DevSecOps

dynamic, fast, effective
and secure

swisscom

Publishing details

Editor	Swisscom Ltd
Author	Group Security
Editorial Staff	René Mosbacher, Faktor Journalisten AG, Zurich
Illustration	Agency Nordjungs, Zurich
Copyright	© March 2019 by Swisscom Ltd, Group Security, Berne

Contents

1 Foreword

Today, we live our lives online. The internet has no geography. It has no borders. By creating the internet, mankind opened up a Pandora's Box where tangible borders and recognizable enemies ceased to exist.

In fact, we are the first generation in mankind's history that is more likely to become a victim of a crime in the online world instead of the real world. This is a major shift.

In this new world, companies are facing new kinds of risks too. Headlines are full of examples of data breaches in large companies. Today, every company is a software company, and every company should pay attention to online security. Today, cyber security should be a permanent board-level topic.

In the end, there are only two types of security problems: technical problems and people problems.

Technical problems can be fixed by patching. But people problems can't. There's no patch for stupidity. No matter how many times you tell them, users will always follow every link, they will always double-click on every attachment and they will always type their password on every phishing site. Guaranteed.

Vulnerabilities in our systems are, in the end, just bugs in the code. Why do we have bugs in the code? Because programs are programmed by human beings and humans make mistakes.

So how can we minimize the number of mistakes programmers make? We need better security engineering. We need a better culture around security. Security should be built-in, instead of being a perimeter around applications and the data they handle. Security should be an integral part of the entire life cycle of a software project.

Security does not end with software engineering and security audits. No matter what kind of a system you build, you should always assume there's going to be a breach anyway: someone will get in via a route you never thought of. Therefore companies should focus on detecting breach and reacting to them. Resilience is the key.

We all have limited resources and limited budgets to defend our networks. Understanding the enemy enables us to focus our resources to where it matters the most.

And when we are trying to fight hackers, one of our best resource is hackers. You see, we need good hackers to catch bad hackers. This is why bug bounties work. Starting a bug bounty program takes a lot of work, but it will give new viewpoints to the security of your systems. I personally love the fact that many large companies now run open bug bounties, as this gives me an easy answer to give to eager young hackers I meet. "So, you want to hack a company? Okay: Go hack Microsoft, or Google, or Apple. That's perfectly legal, as long as you do it within their bug bounty programs. They'll even pay you for it."

I've been working with security for close to 30 years. Our work never ends.

I wish you the best of luck in developing secure systems. This booklet on DevSecOps is a great example on practical things we can do to make the world around us more secure.

Thank you for your work.

Mikko Hypponen
Chief Research Officer, F-Secure

2 About this publication

2.1 Why this booklet?

Swisscom, with around 20,000 employees, is a large company with roots in telecommunications. Over the years, further business fields have been added and today Swisscom is also a classic IT service provider and the most important information and communication technology (ICT) company in Switzerland.

In 2016, it was decided to align the company with agile principles. The first step here was taken by the Innovation department, followed by the Development departments and now the whole company is on board. How Swisscom uses DevOps in practice is explained in chapters *3.3* and *3.4*.

In the course of the transformation, it became clear that the support previously offered by the central security organisation could only be applied to a limited extent in an agile environment. On the basis of this insight, the security initiatives described in this booklet were launched or expanded.

This publication offers a collection of best practices from the Dev(Sec)Ops environment. It was written from the point of view of the enterprise environment and also documents the experiences within the company so far. This should benefit other companies and experts facing similar tasks.

> In 2016, Swisscom created its first Development department based entirely on the DevOps principles. This enabled the company to learn how to scale agility to an area with more than 100 people.
>
> In addition, it was found that the "agilisation" of development teams did not reduce the problems in the company. More and more operators were involved in development activities, who then also ensured that really operable solutions were built. But for the actual 7×24-hour operations of all the existing applications, fewer and fewer personnel were available.
>
> For this reason, the organisation – according to the motto "you build it, you run it" – was once again restructured for 2017. Instead of the classic division between development and operations, IT was reorganised into a software and an infrastructure area. Within these areas, however, all tasks (development, testing, security, operation) are under the responsibility of the product team. This has created and continues to create a much greater sense of responsibility for the code developed.

With these changes, more and more areas were introduced into the workings of DevOps. Soon, however, it became apparent that a scaling framework was needed. Swisscom decided to introduce the Scaled Agile Framework (SAFe) and applies it wherever agile at scale is required.

The adjustments have significantly increased operational agility. However, it has also been shown that the advantages of DevOps are not limited to development and operations – on the contrary. At the core of DevOps has always been the desire to include the entire value chain.

In 2018, for example, the focus was increasingly on transferring agility from the operational to the strategic level. Business should be more involved and business processes should be adapted if they contradict the agile principles.

A DevOps journey needs time and endurance. And so, Swisscom will continue to work on its continuous development over the next few years. On the one hand, this concerns technical capabilities such as decoupling, continuous delivery or test automation. On the other hand, however, it is above all about the cultural level, i.e. achieving more self-organised teams and a stronger flow optimisation.

2.2 Target audience

This publication is aimed at experts who can or want to help shape a DevOps environment themselves. In particular, persons are to be addressed who must also guarantee security in a highly agile IT environment. You will receive an explanation of the basic concepts as well as short field reports from Swisscom's environment.

It is assumed that the readers have a basic knowledge of DevOps. Resources, such as websites or books, that allow a more in-depth look at topics are linked throughout or listed at the end.

2.3 Structure of the publication

The first part of this booklet deals with the basic ideas and principles of DevOps.
The following aspects are highlighted in particular:

- Why – why does an organisation choose DevOps?
- What – what defines and qualifies DevOps?
- How – how is DevOps introduced?

This is the basis for the second part, which specifically explains the security aspects of DevOps. The subchapters first list which points must be specifically taken into account with regard to People, Process and Technology.

The second part is structured according to a simplified Software Development Lifecycle (SDLC, see Figure 1). First, it discusses cross-cutting issues such as Training & Awareness (T&A) and the way in which security must be set up to fit into the dynamics of a DevOps organisation. These issues are crucial if the security department aims to participate in shaping a successful DevOps organisation. Then, step by step along the SDLC, it is examined how security should be handled so that its effectiveness can develop within the DevOps environment.

A chapter follows on the deployment pipeline, the actual factory. Not surprisingly, automation plays an important role here. Finally, it is explained how the new approaches can be profitably used for the rapid detection and defence of attacks.

At selected points, experience reports and solutions from Swisscom's environment are included. They should help to better understand described principles and to learn from mistakes already made. These sections are each highlighted in blue.

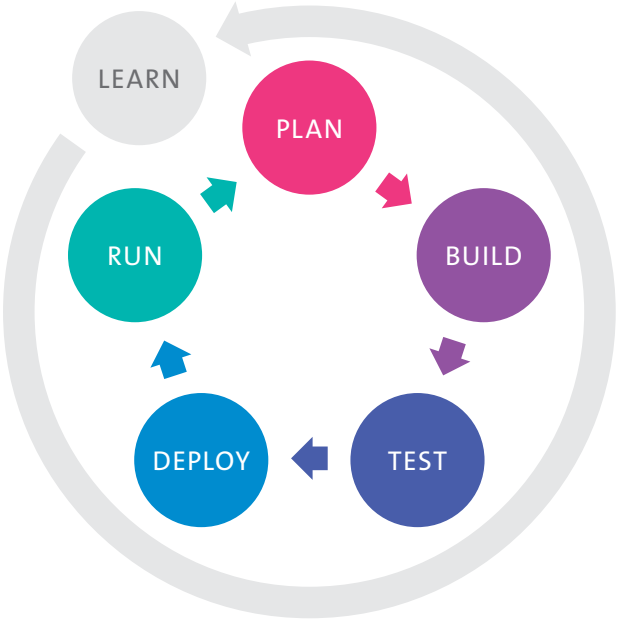

Fig. 1 | A simplified Software Development Lifecycle

3 DevOps – Introduction

3.1 Why – why DevOps?

Faster, better, cheaper, happier – DevOps is about creating business value faster and creating more robust systems that lead to more customer-focused products. Appropriate forms of cooperation and team organisation ensure that these efforts are not at the expense of employee satisfaction. Whoever masters this is equipped for the ever faster moving IT world, which is also characterised by increasing volatility, uncertainty, complexity and ambiguity (keyword VUCA: volatility, uncertainty, complexity and ambiguity).

3.2 What – what is DevOps?

DevOps has evolved from various existing movements. These include the Agile Manifesto, the Lean Movement, the Continuous Delivery Movement and the Toyota Kata [1]. Thus, DevOps is not a framework, a tool or an organisational form. Rather DevOps can be regarded as the consistent integration and further development of the mentioned concepts.

The IT expert John Willis is one of the DevOps pioneers and strongly influences the scene. He introduced the acronym CALMS to describe DevOps (first it was just CAMS – then Jezz Humble added the L) [2]. CALMS consists of:

- **Culture** – a cross-functional collaboration with a culture of shared responsibility
- **Automation** – as many tasks as possible are automated
- **Lean** – visual representation of value flow and work packages flowing at any time
- **Measurement** – forming, validating and learning hypotheses [3]
- **Sharing** – sharing responsibility and success, both between operations and development and across team boundaries

3.3 How is DevOps introduced?

3.3.1 Skill sets

A good overview of the skills required is provided by the IBM reference model «DevOps: The IBM approach – Continuous delivery of software-driven innovation». [4]

[1] The DevOps Handbook by Patrick Debois, Jez Humble, Gene Kim, John Willis

[2] DevOps Culture (Part 1) by John Willis, URL: *itrevolution.com*

[3] Hypothesis-Driven Development by Jeffrey L. Taylor, URL: *drdobbs.com*

[4] DevOps: The IBM approach – Continuous delivery of software-driven innovation (particularly page 6), URL: *developer.ibm.com/community*

Swisscom relies on an adapted version of the IBM reference model. The company uses it as a holistic system that not only covers development, but all relevant skills along the entire value stream. The Swisscom model describes skills that are relevant for a DevOps operation, but it is not a flowchart.

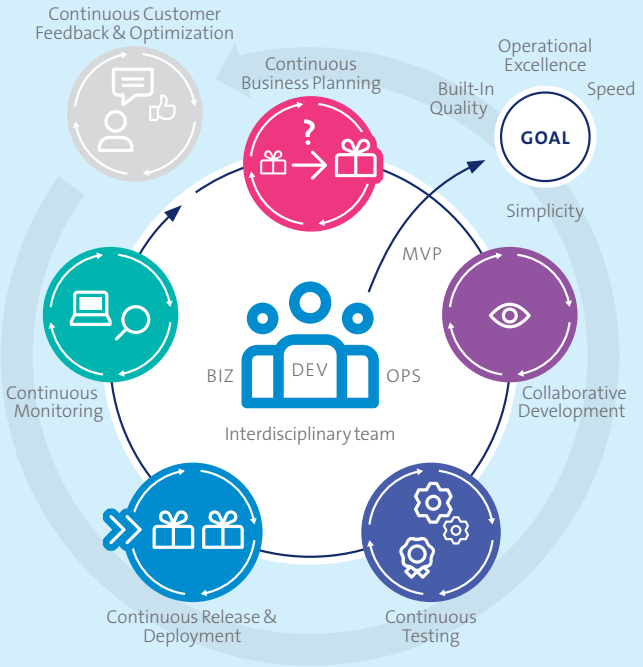

Fig. 2 | Swisscom model of the skills needed for DevOps

For DevOps to succeed, the following prerequisites must essentially be fulfilled:

1. **The team is at the centre:** The focus is on an interdisciplinary, fully dedicated, autonomous team that communicates at eye level *(see also 3.2, «Culture» section in CALMS)*.

2. **Continuous business planning:** This refers to the ability to react quickly to customer needs and to incorporate appropriate feedback on an ongoing basis. The focus is always on the customer's needs and defines the product.

3. **Collaborative development:** This term describes the ability to continuously generate value in short iterations and interdisciplinary teams consisting of representatives from business, development, security, testing and operations. This assumes that source code is continuously checked in to a code repository – in the best case several times a day. Everything is regarded as code – this includes infrastructure, deployment scripts, monitoring, test data and more, in addition to the actual application source code.

4. **Continuous testing:** This includes the ability to test the code in a reproducible and repetitive manner. Automatic tests allow you to examine it in many ways immediately after checking it into the repository. This allows early and continuous testing instead of just at the end of a process chain, shortly before releasing to production. This makes testing part of the overall process. This in turn improves cost efficiency as errors can be corrected earlier in the cycle and thus more cost-effectively[5].

5. **Continuous release & deployment:** This is the ability to automatically carry out a build and deliver it to any environment. This is necessary to exploit the full potential of automated testing. The code check-in to the repository allows you to carry out a number of automatic processes, from build to test to deployment. This significantly reduces manual activities.

6. **Continuous monitoring:** This involves monitoring the entire system on all platforms at all times. This makes it clear immediately after the release of a new feature whether it influences the performance of the overall system. This early feedback helps to further improve quality.

7. **Continuous customer feedback and optimisation:** The ability to continually improve software based on measurements and feedback from both the customer and the business. Customer behaviour is very consciously analysed and various channels are used to receive early and comprehensive feedback.

[5] Figure: Relative Cost of Fixing Defects by Maurice Dawson, URL: *researchgate.net*

3.3.2 Scaling – DevOps in large organisations

All agile approaches focus on the team. This is a good thing because the value of a small, dedicated team (ideally 5 – 7 people) cannot be overestimated. In the best case scenario, such teams can develop their services completely autonomously and deliver code.

Independent teams are rarely possible, especially in large organisations with a strongly coupled IT architecture. This raises the question of how scaling can be solved. The Scaled Agile Framework (SAFe), for example, helps here. It is a collection of best practices that describes a possible scaling.[6]

> Swisscom has decided to use SAFe to practice "Agile at Scale". In reality, this means: If a product is too complex and too large to be carried by an autonomous team alone, Swisscom uses SAFe structures. The framework helps to create a common understanding of how "Agile at Scale" works. However, only those elements and roles of SAFe that effectively offer added value are applied. Swisscom takes over some elements as they are; others are adapted to requirements, such as the role of security.

3.4 Impact – what does DevOps do?

If you want to introduce DevOps into your organisation, you should be aware that it influences the entire value chain. The relevant aspects are discussed below.

For the customer

Actually, customers don't care how the services are produced. They have the legitimate requirement that their needs are met quickly and reliably. Their systems should be available, error-free and reliable. In our rapidly changing world, this is increasingly difficult to achieve with classic models. With DevOps, on the other hand, the customer receives solutions that are better tailored to his needs and can be implemented more quickly.

In the company itself

The State of DevOps Report[7] has been collecting data for several years on how DevOps practices affect companies. It was clearly demonstrated that companies with a high DevOps maturity are economically more successful. However, in this context it is important to point out that DevOps cannot be a cost saving programme. Higher efficiency is the result of well-established DevOps practices – but it cannot be the goal.

[6] SAFe. Scaled Agile Inc., URL: *scaledagileframework.com*

[7] State Of DevOps Report by Puppet + Splunk, URL: *puppet.com*

In the organisational structure
Agility and DevOps rely heavily on autonomous teams. Decisions are taken in an as decentralised as possible manner. Such an environment naturally requires special leadership qualities. It is important that the teams are diverse and cross-functional. Experience has shown that this leads to better solutions.

The composition of the teams should be as constant as possible. This encourages joint and mutual learning. This also results in an important paradigm shift for project/product processes: The work should be brought to the team and not the team to work. This in turn means that work on the continuous development of a product is much more intensive than on self-contained projects. It is important that decisions are made as far as possible where the consequences arise. This is another reason why interdisciplinary teams are so important.

DevOps is also to reduce the organisational silos. In this way, the flow of values is given much greater weight than organisational affiliation. For this reason, the process organisation is also much more important than the organisational structure.

> The implementation of DevOps has led to organisational adjustments at Swisscom, particularly in IT. Previously, application development and operations were located in separate departments with different goals. «Dev» was measured by the number of functions supplied. «Ops», on the other hand, was measured by the stability of the systems and therefore tried to keep the number of changes small. In order to resolve this conflict of objectives and strengthen joint responsibility, these departments were merged. Now each team is not only responsible for the development, but also for the operation of its applications.

In a team and with the individual
DevOps focuses strongly on the team and team responsibility. Goals are set in the team and the team members support each other in achieving them. The focus is on the team goals and not those of individual team members.

DevOps shortens cycles and intensifies cooperation massively. Direct communication between all parties involved must therefore be consciously promoted. The Agile Manifesto provides concepts for this [8].

[8] Manifesto for Agile Software Development by Kent Beck et al., URL: *agilemanifesto.org*

In his book «Drive», Daniel Pink[9] writes that motivation mainly depends on the factors Autonomy, Mastery and Purpose. DevOps does justice to these very well:

- Autonomy is the result of decentralised decision-making and the autonomy of the team.
- Mastery is promoted by the necessary continuous learning.
- Purpose becomes clearer because the responsibilities and effects of the work can be experienced very directly. You are not just a cog in a large machine, you are responsible for a specific product right through to production. Most employees probably appreciate that.

The development process is generally shorter and faster with DevOps. This shortens the feedback loops. This in turn means that information and knowledge, and thus also responsibility, flow back more quickly to the start of the process, i.e. to development. This effect is also called «Shift Left».[10]

With DevOps the members always work in as dedicated a manner as possible for their team. This helps to avoid priority conflicts, develop a healthy team culture and promote joint learning. What has already been described about the influence of DevOps on the organisational structure also applies: the team's work is more focused on the continuous development of products than on the execution of individual projects.

[9] Drive by Daniel Pink, URL: *danpink.com*

[10] Shifting Left – Approach and Practices by Paul Bahrs, URL: *slideshare.net*

4 DevSecOps – security in a DevOps context

By definition[11], security is already strongly mapped in DevOps. Therefore, it seems unnecessary at first glance to give it additional weight with another artificial term like DevSecOps. Experience shows, however, that in the course of "Shift Left", issues are repeatedly neglected – security is particularly at risk here. Essentially, DevOps poses three important challenges:

- **Elimination of fixed process tollgates or milestones**
  What has already found its way into iterative project management processes applies even more to DevOps: Individual milestones that require manual approval are counterproductive for DevOps because they extend the lead time from idea to production. Therefore, DevOps does not allow security reviews to be anchored before "go-live", as is often the case in classically managed projects.

- **Elimination of the separation of powers between system and application (Segregation of Duties)**
  Due to the diversification of skills and the associated end-to-end responsibility within the DevOps teams, all team members receive the same authorisations. The consequence of this is that many more individuals gain access to valuable data.

- **Uncertainty regarding responsibilities for risks**
  What was anchored via the line function in strictly hierarchical organisations is bound to the role in a DevOps organisation. Risks should have an owner who bears or mitigates them.

To take these aspects into account, the security community has created the artificial term DevSecOps. It should help to shed more light on security in a DevOps environment. It also illustrates, however, that all activities relating to the product life cycle must be closely interlinked. For security this means primarily: security activities are only suitable for DevOps if they can be applied iteratively.

[11] The DevOps Handbook by Patrick Debois, Jez Humble, Gene Kim, John Willis

The framework conditions for security can be illustrated using the three pillars *People, Process and Technology*:

People	Process	Technology
People who use DevOps are made to intrinsically consider the security of a product as well	Processes support secure development of a product In changing organisations, the relevant contact persons must be easy to find with regard to security	"Security as Code" is consistently implemented – making it comprehensible, reproducible and mutable

These three pillars are taken up again at the beginning of each subchapter. They aim to describe the most important points in each area of activity. The three columns can be defined as follows:

- **People:** Activities and measures that contribute to the DevOps culture within the company
- **Process:** Everything you need to know about processes and the organisational setup so that security in DevOps gets its proper place
- **Technology:** The core of DevOps is the ability to automate repetitive processes – it's about how to deal with automation and the resulting challenges and opportunities

In a DevOps product development, it is difficult to cleanly assign individual activities to one of the three pillars. That is why the columns are subsequently always presented together.

The following subchapters describe the essential capabilities that should be sought in a DevOps organisation with a corporate dimension. They also show where there are synergy potentials between the various security activities. This helps to minimise the additional effort and the additional friction to what is really necessary. The sequence of the subchapters is based on the various activities of the simplified software development lifecycle *(see chapter 2.3)*, thus:

- Training and Organisation
- Planning and implementation of a product/feature
- Deployment and operations

4.1 Training & Awareness

Central to this chapter:		
People	**Process**	**Technology**
Know where and when security needs to be considered	Training & Awareness is mandatory for the relevant stakeholders	Scalability and sustainability of the awareness and training method(s)

When it comes to providing technical support for decisions, prioritisations or other controlling activities, it is important that all participants receive Security Training & Awareness (T&A). In a DevOps environment, every team member must be aware of where misconduct or a systemic error can have major consequences. Everyone must also be aware of which basic assets are to be protected (for example data).

Good awareness helps to develop a feeling for insecure things. This is very useful because security is often less visible and noticeable compared to other requirements. A healthy gut feeling makes it easier for those involved to think about the consequences of their actions.

4.1.1 For who?

In principle, all members of an organisation are jointly responsible for security. This is why a Training & Awareness programme is needed that imparts a broad basic knowledge and is therefore relevant for all employees (e.g. in the secure handling of data). In addition, such a programme must also impart role-specific knowledge. This allows employees to further their education in their area of competence, if necessary up to expert level.

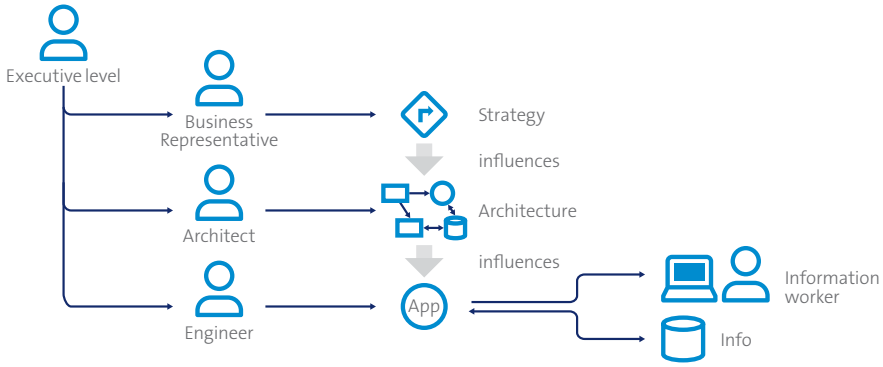

Fig. 3 | Involved roles in DevOps basic knowledge and general awareness

Every member of the organisation must have a minimal basic knowledge of security and data protection. For this reason, basic training courses are to be offered, for example in the form of e-learning. The target group of such trainings includes all roles involved in DevOps:

- **Engineers** – take care of the technical provision of products
- **Architects** – are responsible for correct interactions and processes within products and at their interfaces to the outside world
- **Business Representatives** – prioritise content aspects of the product and thus influence the direction of development
- **Executives** – take responsibility for and manage overarching aspects in the company
- **Information Workers** – the (internal) beneficiaries of provided products

Role-specific Training & Awareness units
Role-specific training sessions for Training & Awareness help team members to identify relevant security and privacy issues and apply their knowledge in practice. The aim here is to convey a basic understanding that, for example, allows an architect to plan in the necessary security components. Or it should help the business representative to plan for sufficient time and resources for security hardening and fixes. For this to succeed, the training content must be tailored to the respective role.

Training & Awareness for specialised areas
Very in-depth, specific security training is especially useful for engineering and development. The goal must be to have at least one specialised person per DevOps teams *(see 4.2.2 → Security Champion)* who has advanced knowledge in security (and data protection). They should act as a link to the central security organisation.

4.1.2 Training & Awareness concept

The concept for Training & Awareness is based on the following three main pillars:

1. Continuous training with short learning units (micro-learnings), some of which also offer repetitive content. Wherever possible, long learning units are divided into several shorter ones. This makes it easier to record the content and increases acceptance among employees.
2. Awareness campaigns repeatedly draw employees' attention to possible dangers (such as fake phishing campaigns). This is intended to create a «background noise» that permanently anchors security awareness in people's minds.
3. Special communication channels help to disseminate security topics and, above all, training & awareness offerings for specific roles.

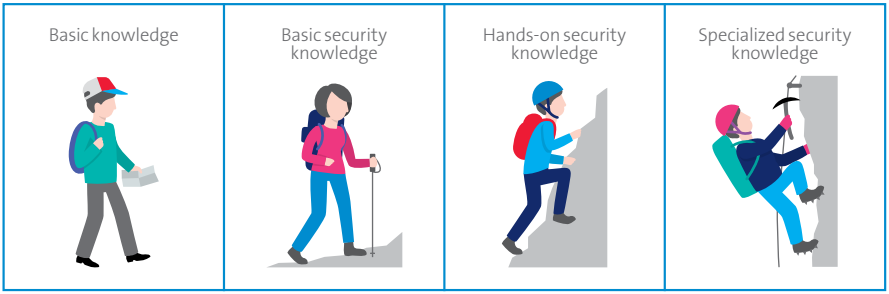

Fig. 4 | The four mascots of Swisscom's certification programme in the DevSecOps area: Trekker, Hiker, Mountaineer, Alpinist

In terms of content, the Training & Awareness concept at Swisscom provides for four levels. With each step, the content becomes more specialised, which also reduces the target audience. Participants can obtain an internal Swisscom certificate at each level. The first stage consists of the **Trekker** modules. Their primary purpose is to raise awareness of internally available and centrally managed services around DevOps. Basic DevOps approaches such as Source Code Management, Continuous Integration/Continuous Deployment are also taught. The participants also learn the common vocabulary, which then runs through the following stages. In the end, even beginners should have the necessary information at hand and be aware of secure DevOps services with which they can quickly work productively. The communication of specific security content is deliberately omitted.

The **Hiker** modules present security services on the one hand. On the other hand, participants will receive an introduction to the most important vulnerabilities such as cross-site scripting and SQL injection. Data from Swisscom's bug bounty programme *(see chapter 4.6)* was used to prioritise the vulnerability classes. Additionally, the Open Web Application Security Project (OWASP) Top 10 project, was used for prioritisation as well. Both stages, Trekker and Hiker, are simple and can be checked in an automated manner by means of a multiple-choice quiz.

The **Mountaineer** modules are attended by people who want to know how to deal in practice with vulnerability classes seen in theory during the Hiker module. For this purpose, participants who otherwise build systems or products should be able to change sides for one or two days in order to get to know the attacker's point of view. Participants who wish to obtain this certificate are expected to be able to apply the findings in their "everyday

job". Their experiences in doing so should be shared with the DevOps community, such that other participants can learn from that.

At the **Alpinist** stage, a high degree of specialisation is required. Here, the aim is to achieve an external certificate that is recognised on the market, for example the Certified Secure Software Lifecycle Professional (CSSLP) or similar ones. The graduates should ultimately be able to influence the internal security culture of the organisation by sharing re-usable contents with the community. Similar to the Mountaineer level, this act of contribution represents also the completion criteria.

4.2 Organisational scaling

Central to this chapter:		
People	**Process**	**Technology**
Keep communication channels short Promote matrix communication	Document and clearly communicate responsibilities	Enable an overview of the organisation and the different roles at any time

To anchor security as an attitude within the organisation remains a challenge, even in the age of DevOps. In contrast to other non-functional attributes such as performance or operational stability, a lack of security is not immediately noticeable in a product. We know that from everyday life: If an application reacts sluggishly, customers become active quite quickly. But if there is a lack of security, they only notice it – if at all – much later, when something has already gone wrong. We must therefore give security the visibility it deserves. If data is lost or manipulated, the effects are often fatal.

Classically, there are around 10 operators per 100 developers in IT, but only 1 security specialist available. Thus, one-to-one support of employees in security matters is unrealistic from the outset. This will not change even in a DevOps organisation. Therefore, it is important that all parties involved in DevOps (business, development, test, operations; *see also 'Collaborative Development' in chapter 3.3*) increase their security competence and are supported by the central security organisation. Matrix communication also helps here. This is the networking of people who have the same role within the different teams. This enables them to constantly exchange information with their peers across the company while contributing their knowledge as a subject matter expert to their own DevOps team.

As already described in *chapter 2.1*, Swisscom uses the Scaled Agile Framework (SAFe) in an adapted form. One of the adapted areas is the integration of security into the organisation – it is not sufficiently designed for Swisscom purposes in the basic framework. This chapter looks in more detail at where and why the implementation of security at Swisscom differs from SAFe. It also shows how security is embedded within the organisational structure.

4.2.1 Skill diversity in the DevOps organisation

The "Shift Left" approach shifts many responsibilities to the DevOps team. This means that in addition to the functional behaviour of the product, the team must also keep an eye on many other qualitative aspects. The range of topics to be dealt with is thus broadened considerably. It therefore requires a holistic view of the product and this is only possible if the team is heterogeneous and diversified.

If there are specialists from many different fields in the team, the individual member has fewer problems in formulating relevant threat scenarios for their field of expertise. This also allows to gain a better understanding of why certain activities that are not immediately profitable can be extremely important. Overall, security awareness in the team increases due to the diversity of skills.

4.2.2 Security roles in the organisation

In order for the gained security awareness to unfold its full effect, it must be supported by a scalable system of security-relevant roles. In the Security Community, the benefits of this scalability are undisputed. It can be implemented using a Security Champion model, for example[12].

Typically, Security Coaches and Security Officers are affiliated with the central security organisation. The Security Officer has an overall view and oversees technical risks as well as resulting business risks. His counterparts are the business decision makers who need to understand and prioritise these risks. The representatives of the business act as risk owners. So, the Security Officer helps in correctly classifying risks so that they can be correctly prioritised when developing a product or service.
The Security Coach primarily deals with threats and resulting risks or vulnerabilities, depending on who he is talking to. He supports the Security Champions in the teams. These are, in turn, permanent members of the DevOps teams and wear the security hat in that specific team context. Together with the engineers in the team, the Security

[12] Security Champions. Open Web Application Security Project, URL: *owasp.org*
Security Champions Playbook. Open Web Application Security Project, URL: *owasp.org*

Champion investigates vulnerabilities, deals with technical threats and, most importantly, makes all of this very clear to his team.

True to the "Shift Left" approach, responsibility for vulnerabilities and threats is concentrated directly in the DevOps team. The central security organisation provides support and empowerment and controls the handling of security risks.

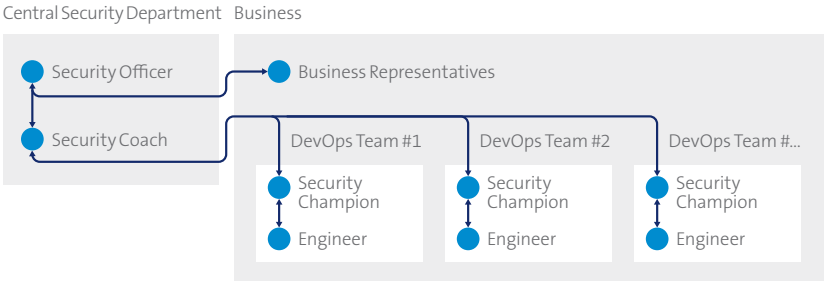

Fig. 5 | Organisational arrangement of security roles

Because DevOps organisations are very dynamic, it can help if the aforementioned roles are maintained in a central registry. This always makes it clear who has what role and in what form they need to be informed. In any case, the definition of new roles should be handled sparingly, as each additional role generates new interfaces and thus additional coordination effort. In order to keep the organisation as lean as possible and to be able to react quickly to changing requirements, it makes sense to introduce security roles step by step. For example, in a first step, the roles of Security Officer and Security Coach described above can be combined.

> If a DevOps transformation is carried out in a large company, this usually generates a lot of uncertainty (in the entrepreneurial sense). In such an environment, a Security Champions model offers the central security organisation the opportunity to become involved at an early stage (as a partial organisation) and to create the necessary structures.

Security Champions

Various, slightly different definitions circulate in the industry for the role of the Security Champion. There is agreement, however, that he should primarily be the contact person in the team for everything to do with security. In this function he promotes awareness. Security Champions help to identify threats and vulnerabilities, make them transparent, and ensure that they are addressed.

However, the responsibility for the security of a product should not lie solely with the Security Champion, but with the team and the organisation as a whole. It's also not intended that the Champion will tackle all security issues alone. Rather, he does this in constant interaction with his team, his Security Coach, but also with other Security Champions *(see below: Security Community)*.

When exchanging ideas with his Security Coach, the Champion specifically benefits from his know-how. In return, the Coach can also get an idea of the (security) situation in the team or project. Thus, the Security Champion enables a consolidated, team-oriented view that reflects the concrete needs.

At Swisscom, the following has been shown in practice: for the acceptance and thus the success of a Security Champion model in an enterprise context, it is important that the Security Champion is not only integrated into a DevOps team, but also speaks their technical language. Moreover, when different teams work on a common product, the higher-level security view must not be neglected. Because each team usually develops a part of the final product, the overall view can otherwise be lost. That's why it's important to nominate a Champion at the business level as well. This «Champion of Champions» keeps an eye on the architecture and the interaction of the different teams regarding security.

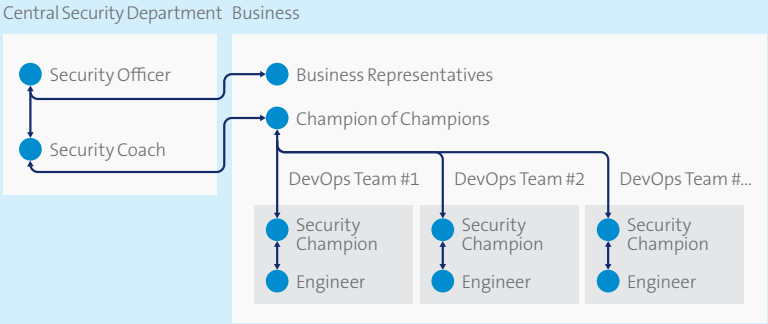

Fig. 6 | The Security Champion of Champions keeps track of larger products.

The larger the organisation, the less it makes sense for the Security Champion to become an actual security specialist on a technical level. To ensure that the necessary know-how nevertheless flows into the end product in a large company, the role of the Champion can be distributed among several people within a product team.

The appointment of Security Champions can lead to tensions. This happens especially when no one in the DevOps team wants to voluntarily slip into this role. Often a member then takes the plunge – or he is pushed into the role. In such cases it takes much prudence and pragmatism on all sides for the "champion against his will" to feel comfortable in his role.

Significantly: with the nomination of the Security Champions, the work is done neither for the business nor for the central security organisation. In order for Champions to be able to play their role, it is necessary to ensure that they are also given the necessary resources (time, money, decision-making authority). It is also important to ensure that they are given the necessary knowledge to enable them to take responsibility for security as part of a team.

But you have to be clear: at the corporate level, the responsibility for security risks and the associated measures always lies with the body that makes the final decision about a product or service. DevOps delegates a large part of the responsibility and thus also the decision-making authority to the teams. However, the security organisation must support the teams via the Security Champion model. And: It must have the competence to intervene and pull the rip cord when safety is grossly neglected.

Security Coaches

The task of the Security Coaches is primarily to enable the Security Champions to do their job.

This point must not be underestimated. If a Security Champion lacks know-how, this leads to additional work for the Security Coaches. If the know-how hiatus remains unnoticed, security blind spots will occur, leading to undetected (and therefore unpredictable) security vulnerabilities in the worst case.

The Security Coach should be available in an advisory capacity to the Security Champions in their daily work. They should recognise, bundle and serve synergies between the needs of several teams. In the case of critical security issues, the Coach can provide the Champion and the project with in-depth support, if necessary, also in terms of content. For example, he can help the Champion with security audits.

The Security Coach is typically the contact person for several teams. He does not concentrate on implementation-technical specifics. Rather, he also helps with the compilation

of the relevant framework conditions, reviews of architectural adaptations and the implementation of a meaningful threat-modelling process.

A Security Coach is particularly effective if he can refer directly to precisely fitting solution modules, procedures or sample implementations for a set of given security requirements. Ideally, he can use a searchable catalogue that links requirements with possible solution variants. By the way, the Security Champion can also profit from such a catalogue. If it is freely available to the DevOps teams, they will be able to find the best solution for themselves, while at the same time meeting all security requirements.

It is also the task of the Security Coach to make transparent technical security risks, which were identified in dialogue with "his" Security Champions. If necessary, he must ensure that all stakeholders understand these risks.

Security Officer
Depending on the size of the organisation, additional security roles can be introduced if needed. Especially in complex and nested organisations, it makes sense to introduce additional levels of abstraction.

The Security Officer is linked to the business stakeholders and has an overall view of the various products and their interaction. This expanded view allows him to better assess where major risks lurk and how the various elements, teams and departments interact. The Security Officer is therefore mainly concerned with security risks applied to a business context. Thanks to his view of the business, he can identify both technical and business risks and communicate them to the relevant authorities. The Security Officer assists in prioritising mitigation measures.

The Security Officer regularly communicates with the Security Coaches in his domain of responsibility. Important information is shared, and the strategic orientation is defined together.

Security Community
If new decentralised security roles are created, it is advisable to link the know-how of the role owners and thus enable so-called matrix communication. This means that there's only one Security Champion role in a team but all Security Champions have the opportunity to exchange ideas with professional peers within the Security Community.

This exchange of experience is particularly important in security. In order to promote and maintain such exchange, it can make sense to set up a Security Guild. It ensures that community members support each other and that solutions to security problems can also be discussed, challenged and worked out together.

Experience has shown that many aspects faced by a Security Champion are also relevant for other teams. Therefore, the solutions developed can also be reused by other teams. If experiences and solutions are shared and developed within a Security Guild, the security maturity of the whole company increases. This in turn relieves the central security organisation.

> At Swisscom, the Security Guild was implemented as follows: all Security Champions and Coaches meet several times a year to work together on security issues. Workshops, training courses and events are organised regularly to increase know-how, promote exchange and motivation. These include, for example, the annual «Capture the Flag»[13] hackathon. There, participants have the opportunity to solve security challenges in a playful way and on the basis of practical examples. The Security Guild also maintains a chat, which enables and accelerates exchange.

4.3 Security in planning activities

Central to this chapter		
People	**Process**	**Technology**
Consideration of security requirements Create understanding of how iterative planning, typical in an agile environment, affects security	Make available efficient and iteratively applicable processes for identifying threats and risks	Support with tools, where possible, to version, share and annotate information

"Everything as Code" is one of the central principles of DevOps. However, it should not be forgotten that code cannot be written without preparation and careful consideration. In principle, security activities should already be considered in the abstract phases "requirements gathering" and "architecture & design" of the software development lifecycle.

[13] CTF? WTF?, CTFtime team, URL: *ctftime.org*

4.3.1 Define security requirements

In any case, the following decisive sources have an influence on the catalogue of requirements for product development:

- Legal provisions regarding the information processed (e.g. the Swiss Telecommunications or Data Protection Act)
- The compliance to be fulfilled (e.g. with the circular of the Swiss Financial Market Supervisory Authority Finma or with ISO/IEC 27001)
- Handling of data and information according to internal data classification and specifications and, depending on the case, customer requirements

In an enterprise context, these sources of requirements are business-critical. An infringement can have serious consequences – from fines and damage to reputation to the loss of trade secrets and thus the foundation of a business.

Once the catalogue of requirements has been thought through, it must be communicated to all parties involved as the basis for the complete product and its architecture, implementation and deployment. Of course, it must also be ensured that it is adapted when the use of the product is extended to new areas.

4.3.2 Threat modelling

To be able to adequately protect the processed information, the technical solution must be armed against attacks. Threat modelling makes it possible to identify possible forms of attack to which the product is exposed.

If threat modelling is based on a systematic approach (such as STRIDE [14]), additional aspects such as traceability and reproducibility can be considered. It often makes sense to supplement and refine the basic approach with empirical values.

> The authors of this booklet believe that threat modelling offers the greatest benefit for the security of a product. However, its consistent implementation poses many challenges and requires a high degree of security awareness among all parties involved. Particularly in view of the diversity of activities in a large company, ensuring the consistent quality of threat models becomes a Herculean task. Consistent threat modelling can therefore be understood as a quality and maturity characteristic of an organisation.

[14] STRIDE is an acronym and stands for the 6 threat classes Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege; STRIDE was originally founded by Microsoft employees.

Results of a threat model

The profitable result of a threat model is not primarily the threats it works out. Although they are important for traceability, mitigations and measures derived from them as well as the test and detection scenarios are more valuable.

The choice of the mitigation to be applied shall be made in descending order of importance:

- Threats should be avoided by **adapting the architecture** if possible.
- If the architecture cannot be adapted, the technical mitigation should be implemented with a **standard solution** if possible.
- If no standard solution is available, a **custom-made solution** should at least be checked by a specialist.
- If a technical solution is also impossible, the (remaining) vulnerability must be **documented** and communicated as a **risk**.

Test and detection scenarios can be defined based on the selected mitigation.
All results obtained in the threat model influence the security activities in subsequent phases of the SDLC:

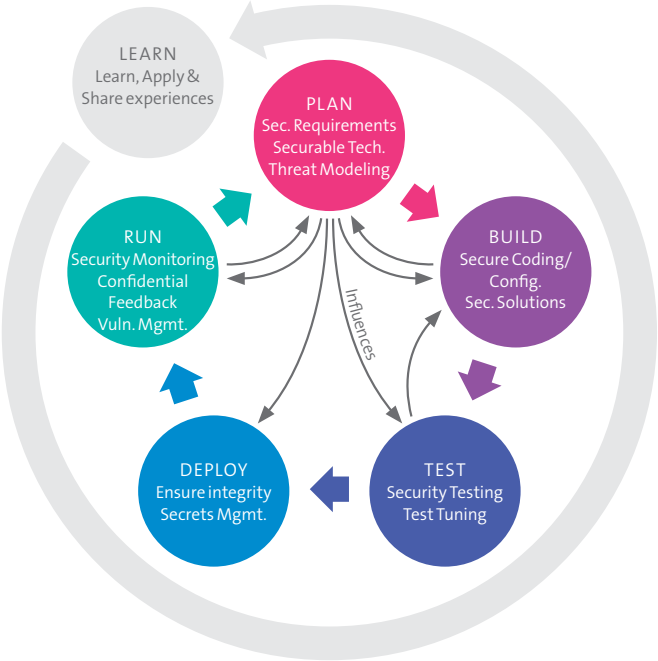

Fig. 7 | A threat model influences all other activities in a SDLC

- **Implementation:** the technical mitigations are implemented directly or via integration of a solution.
- **Test/Integration:** the implemented mitigations are tested automatically. Dedicated security scanning tools and services can be used for this. If necessary, they are aligned more specifically to the product with the findings from threat models.
- **Deployment:** threats that occur during software distribution, configuration, or with regard to confidential information such as passwords, private keys or tokens that are required during runtime, can be addressed and monitored early with appropriate technical mitigations.
- **Operations:** for attacks to be detected during runtime, two important prerequisites must be met. On the one hand, the identified detection scenarios must be planned and implemented. On the other hand, it must be ensured that the product aggregates the necessary information (like logs and other information sources) on a central platform.

Managing threat modelling documentation
In practice, there are several approaches to managing threat modelling documentation. They differ essentially in the way they are handled:

- With a **central** threat model, the entire team works on the same centrally stored documentation. When implementing individual features, the central threat model must be referenced here.
- With the **distributed** threat model, newly emerging threats and the associated mitigations become part of the corresponding work package (feature, ticket, user story, etc.). Therefore, threats are specified and implemented directly in that specific context.

Both approaches have specific advantages and disadvantages, depending on who works with them and at which level. They also provide a somewhat different view.

A centralised threat model allows quick orientation in a product context. This makes it easy to identify synergy potentials. However, it causes cognitive effort on the part of the DevOps Engineer when implementing individual features, user stories or tasks. He must then first select the points relevant to him from the whole. In addition, an isolated threat model that is detached from the actual product can quickly become asynchronous to the product. Therefore, it is often time-consuming to work with it and keep it up-to-date.

The distributed threat model is closer to the DevOps Engineer and isolates the relevant part for each feature. Because a ticket is already available as a working basis, the distributed model is easier to maintain. Conversely, additional effort is required to obtain a complete overview of the threats and mitigations relevant to a product. Then, all the individual pieces must first be assembled.

Depending on the point of view one wants to gain, both approaches cause additional effort. In order for the handling of the model, its maintenance and follow-up activities to take place in an orderly and easy manner, approaches should be described and mapped to each other in a simple and useful threat-modelling process.

DevOps and threat modelling

Threat modelling is critical to the security of a product throughout its lifecycle. However, clean integration into the development process is demanding. There are also different approaches for this.

Either way, threats must always be identified when planning an expansion to architecture. In order to do this, it is necessary for a person to be able to put himself in the shoes of an attacker. This is difficult because one must slip from one's systematic-constructive attitude (as developer, operator or DevOps Engineer) into the creative-destructive role of an attacker. This is often the biggest challenge when a DevOps team has to develop a threat model themselves.

Alternatively, you can have threats generated automatically by a tool. However, the results often lack the necessary specificity to the product context. Therefore, many of the generic threats are oftentimes considered inapplicable by the users of the tools. As a result, the results of the tool are generally analysed less precisely, and their effect is again lost.

Ultimately, the quality of a threat model benefits most when an experienced threat modeller supports the development team. However, this approach often leads to scaling problems. In the enterprise context, in particular, security experts often lack a technical or thematic deepness of understanding of the product. That is why bottlenecks arise here that have to be dealt with.

To close these gaps, a continuous and simple threat modelling process is necessary. It should create the necessary interfaces, for example to the following steps in the product product development lifecycle and ensure that those involved can work together without friction losses. For example, an experienced threat modeller with little knowledge of the product should be able to support a team that, in turn, has little knowledge of security threats.

> Because DevOps is an iterative concept, the architecture of a product can be changed with each new user story or product feature. Therefore, it is not practicable to create an initial threat model and reference it over a longer period of time without maintaining it.

> Instead, care should be taken to ensure that the model can be processed both as a whole and in the form of individual, specific sections. The former is important to get a holistic security view, and the latter to address threats accurately. The constant change of the threat model over the lifecycle of a product does not make its maintenance any easier.

4.4 Technical security activities

Central to this chapter		
People	**Process**	**Technology**
The team must know and understand areas of application of technical solutions	Discovered vulnerabilities need to be tracked and it needs to be clear who is responsible for them Technical support is required for integration, set-up and provision of security solutions	Security solutions should be easy to integrate

Technical security activities are those steps that are either directly technically feasible or which directly influence the implementation of a product.

4.4.1 Security solutions

With the adoption of the "Shift Left" principle, DevOps teams are not only entrusted with security topics – which are a mammoth task in themselves. There are many aspects to it, such as stability, change management, operational monitoring and so on. It must not be forgotten that all these topics, just like security, need to be reflected in the entire product life cycle.

So, the responsibility given to the team is enormous. If adequate support is lacking, mistakes with serious consequences can easily occur. This is where the central security organisation comes in. It can provide the necessary expert knowledge by providing easily, scalable and configurable security solutions.

Ready-to-use solutions
To maintain or increase the security level in the company, DevOps teams should be provided with a catalogue of ready-to-use solutions, patterns and reference implementations. Appropriate technical integrations for solutions can be offered and expanded via Inner Source (an intra-organisational open source approach). That way, teams can contribute and extend integrations if required.

In order to keep the solutions catalogue standardized and clear, content control of the catalogue should remain in the hands of security specialists. They control the portfolio based on security risks and user feedback. The goal must be to make the use and integration of solutions as simple and uncomplicated as possible.

> It has proven useful to centrally develop and operate generic security solutions from an organisational perspective by a separate DevOps team. As part of Swisscom's central security organisation, the DevSecOps Tooling Competence Centre takes over this task. The goal of such a set-up must be to bring the specific security expert knowledge into product deployment pipelines in as automated ways as possible.
>
> It should be noted that this team must also perform the usual DevOps tasks such as support and documentation. Correct integration into the target systems and availability of services is crucial, especially when offering Security Solutions.

In order to keep an overview of the catalogue and to identify relevant security solutions, a threat model can once again come to the rescue. This means that relevant security solutions can be referenced directly from the threat model instead of requirements mitigating certain threats of a product. This not only helps the DevOps team because they don't have to translate security requirements and develop their own solutions, but also the Security Coaches and Security Officers. They can then be sure that the centrally offered and professionally operated/maintained service is waterproof.

Some examples of security solutions
The list of security solutions that can be offered through a catalogue is long. Some examples based on the well-known STRIDE method for threat identification:

- **Authentication:** (multifactor) authentication as a service that can be integrated into an application
- **Integrity:** Public Key Infrastructure (PKI) as automated service to enable signing and encryption of artefacts and user data

- **Traceability:** logging solutions that allow for security monitoring
- **Confidentiality:** Libraries and services for encrypting and accessing sensitive information, such as operational secrets (passwords, SSH keys, certificates, etc.)
- **Availability:** Libraries and services that can detect and block attacks at the application and network levels
- **Authorisation:** Identity management solutions whose integration allows easy authorisation at the application level

4.4.2 Automated security testing

Automated verification of source code and products forms the core of DevOps. Automated tests make it possible to quickly detect and eliminate errors. In other words: the earlier a defect is discovered, the easier it can be remedied, and the less effort is required.

Security is no exception to this. On the contrary – completely automated deployment pipelines (including implicit security review and clearance) make it possible to make security aspects visible during implementation. The automation of security testing is therefore not only a quality check, but also an instrument for increasing security awareness. Some preliminary work is needed to make full use of all advantages. For example, security testing services must be provided that meet the needs of DevOps teams in the context of automation. These include, among other things:

- Simple and easy technical integration of the security testing solution into deployment pipelines
- No slowdown of the build (test performance)
- Minimal onboarding effort to ensure high-quality results

Anyone who follows these principles can improve the acceptance of security testing services. This in turn facilitates the enforcement of a set of required security services.

> It is repeatedly the case that individual DevOps team members attach less importance to security. Often, they are also critical of security tools and – sometimes – don't hesitate to let others know.
>
> In order to avoid such frustration being aimed at security, a suitable support point is needed. It's supposed to be a point of contact that helps with problems related to integration or the use of security tooling. Users can also reach out for security-related questions like detected vulnerabilities that are not well understood. At Swisscom, the DevSecOps Tooling Competence Centre again takes over this task. This multidisciplinary team expands and operates the security tools. At the same time, it is a contact point for users and also receives feedback, which in turn flows into existing or new solutions.

Requirements for security testing tools
At the heart of automation are the tools that perform the tests at the end. Some basic aspects need to be taken into account when selecting them:

- **Future viability:** Needs and requirements change quickly, especially in a DevOps organisation. Therefore, a tool must be flexibly integrable and addressable in different ways. This can be achieved if all work steps can be automated via APIs. Ideally, the security tool is developed according to an API-first approach.
- **User orientation:** The tool should satisfy the main user groups, which are:
  - The DevOps teams: They are technically on the move and therefore need to know their core technologies are supported. The tool is also required to be extensible as such to support further technical development.
  - Security Champions and Coaches: They help the teams to incorporate security into the product. Therefore, the tool should offer the possibility to aggregate defects in such a way that areas can be identified in which the team still needs support.
- **Multitenancy:** In large companies in particular, it is interesting to evaluate test results by business area. This may, for example, be due to the fact that areas are differently critical for the organisation. Alternatively, a tool may be required to provide an overview of the areas in which security investments should still be made. A tool should support such specific considerations.

In service-oriented environments in particular, many different technologies are often used on a daily basis. Their support in the market often varies greatly. Often, legacy services are maintained in outdated programming languages while more recent services are developed using techniques so new that hardly anyone knows about them. This diversity can be met in different ways. Either one tool or one service per technology or programming language is provided, or you will find a product on the market that covers the entire spectrum. Either way: In the enterprise environment, the 80-20 rule is to be followed here, controlled by the criticality of the products to be scanned.

Test scope
Automated security tests can be used to examine various aspects. In practice, at least the following cases can be distinguished:

- Self-developed **source code:** If a product is developed in-house, the source code can be checked. Vulnerabilities can thus be addressed directly and mitigated or removed.
- **Third-party libraries:** Almost all software is based at some point on software artefacts which stem at some point from a third party. With open source libraries, for example, any responsibility is often transferred to the user. All the more reason for him to be informed about all known vulnerabilities.

- **Infrastructure:** Every software bases on some kind of infrastructure. This foundation must be securely configured and kept up to date. Vulnerabilities such as unnecessarily exposed communication ports, unsafe cryptography algorithms or missing security headers must be avoided.

Test types

There are types of security tests that can and often should be used in a complementary manner. Even if the test scopes of individual types overlap, other vulnerabilities can still be found. The most common security testing types are described below. They can be more specific to the product under test, both for themselves and for each other, if information from the threat model is used.

- **Static Application Security Testing (SAST):** The static analysis examines the source code itself. Depending on the technology, keywords or combinations of them are searched for that are known to lead to insecure behaviour. Advanced analysis tools build graph models from source code, which are then examined for specific patterns. In general, SAST solutions are efficient because they provide a very deep insight into the source code. Their major disadvantage, however, is their susceptibility to false positives, which often have to be processed manually. Furthermore, SAST tools often lag behind the further development of programming languages and frameworks. Since SAST does not take the application context into account, logical errors are often not detected.
- **Dynamic Application Security Testing (DAST):** With dynamic analysis, the software is executed in a protected environment and called up with security-relevant inputs. Inputs can either be formatted sensibly or generated randomly (so-called fuzzing). Depending on how the application behaves or what it returns, vulnerabilities can be identified. DAST tools generate fewer false positives. Because the application context runs along with it, they are also better suited for detecting logical errors. Due to the limited insight into the software, however, only directly exposed vulnerabilities are discovered. Of course, the tools can be adapted to the target applications to a certain extent. However, it remains difficult to realistically simulate a creative attacker.
- **Interactive Application Security Testing (IAST):** The “Interactive” in IAST refers to the interaction between the two approaches already presented. This is an attempt to bring together the positive aspects of SAST and DAST. For example, IAST allows you to estimate how a certain input is processed in a data flow. This reduces the proportion of false positives. However, IAST solutions are often very technology-specific.

All three concepts are dedicated application security solutions.

When a SAST solution was introduced as a self-service for the teams, Swisscom was able to gain the following valuable experience: The DevOps engineers, often with little know-how about SAST and security, had some of their complete source code scanned, including test directories, fixture data, etc. This meant that the individual scan took a lot of time (several hours) and therefore could not be integrated into automatic build processes. Consequently, these scans were only performed manually and sporadically.
The unlimited "tipping in" of code also presented users with a huge number of defects – in most cases false positives. As a result, the interpretation of the scan results became a proverbial search for the needle in the haystack.

Accordingly, acceptance of the new SAST service fell massively. Because of the negative experiences, many users developed a real aversion to the service and refused to continue using it.

The most important finding from this situation is that a SAST tool cannot do without specific information about the target object (context). In order to get a grip on this, two primary positions were used:

- The complete onboarding process has been redesigned.
- The target audience was readjusted, primarily to be able to restart without reservations.

In addition, the service was relaunched under a different name. Thus, the negative image could be cast off and restarted with the intrinsically good tool.

Today, the service is provided by a specialised SAST team with the appropriate know-how. DevOps teams that newly register their product for the SAST service go through three steps of onboarding:

- In an initial threat model, data flows are analysed, and irrelevant code parts are identified.
- The SAST team then tailors the scan specifically to the product to eliminate false positives, detect false negatives and optimise throughput time.
- After the first scan, the defects found are discussed in a fact-finding meeting. There the teams also receive support, which makes it easier for them to repair the defects and avoid them in the future.

These changes to the process reduced the number of false positives by almost 95% of false positives. The scan throughput time has been reduced from several hours to a few minutes.

> With the short running time and the high relevance of the findings, the scans can now be automated. All in all, this has led to a noticeable improvement in user acceptance.

In principle, security aspects can also be tested without a dedicated security test environment. For example, input validation can be covered directly by automated **unit tests**.

It is obvious that the number of vulnerabilities introduced by third party libraries should be kept as small as possible. To achieve this, the code can first be searched with a kind of a SAST approach for embedded external dependencies. With the results obtained, the so-called bill of material, relevant vulnerabilities can then be identified via the correlation of CPE (Common Platform Enumeration) and CVE (Common Vulnerability Enumeration) identifiers.

Another approach, which can be used in addition, executes the scan only at the end of the deployment pipeline on the assembled artefact. To get information about included third party libraries, the artefact can be unpacked. This is a different approach to obtain a bill of material, representing an inventory of software components and versions the artefact contains, which can be used to correlate the vulnerabilities found. The inventory can be persisted for that artefact, so that vulnerability correlations are also possible at a later point in time. This process is described *in 4.6.1 under Security Monitoring*. It is called Artefact Vulnerability Management (AVM) or sometimes also Software Component Analysis (SCA).

Various tools are used to test the **infrastructure** for secure configuration (sometimes also called "hardening"). These include nmap[15], nessus[16], sslscan[17] and the like. These tools can be automated either individually, via command line integration or as part of established test frameworks. Such frameworks can, for example, be orchestrated with the abstraction language "Gherkin"[18], which has established itself in end-to-end testing. Gherkin is a syntax used in Behaviour Driven Development (BDD). It should enable non-technicians to read and write test cases by formulating scenarios using prose text *(see Fig. 8 | Example of a test case in Gherkin syntax)*.

In principle, the infrastructure test cases for this approach should be derived directly from the threat model or even generated automatically. In the community, this is usually referred to as Security as Code. Of course, this fits nicely into the DevOps requirement "Everything as Code".

[15] Nmap by Gordon Fyodor Lyon, URL: *nmap.org*

[16] Nessus (Software). Wikipedia, URL: *en.wikipedia.org/wiki/Nessus_(Software)*

[17] sslscan, rbsec, URL: *github.com/rbsec/sslscan*

[18] Gherkin Syntax, Cucumber Community, URL: *docs.cucumber.io/gherkin*

```
Feature: Google Searching
      As a web surfer, I want to search Google, so that I can
      learn new things.

Scenario: Simple Google search
      Given a web browser is on the Google page
      When the search phrase "panda" is entered
      Then results for "panda" are shown
```

Fig. 8 | Example of a test case in Gherkin syntax *(Source: automationpanda.com)*

There are several automated security testing methods. It is important to understand that each of them looks at the application from a different perspective. A single method therefore provides only an inadequate view of the application. Optimum security is achieved by combining various automatic testing methods and concluding with penetration testing. Then you can be sure that the creativity of an attacker will also influence the result.

4.4.3 Manual security testing

Many technical security tests can be automated. However, their quality depends on how well the automation technology performs and how well they are maintained. Thus, the results of automated tests provide a good picture of basic security. However, they cannot represent the creativity of an attacker who is able to put information into context and to interpret the resulting picture.

Penetration testing

Therefore, automated tests cannot replace periodic technical penetration tests. It does not matter whether the tests are white, grey or black box tests[19] carried out by trained specialists. However, it is optimal if the software has already been tested automatically before the penetration test and the vulnerabilities found have been eliminated. This will allow testers to better focus on complex vulnerabilities and less be distracted by emerging «simple» vulnerabilities.

[19] For white-box testing, the tester receives source code and configuration of a product to be tested. For grey box testing, only part of the information is disclosed. For black box testing, the tester has no prior information available.

Coordinated audits
Periodic audits are primarily useful for critical and exposed products. True to the DevOps approach, a dedicated product environment can be instantiated for penetration testing. The testers should be provided with as much product information as possible. This enables them to try out or exclude even advanced attack scenarios from the outset. The aim of the action should be to achieve effective results as efficiently as possible (corresponds to a white/grey box test). In this set-up, defensive attitudes do not help to realistically test and assess product quality.

Red Teaming
While penetration testing is about finding the vulnerabilities of a particular application, Red Teaming [20] focuses on exploiting a vulnerability to penetrate a corporate network as widely and unobtrusively as possible. Red Teams proceed under realistic conditions. In other words, they have little internal information at their disposal and develop it in the course of their action.

The Red Team in the attacker role is the opponent of the Blue Team in the defender role. The Blue Team describes all defensive and reactive security measures mobilised by a company. In the broadest sense, in addition to security teams participating in attack response *(see chapter 4.6)*, DevOps teams are also part of the Blue Team.

The concept of Red Teaming in itself remains unchanged, even in a DevOps organisation. The differences lie primarily in the potential response time and in the administrative impact of an attack. Since with DevOps a product is developed and operated simultaneously by the same team, anomalies in data traffic, for example, can be tackled much more quickly and easily *(see chapter 4.6)*.

Concluding Red Team publications are often entertaining and easy to read "War Stories", which nevertheless offer a lot of technical content. They allow the DevOps teams to directly build a reference and make comparisons to their own product. In terms of security awareness, such a reconditioning is therefore very efficient and valuable. Of course, systematic measures can also be derived from the Red Team actions.

> Red Teaming has been in operation at Swisscom since 2015. Each year one to two attack simulations are carried out by a changing team. The spectrum of attacks ranges from social engineering and phishing to active takeovers of Swisscom systems using malware. All this happens naturally, without causing actual damage. The aim of these attacks is, on the one hand, to identify

[20] Red team, Wikipedia: URL: *en.wikipedia.org/wiki/Red_team*

weaknesses and risks before real hackers exploit them. On the other hand, the Blue Team should be confronted with realistic scenarios in order to identify weaknesses in capabilities and processes.

As it turned out, the biggest challenge is to get the Red Team's active employees out of their daily business so that they can take the necessary time to hack. Since the Blue Team is also expanding its detection skills and constantly learning new ones, the task for the Red Team becomes more and more difficult over time.
As described, internal processing and communication of the so-called Red Team Cases has proven to be very valuable. Employees in all business areas and at all levels feel specifically addressed by the prepared stories and reports. This in turn leads to adjustments in their environment, which benefit security.

4.5 Deployment pipeline security

Central to this chapter		
People	**Process**	**Technology**
The team knows the threat situation regarding the central Deployment Pipeline systems (i.e. the "factory") There is an awareness of the risks associated with own assets and those of partner teams using the same platform	There is a separation of powers on a need-to-know basis	The Deployment Pipeline supports modular extension with components for quality inspection Central systems are adequately secured

Because one of the core capabilities of DevOps organisations is automation, the integrity and confidentiality of automation services must always be maintained. The following systems can be counted as automation services:

- **Workplace** – the system where code is written
- **Version Control System (VCS)** – where the code is stored and versioned
- **Continuous Integration/Continuous Deployment (CI/CD) System** – where code is processed, and artefacts are assembled
- **Scanning Services** – code and artefacts are tested
- **Artefact Repositories** – here, files for software packages and more are stored
- **Runtime environments** – are required for value creation by actually running the software product

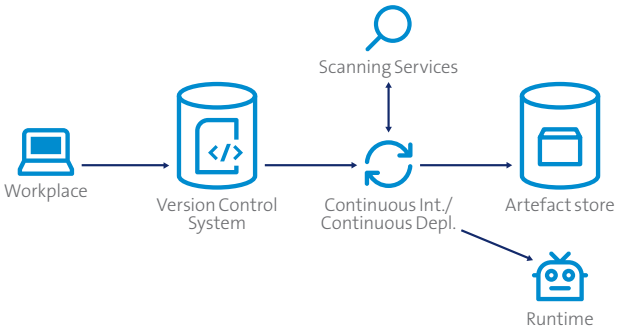

Fig. 9 | Abstract representation of a deployment pipeline with the various components that need to be saved.

This chapter deals mainly with interaction with external partners. This is important because the increasingly central role of the deployment pipeline enables interfaces to partner systems and thus new attack vectors.

It is part of the DevOps principle that the software and its configuration are stored separately. In other words: in addition to the software, its configuration must also be integrated into the production environment. Because configurations often contain sensitive parameters that are used for data exchange with third-party systems, this communication must also be secured. In most cases, this is done using certificates and authentication secrets. With DevOps, new requirements arise here.

The influence of "Everything as Code"

The maxim "Everything as Code" means for DevOps organisations that all tests, builds, repositories and deployments are described by code. This has enormous implications: an accidental adjustment at the wrong place can cause tests to fail. In the worst case, a malicious change to the code can exfiltrate secrets (like passwords, keys, etc. or even of a business nature). Reading access to a codebase alone can provide attackers with a comprehensive view of the product being operated. This makes it much easier for them to search for weak points in a software product and weaponize for an upcoming attack.

The concentration of logic in the code eliminates segmentation and thus manual control mechanisms. In a transformation phase towards DevOps, it is particularly important to replace quality controls with automated measures. They are stored, again as code, specifically adapted to the product. This creates a kind of logical redundancy between the code and tests that helps to preserve integrity.

As a result of the translation of (almost) all aspects into code, the value or criticality of the code repository, i.e. the Version Control System (VCS), multiplies. If this system is compromised, it has far-reaching consequences.

Another critical component is the Artefact Repository. Here the challenges are comparable to those of VCS. If artefacts can be modified or even compromised, entire production platforms may be at risk.

For both systems, maintaining the integrity of managed assets is most important. Unwanted changes to source code or artefacts can lead to fatal problems in production environments. The second most important thing is confidentiality. The leakage of data from a VCS or artefact repository can lead to the loss of secrets and other internal information.

Of course, there are other threats related to "Everything as Code". However, these are rather specific for the technologies or processes used in each case.

Integrity and confidentiality of code and artefacts
To protect the integrity of source code, Git-based [21] VCS should enforce commit signing. Technically, commits with unknown signatures must be rejected. This is the only way to ensure that only authorised users can make code changes.

For artefacts, the integrity should be detectable with at least one checksum, i.e. a digit sum across the entire artefact. A better solution, however, is to enforce a signature on artefacts, too.

To protect confidentiality of internal information and secrets, technology is not the only approach. For example, it is necessary to control access with suitable Identity & Access Management (IAM) and, important, adequate processes. This ensures, for example, that only the assets that are needed can be accessed. Excessive accesses should be detected via monitoring and slowed down via throttling/rate limiting. Countermeasures can be IP or account blocking to prevent possible attacks *(more in chapter 4.6)*.

[21] Git has become the de-facto standard of versioning technologies for source code, Git, Git Community, URL: *git-scm.com*

But also the authors of code or system specifications are stressed. A consistently enforced clean coding specification can reduce the risk of leakage of operational secrets (certificates, keys, passwords, etc.). Secrets have no reason to be kept in a VCS, because even after removing them from the finished code, they can still be found in old versions (unless thoroughly removing a lot of information from the repository). The same applies to secrets in artefacts.

Algorithms should also be protected if they are commercially viable innovations. They should be regarded as assets. Completely segregated VCS are often justified in such cases.

Cooperation with partners and providers
Value-adding data or systems are primarily valuable to the owner. Because partners and suppliers have less connection to these assets, they are often not particularly willing to implement additional security measures. All the more reason for the need for contractually regulated security precautions vis-à-vis all partners. On this basis, processes can be audited, for example, or it can be checked whether agreements are being adhered to.

Because personnel changes can occur quite frequently and at short notice, the IAM process used should enable fast onboarding. "Offboarding" should also not be neglected, for example by ensuring that individual employees can be quickly denied access if necessary.

In a DevOps context, it is very advantageous if the good practices used are checked technically and fully automatically. For example, unsigned commits from external supplier staff can be rejected. The same applies if suppliers push commits from employees who have not registered with the customer. If one takes this concept a little further, automatically detected security defects can also be returned directly to suppliers, as it is possible to clearly identify who's introduced what defect in the common source code base.

All in all, the examples show that cooperation with partners and suppliers must be well thought out. The processes should allow errors to be corrected quickly. The contact persons on both sides should be able to talk to each other on a technical level in order to avoid misunderstandings, especially when it comes to security-related aspects of products.

Access to productive data and systems
The ultimate goal of a DevOps organisation should be to map its complete infrastructures as code. That way, adjustments on runtime systems become an anti-pattern. Access to productive systems is then only possible in exceptional cases, for example to understand specific incidents and circumstances. Actually, such accesses would not be necessary either, because problem cases should be traced and evaluated via clean logging or at least reproduced in a non-productive environment.

Actions taken in the course of an extraordinary access to live systems can be recorded for traceability. In the optimal case, a single system node would be directly replaced by a new instance right after manual access in order to disfunctions due to unwanted manual changes on single nodes.

From a security perspective, the benefits of an Infrastructure-as-Code approach are great – to name just a few aspects:

- **System hardening** (secure configuration) can be verified by code analysis
- **Adjustments** to the system hardening can be clearly traced
- **Instantiation** of dedicated test environments identical to the production environment is possible without additional effort, e.g. for penetration testing
- Systems that have potentially been weakened by manual actions can be **replaced** immediately (also: Immutable Infrastructure)
- **Production data** is better protected because there is usually no manual access to productive systems

A functioning IAM is also a prerequisite for the protection of data and systems. If identities cannot be strictly assigned, accesses can no longer be assigned to trustworthy persons. This also means that the traceability of actions can no longer be guaranteed. Another challenge arises when it is not human users who access services, but technical accounts.

Management of secrets and certificates
Inevitably associated with DevOps is the issue of automated secrets management necessary for systems to access other systems (e.g. databases or other services). It has always been best practice to replace secrets such as passwords from time to time. This reduces the risk that attackers can gain access to an old password and thus illegally and anonymously view or even change data.

In an organisation with enterprise dimensions, many systems are accessed from automated processes. Therefore, a lot of effort was required if system administrators had to periodically change all passwords manually. The manual password change could also trigger operational problems whenever an accessing system still uses an old password and suddenly no longer obtains the necessary access grant. This can happen if a password is hardcoded in source code. This gives rise to further problems. Including, for example, how to proceed if a system administrator who probably knows one or more service account passwords leaves the company.

Another point in favour of automatic secret management is the shortened technical lifecycle of runtime instances in a DevOps environment. By this point at the latest, manual solutions definitely become impracticable. Secrets must then be rotated independently of the product version and injected to the runtime setup. The same applies to (web) certificates and their respective counterparts, the private key.

Anyone who approaches these topics with a degree of automation worthy of DevOps and nevertheless wants to guarantee an appropriate level of security will find various Secrets Management solutions on the market. Some of them are available both as commercial products and in the form of open source.

For the certificates, it is necessary to integrate a Public Key Infrastructure (PKI) that fits the automation requirements. Here you must decide whether a free service is to be used or whether the organisation is to set up its own PKI. In the latter case, the automation can be addressed with the de facto standard Automated Certificate Management Environment (ACME).

4.6 Productive operations and attack response

Central to this chapter		
People	**Process**	**Technology**
Communication and preparation of threat scenarios	Manage monitoring and response processes Ensure an organisation that ensures responsiveness	Use of solutions to detect attacks

Security incidents happen – with or without DevOps. This unpleasant truth puts all preventive security measures in a different light. It also shows that there is not only one (preventive) or the other (reactive) side of security. Rather, the challenges extend over the entire product life cycle.

In larger organisations, there are always areas that are a little less in focus with regard to the implementation of security measures. They each have to get by with less support than other areas. From this perspective, it is important to have simple interfaces and procedures for collecting security-relevant logs and feedback. From the DevOps team's point of view, the processing and evaluation of data and feedback should also be as simple and comprehensible as possible.
In an efficient product development cycle, all types of feedback should be reduced to a core statement so that they can flow back into the preventive part of the SDLC. For example, you can add hints for threats that have not been considered to a threat catalogue, which can be used again during planning activities in threat modelling. In this way, the threat catalogue and model are iteratively expanded with the aim of obtaining a picture as complete as possible.

User feedback

One of the principles of DevOps is to collect implicit (measurable) as well as explicit feedback from the users. Security-relevant feedback on a product, usually, must be treated confidentially. It may also only be made accessible to a restricted circle of persons. Otherwise, there is a risk that potential attackers could gain knowledge about existing vulnerabilities and harm the company.

In order to prevent security assessments from being read in forums or social media, legal and confidential channels must first be created through which vulnerabilities can be reported. In addition, incentives must be created to encourage external security researchers to give feedback at all. Incentives can also help to maintain the quality-bar of feedback high.

Vulnerability management

In practice, it may be the case that new vulnerabilities are published for individual components during their productive use as part of a product. The information can come from sources inside or outside the organisation *(see 4.4.2)*, such as the Common Vulnerability Enumeration (CVE). In order to be able to react quickly, such sources need to be constantly monitored. The general procedure for dealing with vulnerabilities is formally described by ISO Standard 30111 on "Vulnerability Handling Processes".

DevOps enables new approaches here. On the one hand, it simplifies the exchange of affected components, since the complete application structure is described by source code. On the other hand, fully automated tests help to quickly decide whether the functionality of a product is still guaranteed after replacing a vulnerable component. Both aspects help in rolling out new product versions quickly and with confidence that availability is not degraded. For the entire vulnerability management process to run smoothly, a precise inventory of products, software and included components must be available from which it can be deduced which product instances are affected by vulnerabilities.

> One way of dealing with vulnerabilities is to combine centralised and decentralised organisational approaches: the DevOps teams are basically responsible for the weak points and their elimination (decentralised). They receive support from the central security organisation in the form of specialised expertise. Security specialists can help in some kind of task force structure, their support in eliminating critical vulnerabilities and thus ensure short response times.

Bug bounty
An increasingly popular way to get feedback on security is to use bug bounty programmes. They combine the confidential feedback channel with vulnerability management by creating another source of information on vulnerabilities. In doing so, they implement ISO standard 29174 on "Vulnerability Disclosure".

Bug bounty programmes work roughly like this [22]: A user reports a vulnerability he found in a product to the operating company, including traceable documentation. If the vulnerability is confirmed, he will receive a reward if he commits himself to secrecy for a jointly agreed period of time. The company uses this time to fix the vulnerability. This procedure is also known among experts as the "Responsible Disclosure Model".
In DevOps organisations of large companies, the underlying agility of a company creates ongoing relevant security challenges. Therefore, the bug bounty programme is best run from a central location. The prerequisite, however, is that the operators know the entire attack surface of the organisation. This is given by the sum of the exposed products.

In order to process bug bounty reports efficiently, you need a complete and up-to-date directory with the necessary information about products, associated roles and persons *(see also: "Alarm – and then?" under 4.6.1)*. Keeping such a directory up-to-date, however, is time-consuming because the products, responsibilities and contact persons are constantly changing. In addition, it is recommended that the technical scope of the bug bounty programme is precisely defined and documented for the general public. Documentation of this type then states exactly which applications and services a company exposes to the Internet and for which of them bug bounty reports are accepted for. The security researchers may then only search for vulnerabilities on these.

To cover the entire attack surface, the focus should also be expanded to include the internal actors of the organisation. Mature organisations also provide internal users with a confidential communication channel for security-relevant feedback.

> Swisscom was one of the first companies in Switzerland to offer a bug bounty programme [23]. Since its introduction in 2015, the channel has proven its worth in numerous cases. Vulnerabilities relevant to the general public are published on the portal as "Security Advisories" and receive a CVE number (Common Vulnerability Enumeration [24]) for unambiguous identification.

[22] The terms and conditions of bug bounty programmes may vary from company to company. The definition given here corresponds to the general understanding.

[23] Bug Bounty, Swisscom Ltd, URL: *swisscom.ch*

[24] Common Vulnerabilities and Exposures, The MITRE Corporation, URL: *cve.mitre.org*

> Swisscom has covered a learning distance in setting up the bug bounty programme. For example, it was found that the security researchers had to be supervised by equally specialised representatives of the internal security organisation. This promotes good communication and ensures adequate responses to the reports received. Since then, the relationship between the security researchers and Swisscom has developed in such a way that there are even joint public appearances.

4.6.1 Security monitoring

Security monitoring is generally understood to mean the monitoring of security-relevant events. It is often forgotten that appropriate use or abuse case specifications are required for a meaningful and product-specific evaluation. They describe what can go wrong in a case of emergency and what information is needed to detect an attack. Although log data and other sources for security event data can be evaluated generically, they often deliver far too few specific results or even false positives.

Another component that needs to be monitored is the ecosystem in which a company or product operates. However, when doing so, different levels of abstraction must be taken into account.
Effective monitoring in a DevOps organisation also means that, in the event of a security incident, all contact persons are defined from the outset. If this is the case, the DevOps setup can play out its advantages: Because the communication channels within the teams are short, reactive mitigations can be implemented efficiently.

Because orientation is particularly difficult in stressful situations, i.e. in emergencies or crises, all scenarios and emergency plans must be practised in preparation. This reveals weaknesses or even errors in the choreography of the parties involved (DevOps team, participating security response teams, etc.).
It is in the nature of security monitoring that incidents only become visible during or after their occurrence. Therefore, an efficient DevSecOps organisation attaches importance to a well-functioning communication between reactive and preventive actors. This makes it possible to quickly translate the findings from the incidents into preventive measures.

Detection of known risks
One of the sources for security monitoring use cases is the threat model for the product. Both mitigated and unmitigated threats can be formulated as monitoring use cases. This allows the validation of the threat model on the one hand and the detection of security relevant events on the other hand. The latter should be weighted much higher, since an event becomes an alert and then an incident under the given conditions.

To be able to perform effective evaluations, there should be a "backwards calculation" from the desired result, i.e. the monitoring use cases. This gives rise to the following questions:

- Which components in the product infrastructure must deliver data? These may include: the Intrusion Detection System (IDS), the Reverse Proxy incl. Load Balancer and Web Application Firewall (WAF), but also the infrastructure underneath the application, like data storage, etc.
- Which data is required? Can they be accessed at all or are shared components in use?
- Which data does not bring any added value? A meaningful "signal-to-noise ratio" can improve the quality of the end result.

From the point of view of security monitoring, it usually does not make sense to collect highly detailed and verbose logs from all systems. Instead, you should save a little more log data than you currently need, deviated from your security monitoring use cases. A good start is to write logs that make transactions traceable across all components of a system using a common identifier. For example, the identifier can be a random number generated by the first component that "sees" an access.

Detection of anomalies

The shift-left approach creates exciting synergy potentials, especially in security. Thus, security solutions such as Runtime Application Self Protection (RASP) frameworks that could not be used efficiently due to personnel or organisational barriers can suddenly become interesting.

Such self-defence frameworks are often found in the web environment. There they are supposed to recognise and fend off attacks independently due to atypically changing attributes. For example, an application can sound an alarm if it detects access to a honey endpoint [27]. In another case, it may be suspicious if the agent identification suddenly changes during a running session.

Of course, the fine-tuning of such RASP solutions depends heavily on the application code. It is therefore worthwhile to attach great importance to their correct technical integration and the corresponding automated tests. RASP solutions can also be used as an additional source for security monitoring.

Alarm – and then?

If a (monitored and confirmed) security incident occurs, the previously implemented countermeasures are ideally triggered immediately and automatically. In business-critical products, however, such automation quickly reaches its limits because false positives cannot be ruled out. Then it helps if manual and/or technical countermeasures can be taken quickly thanks to the proximity to the DevOps team.

[25] Honey endpoint: an endpoint that is not accessed during normal operation and whose purpose is to identify attackers who scan the entire attack surface of a web application.

One possible manual countermeasure is to adapt the Web Application Firewall (WAF) so that it allows hot patching. At best, a vulnerability can be shielded via WAF, and the DevOps team gains the necessary time to cleanly correct the product itself.
A technical countermeasure would be, for example, the inclusion of a vulnerability exploit in the automated test set of a product. This allows continuous testing of whether the vulnerability is (re)occurring due to code changes. Without such regression tests, the awareness gained in connection with an incident usually goes up in smoke after some time.

A systematic inventory within the DevOps organisation is required to ensure that all necessary information for such countermeasures is known in an emergency. The inventory should contain the following information, which is permanently available to the Security Operations Centre (SOC):

- List of members of the DevOps teams (technical contact persons)
- Product responsibilities (business contacts)
- Technologies (may contribute to faster mitigation)
- Product inventory (endpoints, test results etc.)
- Further information sources (VCS, Threat Model etc.)

When a routine analysis escalates into an incident, the Computer Security Incident Response Team (CSIRT) takes action. It benefits from the fact that DevOps principles basically supports forensic investigations into the causes of an incident and the damage done. Thanks to the high level of automation and the accompanying logging of executed steps, it is always possible to trace where, what and when changes were made to the product, where it was exposed to the attackers and thus what made the attack possible at all.

In order to avoid destroying data in the hectic rush that could have provided information about the attack, those involved must be informed and trained right from the start. Basically, SOC and CSIRT are provided as internal, central specialised security services. They complement and support the DevOps teams primarily through professional response measures.

> Swisscom's Security Champions programme in particular has proven its worth in the processing of security incidents. It leads on the one hand to the fact that during the response to a security incident, the contact person on the side of the DevOps team is already known from the outset. On the other hand, the Security Champion has already built up a basic understanding of security terminology through the training and awareness units he has completed. This reduces friction in the interaction between CSIRT and DevOps team and also helps in general communication.

5 Summary

Efficient security in a DevOps organisation is primarily characterised by the strong integration of the various security activities. The better the individual steps are linked, the more effectively security can be implemented. DevOps as an approach helps to create the necessary conditions here. It tears down disturbing organisational barriers (for example between development and operations) and brings together what actually belongs together. This encourages a holistic view of the entire product life cycle.

The challenge, however, lies in sufficiently prioritising security as a topic. Because the DevOps teams will never reach the same expertise as security professionals, the central, supportive security organisation remains important for success. The motto must be: "Make it easy to do the right thing". In practice, this means: It should offer content, processes and services in such a way that the simplest variant is also the most secure one.

Anyone who does not clearly set the priorities during a DevOps transformation risks losing attention to security. To avoid this, the following points must be actively addressed:

- **General awareness:** with a training programme that provides the necessary visibility of security topics.
- **Intrinsic security:** by making security as simple and accessible as possible, for example with a set of solutions that can be adopted as-is → The elimination of process tollgates can be compensated for by creating transparency through continuously testing for security.
- **Identity & Access Management:** Mapping of all DevOps roles to ensure segregation of authorisations and traceability of accesses → A classic separation of powers is omitted, but the value of IAM as sovereignty over fine-grained access authorisations is increasing.
- **Organisation:** Security, with all its aspects and roles, should be anchored in the DevOps organisation. To make it scalable, responsibilities must be transferred across the board, i.e. to the teams. → However, it must always be clear who is responsible for risks and vulnerabilities, and the right people must be found and addressed at all levels.

If the right measures are taken, DevOps and Security do not stand in the way of each other but open up new possibilities for each other. In this way it can be possible to make the world a somewhat safer and more predictable place, at least from a technical point of view.

Swisscom has been implementing and applying these best practices since mid-2016. As usual in agile environments, this is a constantly changing process. It's all «Do, Learn and Adapt».

So far, challenges have emerged, especially in the fields of automation and culture. In the future, it is planned to further improve the maturity on all three pillars People, Processes & Technology and thus reduce the level difference between pioneer and laggard teams.

It is also necessary to improve measurability in all areas in order to be able to make data-driven decisions. One example is the combination of numbers on security vulnerabilities and the security training certification level of a team. In addition, data should be presented in a simple way and be available quickly if required. In order to achieve the desired transparency, it is also necessary to summarise relevant key figures on live dashboards.

6 Abbreviations

ACME	Automated Certificate Management Environment
AVM	Artefact Vulnerability Management
BDD	Behaviour Driven Development
CALMS	Culture, Automation, Lean, Measurement, Sharing
CI/CD	Continuous Integration and Continuous Delivery or Deployment
CPE	Common Platform Enumeration
CSIRT	Computer Security Incident Response Team
CSSLP	Certified Secure Software Lifecycle Professional
CVE	Common Vulnerability Enumeration
DAST	Dynamic Application Security Testing
DevOps	Artificial term, composed of Development (Dev) and Operations (Ops); stands for the complete product life cycle
DevSecOps	See DevOps; Security (Sec) has been added to give more weight to the issue
IAM	Identity & access management
IAST	Interactive Application Security Testing
IDS	Intrusion Detection System
OWASP	Open Web Application Security Project
PKI	Public Key Infrastructure
RASP	Runtime Application Self-Protection

SAFe	Scaled Agile Framework
SAST	Static Application Security Testing
SDLC	Software Development Lifecycle
SOC	Security Operations Centre
STRIDE	Stands for the six threat classes: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege
VCS	Version Control System
VUCA	Volatility, Uncertainty, Complexity, Ambiguity
WAF	Web Application Firewall

7 Index

8 Further resources

The following is a list of resources we recommend to readers interested in knowing more on different topics of DevSecOps:

- A Leader's Framework for Decision Making by David J. Snowden and Mary E. Boone, URL: *hbr.org*
- Accelerate: Building and Scaling High Performing Technology Organizations by Nicole Forsgren Phd, Jez Humble, Gene Kim
- Agile Application Security by Laura Bell, Michael Brunton-Spall, Rich Smith, Jim Bird
- Building Microservices by Sam Newman, URL: *samnewman.io/books/building_microservices*
- DevOps Culture (Part 1) by John Willis, URL: *itrevolution.com/devops-culture-part-1*
- The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations by Gene Kim, John Willis, Patrick Debois, Jez Humble, URL: *itrevolution.com/book/the-devops-handbook*
- Drive by Daniel Pink. URL: *danpink.com/drive*
- Manifesto for Agile Software Development by Kent Beck et coll., URL: *agilemanifesto.org*
- The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win by Gene Kim, Kevin Behr, Georg Spafford, URL: *itrevolution.com/book/the-phoenix-project*
- Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework by Mik Kersten
- Securing DevOps – Security in the cloud by Julien Vehent
- Security Champions Playbook/OWASP Wiki, URL: *owasp.org*
- State Of DevOps Report by Puppet + Splunk, URL: *puppet.com*
- Threat Modeling – Designing for Security by Adam Shostack

Notes

Swisscom Ltd
Alte Tiefenaustrasse 6
3048 Worblaufen