure an adequate investment of time and resources into DevSecOps, but also to ensure the on-the-ground support for these efforts. Employees can recognize the difference between token or grudging executive support for a security program and serious commitment, and their recognition is inevitably reflected in the way they execute their part of the program.

Getting management buy-in for the development of a security culture is often done by presenting a comprehensive, results-driven program rather than as an ad hoc set of activities. Designing a program-level approach requires, at a minimum, the development of a business case strongly aligned to the business strategy, a detailed implementation plan, and key program metrics. Participation in organizations like CSA and SAFECode may also help shed light on how similar organizations are treating secure culture development, which can also support the business case.

Program Design and Implementation

Building a security culture should be viewed as an important and sustained program-level effort rather than a simple collection of ad hoc activities. This requires careful planning and thoughtful deployment for new initiatives aimed at fostering a sense of collective security responsibility. Common considerations for those looking to apply a program management approach are covered here.

When a business is designing a program to foster a security-supportive culture, the task is to bring a security mindset and sense of collective responsibility to an existing organizational culture. This requires security leaders to consider the existing culture and business practices of their organizations and development teams in program design. One useful exercise is to study past examples of requirements or process change to try to identify methods of communication and socialization that were effective and can be applied to secure development efforts.

For example, security leaders might consider how the Finance and Accounting team worked together with the development team to facilitate and celebrate the roll-out of its new invoicing and expense reporting system companywide, how the initial Threat Model contributes to a simplified product design and lowered cost and risk, or why the Product Management's new documentation requirements were so quickly accepted despite the increased workload they created? Some other common considerations include the following questions:

- Do teams tend to respond better to corporate mandates from senior executives or do they tend to embrace a more grassroots approach from within engineering?
- What types of incentives tend to work well for fostering behavioral changes?
- Who are the key influencers in the organization and is there a way for them to help in this effort?
- What is the relationship between a culture of collective security responsibility and the goals and vision for the engineering teams as well the company at large?
- In what ways do security requirements and practices clash with development processes and how can those conflicts of purpose be minimized?
- How can/should mid- and upper-management be involved?
- Are the existing culture and associated processes still aligned to business aims, or are they now outdated? If outdated, is there still value/can improvements be made? Internal talking points your team can use include these considerations:
    - Predefined structures and processes are often internally focused, serving the structure rather than the wider business need (case in point - ineffective change control processes, where neither the management nor the engineers are happy with the process and don't see value in it, yet the process remains unchanged and begins to act as a blocker without adding much value).
    - In the siloed model, communication across hierarchy can be slow and ineffective. Information flow through multiple layers top to bottom and vice versa is slow. Can things be simplified and/or complemented with initiatives such as shared security responsibility (empowerment), the use of automation for both control and metrics, and the clear reporting of meaningful metrics to management (data driven business decisions)?
    - Policies, rules, and procedures become barriers to strategic speed and decision-making. We see the movement to decentralized control with automated security "buffers" and individual accountability, supported by better security training and initiatives such as the Security Champions.
    - People cling to their habits and fear loss of their power bases and stature. Culture and progress may be stifled if we repeatedly return to the same small number of trusted people to lead key initiatives rather than taking input from a wider pool of stakeholders (again, this speaks to empowerment and better communication/collaboration).
    - Is there a legitimate business need for rapid innovation and disruption, and a higher risk appetite or, conversely, is the business need more risk-adverse or compliance-driven?

Security teams are often too far removed from the development process to be seen as partners; They're seen as stepping in late in the cycle to give directives or request changes, but without adequate considerations for the realities of the engineering environment. In fact, one of the toughest lessons learned by early pioneers in software security was that the fastest way to fail was to ignore what the developers needed to succeed. This challenge could be more acute in a DevOps environment given the high value that teams place on automation and speed of iteration.

At first glance, a DevSecOps approach seems to fully integrate security into the development and operations teams, eliminating the need for a separate security group. While this may be the desired future state for many, in reality most organizations have found the need to maintain a dedicated software security team to ensure the needed expertise and focus on security issues. In practice, the biggest organizational challenge is how to integrate the work of the security, development and operations teams in a way that supports the hallmarks of the desired security culture.

In earlier software development models there was a natural tension between security and development people. Security people often viewed software management as being morally ambiguous, "publish now and fix it later" while the business goals were to simply realize the income stream needed to build the next, better, incremental software release. DevSecOps allows the merging of these two mindsets. Security becomes an integral part of the product, and the security and privacy requirements of the users become clear functional requirements.

This does not mean that an organization using a separate security team is doomed to fail. Nor does it necessarily mean that fully embedding security within development is the right approach. In fact, there are successful organizations with both models. The important takeaway is to consider the ways in which staffing plans and the surrounding organizational chart supports or impedes the cultural development so that any challenges that arise from either approach can be mitigated.

Furthermore, it doesn't need to be an all or nothing discussion. In fact, there is a growing momentum in the use of a hybrid approach—using Security Champions to bridge the gap between security and development.

Bringing Champions to the Challenge

Security Champions are members of the development team that have been empowered to stay consistently involved in helping to guide and execute security activities on a day-to-day basis. Unlike security team experts with limited or restricted availability, Security Champions are reliably available to support DevSecOps execution and security activities at the development team level. Many organizations have found Security Champions programs to be the key to scaling their software security efforts. Further, Security Champions have a direct impact on the culture of the organization. In fact, they often serve on the front line of any effort to socialize a collective security responsibility and are empowered to escalate problems to the security team and management when needed.

When SAFECode undertook an effort to compare successful Security Champions programs among its members, it found they shared a few common attributes[2].

[2] Software Security Takes a Champion A Short Guide on Building and Sustaining a Successful Security Champions Program http://safecode.org/wp-content/uploads/2019/02/Security-Champions-2019-.pdf

First, the role of Security Champion should be treated as a primary responsibility, ideally a full-time or near full-time job. Simply tagging an already-overloaded software developer as the new resident security expert is not sufficient. Security Champions need to partner closely with the broader security team and be able to prioritize security issues based on risk. In this way, they are a context-aware security driver in the engineering team and an interface to the security team.

Second, while Security Champions initially may need to serve in a tactical security position on the engineering team, the goal should be to evolve their role as the team's security knowledge grows. Rather than doing the security work for the developers, Security Champions should serve as mentors that train developers so that they can tackle security issues on their own, reaching out for additional clarification or help when needed. Ideally, the Security Champion would work closely with the Chief Architect and Stakeholders, own the threat and privacy models, and be the primary focal for resulting activities (identifying security-related features, component selection, etc.). In the absence of this mindset, it will not only be difficult to scale their impact, but security may continue to be viewed as someone else's job (in this case the Security Champion's) instead of as the collective responsibility of every developer.

And third, just finding a Security Champion and telling them to go fix security will not work. To be successful, Security Champions need the support of a formalized, management-backed program with clear organizational and professional goals, as well as a supportive organizational ecosystem in which to work. In this way, the success of a Security Champions program is closely tied to the efforts of a security culture development program, and as a result many organizations should treat them as a single effort. At the very least, Security Champions and Culture Development programs should be considered together if both are being pursued.

Reinforce the Program through Security Awareness and Training

Training is an essential element of any successful DevSecOps initiative. In fact, it has long been a focus in software security given that secure development practices are seldom included in software engineering education. The CSA DevSecOps Working Group identified training as an essential pillar in DevSecOps planning and is working on developing more detailed guidance to be shared later. SAFECode also addressed the importance of training in the latest edition of its *Fundamental Practices for Secure Software Development* paper[3], stating that "the entire organization should be made aware of the importance of security, and more detailed technical training must be provided to development teams that clearly articulates the specific expectations of individuals and teams."

While the need to provide detailed technical training to those with direct responsibility for implementing secure development methods is fairly obvious, many organizations may not be reaching a broad enough audience in their awareness training efforts. For DevSecOps to be successful, the entire team

[3] Fundamental Practices for Secure Software Development: Essential Elements of a Secure Development Lifecycle Program https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf

– Architects/Designers, Technical Writers, Program/Project Managers, Development Engineers, Service and Quality Assurance (QA) Engineers, IT Operations, etc. –should all have a solid base of foundational security knowledge. Creating awareness of both the importance of security and the specific role each team plays in security is absolutely essential to building a security supportive culture.

Consider other forms of outreach such as brown bag lunch-and-learn sessions, scheduled open hours with security experts, and security mentoring programs. Gamification of security is also gaining popularity as a way to make learning about security more hands on and engaging and can be done by adding periodic hackathons to the outreach mix.

It is understood that there is limited time for training sessions and outreach activities, so organizers be sure to include a way to capture participation numbers and collect feedback from participants and outreach leaders. These assessments will help identify the most popular and impactful outreach activities so that they can be replicated, while less useful ones can be abandoned.

Also, consider how other existing team and peer-oriented activities can be used to further build security awareness and develop security expertise. Established development approaches like peer programming and pairing can be used for security code reviews. If bug bashes are used periodically, there may be an opportunity to invite an additional team to participate that may not typically join such an activity.

Threat modeling should already be conducted as a team activity inclusive of a number of roles. It is important to ensure that participation in these sessions remains high and is not sacrificed due to scheduling challenges. Threat modeling sessions can also be strategically expanded to include others who may not traditionally be invited in order to grow security interest. It is worth noting that those who enjoy and do well in threat modeling sessions often make good candidates for Security Champions and other security outreach roles.

Finally, consider whether additional skills beyond security awareness and techniques can help improve the impact of security programs. For instance, providing a cohort of identified security influencers or Security Champions with leadership training may increase their effectiveness within their teams. Offering opportunities for presentation training may encourage others to step forward and volunteer to lead internal brown bag sessions.

While the exact mix of training programs and outreach activities will vary from organization to organization, and perhaps even team to team, it is important to take a holistic and comprehensive approach to these initiatives in the context of a culture development program. This does not necessarily require a centralized approach to planning training and outreach programs. In fact, sometimes outreach efforts may be most effective when a need is identified and acted upon at the team level. However, it is important to have an understanding of what is being done across the organization. This not only helps ensure that training and outreach remain relevant and impactful, but will also identify gaps in execution, optimize resources, inform other areas of effort, and present additional opportunities to nurture the security culture.

Program Sustainment and Measurement

Developing a monitoring and reporting strategy is an integral element of DevSecOps planning. Actionable metrics enable DevSecOps initiatives by measuring progress and detecting failures in a timely manner. Measurement is such an important piece of DevSecOps that it is also one of the six key pillars[4] of a DevSecOps strategy identified by the CSA-SAFECode DevSecOps Working Group and is the subject of a forthcoming guidance. As such, this paper won't attempt to cover the full scope of DevSecOps measurement and reporting practices. However, it will address two key culture-related measurement issues:

- How what you measure drives culture
- The challenge of measuring something as intangible as culture

First, it must be recognized that what a company measures drives how it behaves. In this way, the metrics an organization selects to monitor the success of its DevSecOps program will have a direct impact on its security culture. Considering that many organizations tie incentives directly to program performance metrics, the link between measurement and behavior becomes even more apparent.

For example, if time-to-production is the only metric that matters, then there is little to no incentive to slow things down to address identified risk. Or if the most valued metrics focus on feature-delivery rather than quality, then that is likely to drive the organization to produce software swamped in future technical debt and a culture that defers responsibility. Conversely, if meaningful quality metrics are prioritized, then there is a much better chance of creating a culture rich in product ownership and significantly decreased levels of deferred debt.

> *If meaningful quality metrics are prioritized, then there is a much better chance of creating a culture rich in product ownership.*

When Collective Responsibility is properly implemented, all security exceptions must have a level of clear accountability at the executive level. Security metrics are no longer just for measuring the performance/effectiveness of a security program. Given that security is embedded in DevSecOps both culturally and technologically, everyone participates in instrumentation and collection of security metrics and collectively owns and tracks the security performance.

In addition, organizational culture is never static; it will change as the organization evolves over time, and what worked when the DevSecOps was initially rolled out may not be as effective as the program matures. Security and development practices are also constantly evolving, as is the tooling available to support them, which in turn impacts the environment in which DevSecOps operates and creates cultural change. Just as there is no one single method of building a security supportive culture, what worked today may not be the same thing that works tomorrow.

[4] The Six Pillars of DevSecOps: Achieving Reflexive Security through the Integration of Security, Development and Operations https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/

It is imperative to have a well-designed approach in measuring program effectiveness, identifying points of failure, and ensuring a process for enabling any change required for continuous improvement. Having a strong measurement program in place will help guide any needed changes and allow an organization to sustain and build upon its initial success. As discussed earlier in this paper, maintaining executive buy-in is an essential element of long-term program sustainment. The ability to measure the impact of the culture development programs will enable the security team to make a stronger business case to senior leaders to continue the organization's investment in these activities.

There is no doubt that measuring something as intangible as culture presents challenges. The specific metrics used will depend on the organization's distinct goals. For example, a cultural program that focuses on ensuring management support of engineering efforts may measure the number of risk exceptions requested before and after program roll-out. If it is measuring developer interest in embracing secure engineering practices, a social measure of developer interest in voluntary security activities such as training and bug bashes would be effective.

Summary

DevSecOps is the natural evolution of cloud-native software development. When executed properly, it delivers products to customers faster by taking advantage of automation, more iterative development methods, and the growing integration of development and IT operations.

This paper identified methods for creating and maintaining executive support and engagement, building an inclusive cultural program based on our cumulative experience, creating deep engagement through Security Champions, and how to use metrics to sustain, build, and help evolve the program.

In closing, we'd like to share a few observations based on our experience:

> - DevOps without solid security built in only increases technical debt. Putting the Sec into DevSecOps must be done thoughtfully.
> - All successful DevSecOps programs share a collective sense of security responsibility from everyone who touches the process. This may be the most vital and most challenging aspect of DevSecOps implementation.
> - Allow leaders to lead. Give them the tools to do so.
> - In order to allow DevSecOps to work, leadership must be willing to allow change. There can be no backlash for structured creativity.
> - Culture will evolve. Just as prior actions created the current culture, even if not intentionally, a set of different actions can change the culture.
> - Since culture is constantly changing, there must always be a plan in place for sustaining positive impact and measuring that impact over time in the spirit of continuous improvement.

Appendix I: Healthy Questions and Discussion Points

Questions to ask when assessing your software security program's organization:

- Do you currently use Threat Modeling to help build functional requirements?
- Do you use functional security requirements to guide software component selection?
- Are the causes for discovered vulnerabilities fed back into the design and development process?
- Do product security incident response personnel (SIRT or Red Team) get involved in security design, process, and training either directly or through a Security Champion?
- Does your executive commitment to a development program's security issues impact product 'build' operations?
- Should a Security Champion own the Threat Model(s) and Privacy Model(s)?
- Should your product teams have Security Champions in each group or can that be a shared resource?
- Does individual and team access to security knowledge allow them to work independently of the Security Champion?
- Should unfixed security bugs (from a scan tool output) that don't have associated exceptions be approved (or approvable)?
- At what point should approved exceptions from current version or prior versions become unacceptable?
- Are you able to build and reuse a library of reference? As well as access to Threat and Privacy models?

Appendix II: Further Reading

A discussion of security culture requires a multi-disciplinary understanding of software and IT security, and this paper builds upon many of the concepts presented in the following works. These papers are useful reading for anyone interested in learning more about DevSecOps, security culture, and managing a software security program.

Fundamental Practices for Secure Software Development Essential Elements of a Secure Development Lifecycle Program: https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf

The Six Pillars of DevSecOps: Achieving Reflexive Security through the Integration of Security, Development and Operations: https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/

Information Security Management Through Reflexive Security: Six Pillars in the Integration of Security, Development and Operations: https://cloudsecurityalliance.org/artifacts/information-security-management-through-reflexive-security/

Software Security Takes a Champion A Short Guide on Building and Sustaining a Successful Security Champions Program: http://safecode.org/wp-content/uploads/2019/02/Security-Champions-2019-.pdf