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

# Software Security Takes a Champion

A Short Guide on Building and Sustaining a Successful Security Champions Program

January 2019

# Table of Contents

# Introduction

At SAFECode, we are always looking for common themes among our members that lead to successful software security outcomes. We've consistently found that while there may not be one single recipe for a successful product security program, the most tried and true recipes do share many common ingredients. One of those ingredients is the use of Security Champions (SCs).

In this short guidebook, SAFECode members share information on how to build and sustain a successful SC program — based on their experiences doing just that. Each chapter originally appeared as an article on the SAFECode blog and are captured in this guide for easy reference. These insights should be of interest to anyone working to build a more security-supportive culture within their development organizations – whether they already have an established SC program, are considering implementing one, or just hearing the idea for the first time. The authors cover questions such as:

- What does it mean to be a SC?
- Do (and can) SCs replace Security Teams?
- How does one identify potential SC candidates?
- Is SC a new career path?
- How does a SC program Strategy tie with a development organization's existing security efforts?
- How should an organization go about building a SC program Strategy?
- What about SC program rollout guidelines?
- How can a SC program be sustained?
- How do you measure SC program effectiveness?

## Meet our Champion Contributors:

**Vishal Asthana, CISSP** is on a 'Work Treadmill Break' debating between running a marathon, keeping Garfield's essence going, becoming a stand-up comic, or coming back to senses and taking up the next challenge in the software security space. As a professional, he is a seasoned practitioner with 16 years of combined work experience in various information security domains, the last 10 years being in software security. During this time, he has become skilled and passionate (read: crazy!) about assisting large enterprises with practical methods for addressing the 'bad cop' image their security teams often struggle with. Contribution to this blog series is the latest by-product of that effect. Specifics about his professional journey are available for 'spare time' reading on LinkedIn here: https://www.linkedin.com/in/vishalasthana/

**Kristian Beckers** grew up in Hamburg, Germany and received a PhD in Secure Software Engineering from the University Duisburg-Essen. He currently works as a Security Lifecycle Consultant at SIEMENS and focuses on secure agile and DevSecOps, as well as security training topics. Prior to that he worked for the Technical University Munich, Fraunhofer ISST, and NEC Network Research in Heidelberg. He is passionate about research, social learning and baking. Follow him on LinkedIn: https://www.linkedin.com/in/kristian-beckers-90439820/

**Manuel Ifland** is a Product & Solution Security Manager at Siemens and is a member of the Board of Directors of SAFECode. He is passionate about increasing the security of critical infrastructure and has been conducting many risk analysis workshops, security assessments, and penetration tests in various technological areas. Follow him on Twitter @maifland and on LinkedIn: https://www.linkedin.com/in/manuel-ifland-184862119

**John Martin, CISSP, CISM**, is Boeing's Security Program Manager with responsibilities ranging from DevSecOps to Commercial Software Security. His career spans the years between Blue-Box MF generators, through the era of automated hacks, and into our modern age of industrialized paranoia. Leading the SAFECode 'Buyer's Guide' effort, he is a frequent speaker on the topic of commercial software security. John was named by SANS as one of the 10 Difference Makers in Cyber-Security for 2016. Follow him on LinkedIn: https://www.linkedin.com/in/johnmartin-public/

**Nick Ozmore, CISSP,** has been involved with security related roles for over 20 years, the last six of which have been focused on application security. Currently in Product Security at Veracode, he is responsible for driving SDLC adoption and activities for Veracode's own products and services. Previously he was in a similar role within EMC's Product Security Office, working with EMC and RSA products. He believes that Product Security teams should maintain a positive consultative partnership with Development and work to build security into the organization's culture. At both Veracode and EMC he has partnered closely with Security Champions to increase the security of development deliverables and bring awareness to residual risk to facilitate informed decision making. Follow him on LinkedIn: https://www.linkedin.com/in/nickozmore/

**Altaz Valani** is the Director of Research at Security Compass and manages the overall research vision and team. Prior to joining Security Compass, he was a Senior Research Director and Executive Advisor at Info-Tech Research Group as well as a Senior Manager at KPMG. Altaz sits on the SAFECode Technical Leadership Council and several IEEE Working Groups where DevSecOps, Security, and Privacy challenges are being tackled at the international standards level.

**Tania Ward** grew up in Portrush, Northern Ireland and graduated from the University of Wales, Aberystwyth. She currently works as a Program Manager driving the security training and champion program across Dell. Previously, she worked in the Dell Product Security Incident Response Team. She is an EMT, ski patroller and believes "there is no wealth like knowledge and no poverty like ignorance." Follow her on LinkedIn: www.linkedin.com/in/taniacorrieward

**Izar Tarandach** is a Lead Security Architect with Autodesk Inc., currently located in Eugene, Oregon. Being an astronaut was out of this world and neurosurgery was too messy, so the next logical choice was a career in Information Security. Main interests are threat modeling, improving SDLC processes, security training, automating things and making stuff do things it is not really supposed to. Follow him on Twitter at @izar_t or on LinkedIn.

.

## Building Secure Software: It Takes a Champion

*By: Vishal Asthana, Security Compass (former); Manuel Ifland, Siemens; John Martin, Boeing; Altaz Valani, Security Compass; Tania Ward, Dell; Nick Ozmore, Veracode; Kristian Beckers, Siemens; Izar Tarandach, Autodesk*

> *"We have to go to production tomorrow – customer pressure, non-negotiable. Really wanted to adhere to this Application Risk Assessment thing in entirety, can't though. Sorry, will be needing a risk exception this time.*
>
> *Our Product Security Head has to comply with such last-minute exception requests all the time! I wish this mindset could be changed someday. Sigh."*

Familiar pain?

Organizations and their development teams often struggle with scaling their Secure Development Lifecycle (SDL) efforts. This is typically due to one or more reasons below:

- **Staffing limitations:** Centralized security teams (especially in large- or medium-sized organizations) often have to operate with only a handful of product security staff. This means the security team usually does not have the resources to support every single product or application team across the enterprise.
- **Skill Shortage:** Development teams willing to support security activities have to depend on security teams for security guidance because organizations hiring developers/engineers typically do not differentiate candidates based on security skills or background. Software security is unfortunately not yet commonly part of software engineering university education or training, so developers may lack a deep understanding of both internal and customer-related security (and regulatory) needs.
- **Diversity of development processes:** Organizations with diverse development processes and product lines may face the challenge of translating the SDL requirements into each product team's specific development process and terminology. Applying the corporate requirements to the local teams' processes requires knowledge at the local level.
- **"Bad Cop" perception:** Security teams are commonly viewed as bad cops, waiting to handout security violation tickets. This sometimes adversarial relationship leads to frequent resistance from development teams to give AppSec a seat on the roadmap discussion table and encourages a culture of AppSec risk exception to push releases through.
- **Prioritization Challenges:** Security development activities are frequently prioritized lower than functional development activities owing to the customer value-driven perspective. As there is a lack of established risk-to-value calculation for security development activities, prioritization is cumbersome or missing, and often the most technical person is chosen to handle that calculation even though they are not equipped to do so.

To build security into an organization's DNA, a company must create a security culture from top to bottom. SAFECode Chairman Eric Baize talked about this in his recent keynote[1] at OWASP AppSec California 2018. To paraphrase Eric, *'Successful implementation of software security at scale has led to an emerging need of bringing a cultural change across development organizations, software developers,*

[1] https://safecode.org/flip-script-software-security-changing-software-culture-code-scale-secure-development/

*and technology users. Development organizations play a pivotal part in serving this need and should plan on creating an ecosystem that makes security pervasive for developers. They could do so through a multi-year program aimed at driving multi-tiered expertise and empowerment across all engineering functions.'*

Development organizations receptive to the idea of building a security culture generally employ an ad-hoc mix of activities to accomplish the same, such as:

- **CTFs (Capture-the-flag) Exercises:** Time-boxed gamification exercises geared at helping development team members learn security concepts in a "lets-hack-dummy-application" way. Often, the belief is that attendees will automatically start applying the knowledge gained during such exercises in their day-to-day roles, but this hope is often misplaced when these exercises are conducted in a vacuum.
- **Finding Firefighters:** Development team members (often senior developers) who have been "voluntold" to help out with one or more security activities due to a pressing need such as audit finding, client ask, etc. For example, they are told to help triage the security scanner report shared by the client and teach others informally.
- **Informal and/or half-baked programs:** Security teams often create an informal understanding with some development team members around the discussion and execution of one or more security activities. While creating these relationships is helpful, such ad-hoc arrangements do not have management-level visibility or support, are not formal, and lack well-defined incentives.

Unfortunately, in the absence of program-level thinking and support, these types of ad-hoc efforts are usually met with struggle, adoption resistance or complete failure.

What is missing from these ad-hoc approaches to software security culture development is the "glue layer" at the program-level to cohesively drive individual project-type security activities. This layer relies on resident Security Champions — members of the development team that have been identified and empowered to stay constantly involved in helping to guide and execute security activities on a day-to-day basis. Unlike security team experts with limited or restricted availability, SCs are consistently available to support SDL execution and security activities at the development team level.

An SC program can drive significant value for any organization looking to improve software security when designed and implemented effectively. To get things started, the next chapter will describe the qualifications and duties of a typical SC.

# Putting a Face to Software Security Champions

*By Kristian Beckers, Siemens and John Martin, Boeing with Nick Ozmore, Veracode*

> *Product team manager: "The security people are smart but they're like helicopter parents. They fly in, make a lot of noise telling us to do things differently, then fly off. We're left with trying to figure out how to handle all the stuff they said 'NO' to."*

The first chapter talked about why a Security Champion program is needed, including the common misconception among many software development teams (and the culture at large) that security issues are at best a hindrance and at worst a roadblock to productivity. In this chapter, we'll talk about the qualifications and functions of an SC and demonstrate that security does not have to be the roadblock it is perceived to be. In fact, productivity, speed and quality improve as a result of a security mindset.

## *So, who is an SC?*

Typically, an SC is an active software developer who is also a knowledgeable member of a larger software security community. They are embedded into a software development team in order to accelerate development, by identifying and solving security problems early in the development lifecycle. SCs ensure that security quality gates are met throughout the development lifecycle. The SC is empowered to escalate problems to the security team and management. In short, SCs are enablers inside of development teams that prevent security from becoming a blocker for product release. In this way, SCs empower development teams to produce continuous security value for customers.

## *What characterizes an SC?*

SCs face the challenge of integrating security by default (by design). It is the SC who ensures that security is integrated right from the beginning. They work to help developers produce secure code starting with the very first line of code they write. They also ensure that security makes it onto the product roadmap. They also need to make sure that security is not only in their team's code, but also in code of upstream vendors.

The role of an SC should be treated as a primary responsibility, ideally a full-time or near full-time job. SCs need to partner closely with the broader security team and be able to prioritize security issues based on risk. In this way, an SC is a context-aware topic driver in the engineering team and interface/translator to the security team. An SC should not be a Scrum Master, a PO (Product Owner), or a person that has multiple roles already and no time to cope with the various challenges producing continuously security value presents.

An SC also checks that verification-type activities are done properly. The SC makes someone accountable to run testing tools and ensures that they know how to interpret the results while keeping the SC involved, at least in an awareness capacity. Also, the SC can be responsible for Secure Code and Design Reviews within their respective teams and assign OPS responsibilities for security issues.

## *What else do Security Champions do?*

SCs are also experts at helping and coaching teams. One SAFECode member company that uses risk-ratings as compliance and production-gate criteria (higher risk software had increased test and compliance requirements prior to production) had a development team that was using binary scanning as the final code quality gate and was repeatedly having a 'one-and-done.' They would scan once at the end of the production cycle, pass, and go to delivery. When the development team began including an SC, it began to recognize that its current "one-time" scan approach was flawed and incomplete. The SC was able to conduct a risk assessment and to help the team identify better components, deal with the increased volume of security findings, and hit the appropriate security production gates. Within just a few months, the team went from being a false-negative (unreported high-risk defects) to proactively identifying and addressing security issues.

The core duties of SCs can be categorized into three areas: (1) Directly support team members in security activities and foster security expertise and awareness among the development team (Security Awareness Duties); (2) enable the integration of software engineering activities with the development lifecycle (Enable Security Duties); and (3) Understand intrinsically both the company's and their assigned product portfolio's risk appetite (Risk-oriented Duties) and apply this understanding to the entire software development lifecycle. Below we break these three responsibility areas down in more detail:

1. Security Awareness Duties include:
    a. Fostering security expertise and awareness among the development team
    b. Motivating team members to address security concerns early and continuously, and raise awareness of security issues
    c. Communicating security status (achievements and problems) to the security team and team members

2. Enable Security Duties include:
    a. Ensuring that security features are prioritized appropriately and not handled with less urgency than functional features
    b. Ensuring that security is addressed right from the beginning (security by design)
    c. Ensuring that security is integrated and injected everywhere and that the development team has the right tools they need to succeed (end to end)

3. Risk-oriented Duties include:
    a. Ensuring a risk-oriented approach (i.e., Risk Appetite) that includes knowing the risks the company or domain is willing to take on
    b. Checking that the risk-oriented approach is integrated into all stages of the software engineering process (SDLC)

It is important to note that the detailed definition of the duties of an SC may vary between organizations of different sizes, domains, and security process maturity. For example, one SAFECode member stated that their SCs aren't pushing new security features, but ensuring the wider development team is following secure development practices and using/reviewing tool output, performing secure code reviews, etc.

Various roles in the software engineering context can become SCs or SC Supporters and contribute to the security effort:

- Engineers know the details of one security mechanism and can check if details of, for example Identity and Access Management (IAM), are securely used and best practices are applied.
- Developers know the nitty gritty details of implemented applications and can spot security problems in code (4 eyes principle or pair programming).
- Solution Designers/Architects understand how system parts interface and where security problems in system composition, commissioning, and integration occur.
- Program Managers explain to the development team the value security creates for their product and why it provides trust to their customers.
- Product Owners understand the security value of the functionality being built in the current sprint. They ensure that acceptance criteria exist that reflect these.

SCs must have knowledge in key areas relevant for conducting security tasks. At least the following competencies must exist or be obtained with training:

1. Software Engineering
   a. Software Development Methods and Tools
   b. Secure Software Development and Deployment

2. Threat and Risk Analysis (see also SAFECode's white paper on Tactical Threat Modeling[2])
   a. Threats Identification & Mitigation
   b. Attack Path Analysis
   c. Software and Information Risk Analysis (such as Threat Modeling)

3. Vulnerability Monitoring and Incident Handling
   a. Vulnerability Lists (e.g. CVE) and Ratings (e.g. CVSS)
   b. SIRT & PSIRT Playbook support

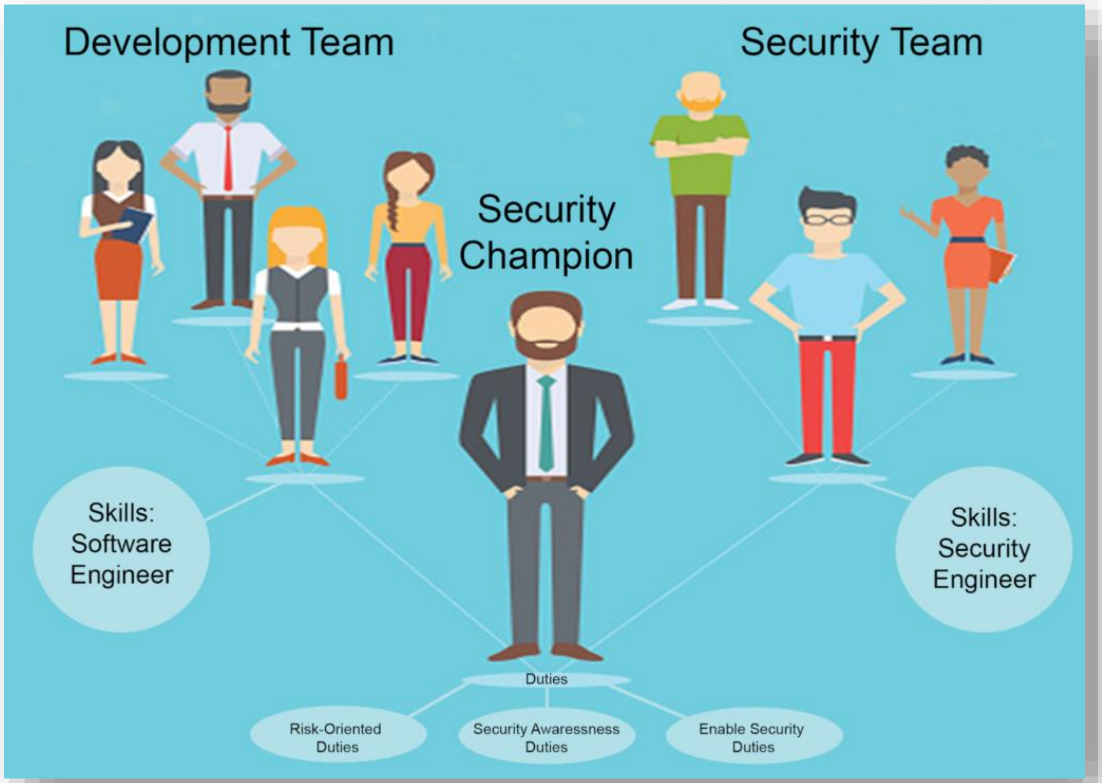
4. Soft Skills
   a. Conflict resolution
   b. Discussion and presentation skills
   c. Motivating people
   d. Drinking a lot of coffee with all stakeholders all the time (team, managers, etc.)

In some contexts, professional certifications may also be required or helpful depending on business needs. While SAFECode members do not endorse any one certification, examples of potentially relevant certifications include: the Information Systems Audit and Control Association's (ISACA)[3] Certified Information Security Manager (CISM) and Certified Information Systems Auditor (CISA) certifications, as

[2] https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf
[3] https://www.isaca.org/pages/default.aspx

well as ISC2's[4] Certified Information Systems Security Professional (CISSP) and Certified Security Software Lifecycle Professional (CSSLP) certifications.

Summing up, an SC has both security engineering and software engineering knowledge. SCs must be able to talk to experts in both fields and be a catalyst for the unification of both fields.



[4] https://www.isc2.org/

# How to Build an Effective Security Champions Program

*By: Tania Ward, Dell with Altaz Valani, Security Compass*

Identifying or hiring an SC is just the first step. They also need to have the right support to truly be empowered to be difference-makers.  In this chapter, we answer the question, “How do you build an effective SC program in your organization?”

## *Lay the foundation*

We all know what happens when a house is built on a weak foundation. It’s unstable in the long run. Building an SC program is no different. There are foundational requirements that need to be in place for the program to be sustainable and effective:

- *Understand your existing security culture* – putting on your security super-hero cap and reactively bouncing from one development team to another won’t save the day. You need to understand the current state of your culture first before you start systematically adding foundational security elements into it.
- *Management/Executive buy-in* – this isn’t about getting a sheriff badge. It is about instituting sustainable change across the organization by getting support to create and roll out a program. It all starts with getting management buy-in and presenting factual data on how that program can save your company money, create value, or minimize risk (through better resiliency or compliance, for example). It is vital that you make the security program measurable. Show the long-term value you hope to achieve as you implement the foundational blocks of the security program.
- *Buy-in from Engineering/SCRUM team* – the team needs to be aware that they are not policed by a new police officer in their town but supported by a member to resolve upcoming problems early on in a practical and risk-oriented way. The team’s support is critical for the success of the role. We need support from top and bottom for the SC to succeed.
- *Policy & Standards* – a book without pages, is it really a book? Put developer-friendly words to your pages and define what ‘building security in’ using concepts like **Security by Design** (meaning across the different phases of your SDLC) and **Security by Default** (making sure the product is secure at deployment). This will help you build a stronger foundation with the developer community in your organization.
- *Funding* – it’s important to have financial backing to help drive reward & recognition, invest in training/conferences/workshops, or sponsor fun activities such as Capture the Flag (CTF) to keep teams hungry for more. You need to engage teams and build a foundation based on proven financial commitment. Obtaining this commitment can then be used as the basis for achieving grassroots developer support.

### *Define and operationalize your program*

This is where you define the main components that will make up your program. Yes, the sounds of nails being hammered in, and frames going up is the sign of progress. However, don't build it too fast!

- *What is your organizational structure?* An SC is embedded into a software development team in order to accelerate development and to identify and solve security problems early in the development lifecycle. You should consider:
    - Whether the program is centralized or decentralized based on the size and development team layout of your company
    - Key problems that the SC will help address in different areas
    - How many teams the SC is expected to support
    - Geographic distribution
    - Risk tolerance across the organization

- *What are the roles and responsibilities?* As we discussed in Chapter 2, the following considerations should be addressed:
    - The target persona of your SC
    - Clear SC roles and responsibilities that not only define the skills needed, but also set clear expectations
    - SC influence by business function as SCs may not only exist in engineering teams, but may also be extended to end customer projects
    - Additional SC functional roles, for example, a developer, QE, or PM
    - Reporting structure that clearly outlines to whom the SC should report, for example, Engineering, Product Management, or Security
- *Where do we find SCs?* There are a number of approaches that have been adopted successfully by SAFECode members:
    - Anyone can become an SC. You need to provide a clear security certification and qualification program.
        - Identify engineers who have a willingness to learn about software security and demonstrated security skills
        - Sell the role as an employee development opportunity
        - Volunteering-driven approach as opposed to "volun-telling". Forcing someone into the role will not last
    - Hire security Subject Matter Experts (SMEs) to fulfill the SC roles and responsibilities.
- *Do I need an operational playbook?* Everyone has a game plan and so should you. Whether it's having regular sync ups, materials ready to go, a cadence for newsletters, or a centralized community – an SC's knowledge can't be static. It's important to define a personal and career growth plan for this role. This requires SCs to have sufficient time to expand their knowledge and skill set. SCs must be enabled (intentional allocation of time and budget) to engage in the following activities:

- *Capture the Flag*: Keep the skills relevant and evolving based on threat landscape
- *Tools*: In a fast-paced environment, tools can save precious time by automating certain security tasks. The right tool is one that easily integrates with engineering's existing processes and tools and is customized to reduce false positives in the environment in which it is being used.
- *Train the trainer*: Train your SCs on the "what?" and "how?"
    - Aligning with what we mentioned in our second blog, the "what?" can include Secure Development Lifecycle processes, secure design principles, secure coding, security testing, tools, and writing security requirements.
    - The "how?" can include internal or external computer-based training (CBT), a buddy system with members from the security team, and on-the-job training. We do not recommend reliance on CBTs alone.
    - Workshops on topics like threat modeling[5] and code analysis.
    - Providing sufficient budget and time in order to update and acquire skills continually without having to go through a lot of red tape.
    - Awareness training so they know the latest security issues to look out for.
- *Conferences*: The goal is to network and share information with other industry security practitioners.
- *Security Forums:* This can be used for educational purposes especially if there are new attack vectors and threats. It can also be used as an avenue for presenting issues.

In conclusion, just finding an SC and telling them to go fix security will not work. To be successful, SCs need the support of a formalized, management-backed program with clear organizational and professional goals.

***Additional resources that may be helpful:***
https://resources.infosecinstitute.com/gartner-report-designing-security-champion-program/#gref
https://www.cypressdatadefense.com/technical/top-reasons-turn-team-developers-security-champions/
https://www.owasp.org/index.php/Security_Champions_Playbook

[5] https://safecode.org/safecodepublications/tactical-threat-modeling/

## Warning: Six Signs your Security Champions Program is in Trouble

*By Tania Ward, Dell with Altaz Valani, Security Compass*

Sometimes, despite the best intentions, Security Champion programs can run into trouble. Often, when launching a new initiative, it takes a bit of trial and error to get things right. Many SAFECode members have already learned some of these lessons in the course of our own program design and implementation, and in an effort to help others avoid some pain, we thought it would be helpful to share some common characteristics of an SC program that's in trouble. Here are a few things to watch out for:

1. **Diminishing resources:** When resources start to go down, you know your program has been deprioritized. Like the game of Snakes and Ladders, go back to the start and reassess your foundation.
2. **Over-reliance on Computer-Based Training (CBT):** With so much out there, it's easy to get caught in the whirlwind of education and think you are doing enough by providing links to various online resources. But our collective experiences have shown CBT is often not adequate on its own. Instead, Consider a blended approach to training. Add instructor-led training and mentoring to the mix, and offer content targeted both at awareness-building and skills-building, as well as training courses focused on promoting specific behavioral changes. Providing the right mix of training – both in terms of content and format – keeps the program fresh and interesting and supports growth in all areas of security culture-building. If you don't see any improvement, verify that your training program is focused on proper teaching, understanding, and retention.
3. **Lack of communications:** If you're not fully committed to executing on your SC program, then don't expect others to be fully committed.
4. **Security is seen as a feature and not as a process:** When engineering teams view security as a feature it is easily deprioritized. Security should be seen for what it is: a process that is designed to build security in (Security by Design) with the end goal of improving the overall quality of your product.
5. **Bad-cop perception**: Driving security through "FUD" (fear, uncertainty, and doubt) and/or as a compliance requirement will only lead engineering teams to view it negatively. Speak their native language and be an integral part of their team.
6. **Lack of accountability:** If the team thinks security is not achievable, challenges should be discussed openly as opposed to simply worked around.

Recognizing the symptoms of program trouble early will help you to diagnose the problem and fix them quickly. A security culture through an SC program is built on aligning all aspects of engineering from the start. Behind every security-aware engineering team stands a SC who has paved the way. So, who's paving the way for our SCs? The next chapter will focus on how to roll out an SC program and how to keep SCs engaged.

# Kicking off and Keeping up with a Security Champions Program

*By Vishal Asthana, Security Compass (former) with Manuel Ifland, Siemens*

> *"Yes – I understand the need for a Security Champions Program and have built a program strategy that has executive management support. Now what? Are there recommended approaches towards rolling things out? Any pitfalls to avoid?"*

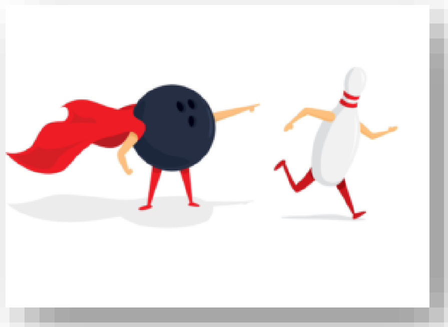
This chapter will cover how to roll out a Security Champions program in a sustainable way. As you can imagine, kicking off the program with a huge disruption to engineering's workflow will not help repair security's reputation as an obstacle. So, a thoughtful kick-off is key to successful program adoption.

First, consider using some sort of change management framework (e.g. Kotter's Eight Steps to Change, Kubler-Ross Five Stage Model, Prosci's ADKAR Model) from the start to help guide implementation.

Second, we highly recommend starting an SC program with a pilot phase. In other words, roll out the program with a distinct team of people within an organization without affecting the rest of the organization. Such a pilot would establish if the program works as planned and if so, strengthen the case for buy-in by management. Other important advantages of the pilot are to adapt the planned program to the actual working culture of the enterprise and to establish if scaling is possible or if things need to be tweaked to enable it. Usually, a smaller (yet important) development team should be selected for running the pilot program.

In addition to management support, getting buy-in from each of the pilot program team members is key to making an SC program pilot successful. Compare it to quitting smoking. If someone does not actually want to quit smoking, registering them for a quit-smoking program does not really make sense. The same applies to the pilot – it is important that the piloting team is open to it and wants to get help in integrating security into their processes.

Rolling out a SC program enterprise-wide says a lot about an organization's seriousness about building a security culture. But the job doesn't stop there! After an SC program has been launched enterprise-wide, it is equally important to take measures to sustain it. For example, one of our series contributors once worked with a large financial institution that onboarded 160 SCs to their program a few years back. That is a great feat and big accomplishment. Unfortunately, they did not build an ecosystem to sustain their SC program, and eventually found that the designated SCs were not actively engaged within their teams and with the central security team. Therefore, even this promising program start led to little/no change to the overall security culture within the organization.



In order to prevent similar disappointment and sustain an SC program, building a supporting ecosystem is key. In large enterprises, the central security team typically plays a crucial role in building this ecosystem. In smaller enterprises, the designated SCs might need to self-organize in order to accomplish the same objective. A list of commonly used tactics to achieve this objective is below:

- **Build an SC Cohort:** Start with having a common mailing list (or a communication tool the engineers like and are used to) for all designated SCs to share important announcements on the SC program, AppSec in general, etc. This also tends to create a sense of belonging to a cause. In parallel, build and regularly update a central AppSec Wiki for to help ease knowledge-sharing. Organize a recurring meeting of SCs on a periodic basis, say monthly. Use such meetings to recognize specific efforts of SCs and reemphasize the SC charter, especially highlighting how SCs fit in the AppSec wheel.
- **Socializing Software Security across the development team:** Regularly socialize the concept of Application Security at all levels in a development team, not just at an individual developer-level. This is achievable through brown bag lunch-and-learn sessions, scheduled open hours with AppSec Subject Matter Experts open hour sessions to discuss "anything security," and similar events. During such sessions, encourage the designated SC in a team to drive or lead the effort. This would help "build up" the SC's image as the resident AppSec SME.
- **Make security fun, not just a heads-down activity:** Gamification in the form of hackathons typically helps achieve this objective. Have the designated SC lead in organizing such exercises on a periodic basis.
- **Train-the-trainer mindset:** The SC's role should gradually evolve from being pure tactical security guides to acting as coaches who train developers so that they can tackle things on their own, reaching out for additional clarification or unresolved topics when needed. In the absence of this gradual cultural shift, security will continue to be viewed as someone else's job (in this case an SC) instead of as the collective accountability of every developer.

In summary, using established change management processes and launching the program with a pilot to identify and correct any problems early (and avoid serious disruption) will help an SC program start on a positive note. To sustain that success, SC programs need ongoing care and feeding to socialize security awareness, build a security-supportive culture, and evolve the program to keep up with changing business needs.

## Defining Success: Is Your Security Champions Program Working?

By Vishal Asthana, Security Compass (*former)* with Manuel Ifland, Siemens

> *"If You Can't Measure It, You Can't Improve It" -- Peter Drucker*

Metrics are a tangible way to track the effectiveness level of an initiative, program, or one or more activities. Their absence often results in reliance on guesswork, slow/no progress, loss of momentum etc. Note that pure quantitative metrics often carry the risk of being gamed to show superficial progress and should therefore be considered in combination with qualitative metrics. Simply put, avoid relying only on a single metric to measure the ongoing success of the SC program.

We recommend organizations customize a combination of metrics that align with their program objectives. To help guide this process, here is a list of commonly used quantitative and qualitative metrics for consideration:

### *Quantitative Metrics*

- **Percentage of team members who have received foundational and/or code-specific training with the SC's guidance**: For example, in a 30-member team with one SC, let's say 20% were trained in the first session, how many have been trained incrementally since?
- **Percentage of senior team members who have received training**: For example, if the SC started with training 50% of lead/senior developers, is this number showing an upward trend over time?

- **Decreased number of entry-level questions to SCs:** For example, if an SC was handling an average of five questions around how to perform vulnerability triaging from twenty developers, when the program was rolled out first, is the number showing a downward trend?
- **Decreased number of recurring similar (internally and externally found) vulnerabilities:** For example, if an organization keeps seeing SQL injection and XSS vulnerabilities again and again, it is a positive trend if the number of those decreases over time.
- **Completion of individual training modules by the development team.** For example, which and how many training modules did developers complete including final tests?

### *Qualitative Metrics*

- **Feedback (from SCs):** As developers learn more about AppSec under their SC's guidance, is there a demonstrated decrease in them pushing back/resisting security efforts?
- **Periodic short surveys (from development team):** Use real anecdotes from developers regarding the SC program's effectiveness. eg: 1-3 question surveys asking 'Do you think the SC program is effective? Why/why not? Do you know how to report a problem to the security team?, etc.' Doing so keeps metrics sane and prevents over-reliance on quantitative aspects only. Apparently, Amazon uses this anecdote-based approach in a big way. On one occasion, Jeff Bezos called customer care support line during an Upper Exec meeting and had to wait for 4+ minutes before getting through to a support representative. He did so as a means to do a quick run-time validation after the Customer Support Head claimed <1 minute wait time.[6]

A combination of these metrics can then be rolled up into a dashboard for Engineering Leadership, Central Security Team, and other potential stakeholders. This information can help guide decisions about sustaining program investment as well as maintaining or adjusting specific program elements. In short, an effective measurement approach can help you determine if your SC program is successful, and if not, provide some insight into how to get it there.

[6] (https://www.businessinsider.in/Amazon-executives-sat-through-a-brutally-uncomfortable-phone-call-that-showed-them-just-how-much-Jeff-Bezos-cares-about-customers/articleshow/63851499.cms

## Conclusion

Many SAFECode members have found software SC programs to be of significant value to their software security effort. The advice offered in this guidebook all comes from the real-world experience of our authors and collaborators in setting up and managing an effective SC program.

For those interested in hearing more from the authors, we invite you to listen to our SC author podcast discussions where they discuss their experiences with SC programs and offer additional insight into the ideas presented in this guidebook.

**Security Champions Podcast: Final Thoughts**[7]
**Security Champions Podcast: The Importance of Security Champions**[8]

To learn more about SAFECode and how to participate in future collaborations like this one, please visit our website at www.safecode.org.

[7] https://safecode.org/security-champions-podcast-final-thoughts/
[8] https://safecode.org/security-champions-podcast-the-importance-of-security-champions/