DevSecOps

Introduction & Implementation

Armin Reiter
arminreiter.com
Version: 1.2
05/11/2020

Table of Contents

1 Introduction

This paper gives an overview of DevSecOps and describes how to implement this new methodology. It starts with an explanation of DevOps and how it developed. DevOps is then extended by security which leads to DevSecOps. Chapter 4 outlines how security can be integrated into software development by using the common model of the Security Development Lifecycle (SDL). Those first four chapters create the theoretical background to implement DevSecOps and they give some background information about secure software development.

After the theoretical part, chapter 5 shows a practical example how a security breach in a company could occur. It quickly shows the benefits of DevSecOps and how it can be useful in such a case.

Chapter 6 is then focusing on how to implement DevSecOps in an existing company. It consists of clear sprints and tasks which can be used to secure the software development lifecycle in the company and brings the whole security process of the developed applications to the next level.

Last but not least the paper finishes with further information and tools that help in the DevSecOps world.

2 DevOps

The term DevOps is made up of the words Development and Operations, where development not only consist of software development, but also of other aspects in software engineering such as testing and quality assurance. The operations part includes all system administration tasks, like the deployment of the software, the production environment and the operation and maintenance of all associated components such as databases or network infrastructures.

Furthermore, DevOps describes practices that helps to improve the collaboration between IT operations and software development. This includes not only technical aspects like the automation of software delivery or quality assurance, but also organizational aspects like the definition of common goals and the creation of a cooperated culture for the mutual support of IT and development. The aim is to ensure that software of higher quality can be delivered more quickly. These points can be illustrated using the following four areas (CAMS):

- **Culture**: Focus on people and cooperation. People are more important than processes and tools.
- **Automation**: Automation is essential for DevOps to get feedback as fast as possible. Tools for Release Management, Configuration Management, Systems Integration, Testing, Monitoring, Orchestration and many more are used.
- **Measurement**: Continuous measurement of results and quality. Without measurable data, improvements are difficult to achieve.
- **Sharing**: Sharing knowledge, ideas, processes and tools.

DevOps contains patterns for collaboration, processes and tools and is therefore more than just a methodology or framework. It is more comparable to Scrum. The framework has become extremely popular in recent years, but there is also the opinion that "NoOps" is the future. With NoOps all operation issues should be automated, so that they are no longer needed. This could be achieved by using cloud technologies like PaaS or Iaas

2.1 Why DevOps?

DevOps has evolved for several reasons over time. In the past, there were conflicts of interest between software development and operations. These were caused by the separation of those departments which have their own goals and processes. All optimization efforts took place only in the departments itself and not in the overall organization. For example, software development wants to deliver new features quickly (we want change!), whereas operations want a production environment that is as stable as possible with at least changes as possible (we want stability!), because every change can potentially lead to problems. The creation of a common culture helps to bring those two views to a common goal.

DevOps is strongly influenced by agile thinking and tries to integrate agile principles. In a classical structure it can happen that a "hero cult" develops, e.g. when a developer creates an application but does not document it sufficiently. In this cult, this one developer is suddenly a kind of "hero", because he is the only one who has the knowledge for deployment and maintenance. DevOps with the use of agile principles is a countermeasure to address such problems.

As known from agile software development, DevOps uses an iterative approach with continuous improvement of the process and the tools. Exactly this iterative approach can also be found in the stages of DevOps.

2.2 Stages of DevOps

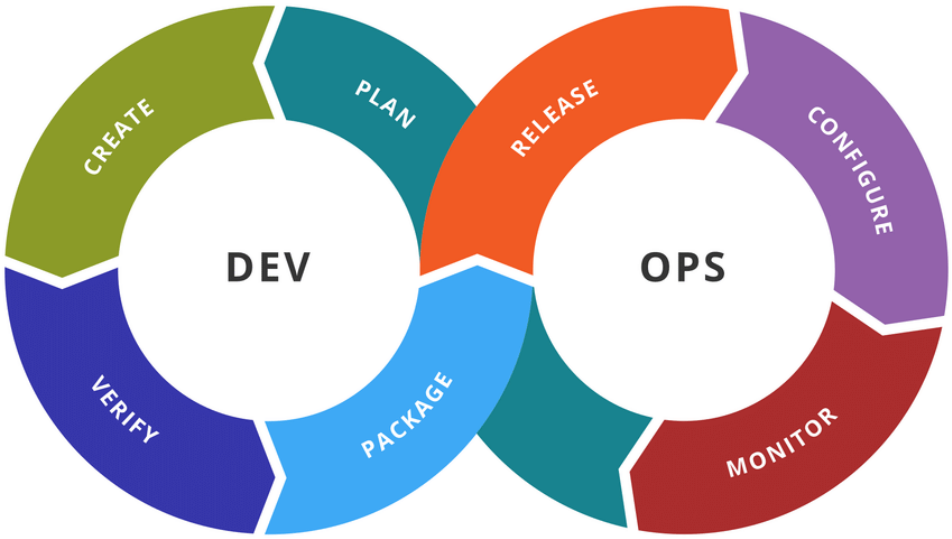

Figure 1: Stages of DevOps (Kharnagy, 2016)

- **Plan**: In the planning phase, metrics, requirements, release plans and security policies are defined.
- **Create**: Design, coding and building of the software.
- **Verify**: Execution of tests (Acceptance, Regression, Security, Performance, Configuration)
- **Package**: A release package is created, and the release gets approved through an approval process.
- **Release**: Includes all steps for the release and deployment of the application.
- **Configure**: Necessary infrastructure is configured (databases, network, and so on)
- **Monitor**: Important metrics of the application are monitored such as performance, user feedback, production metrics, etc.

Those phases as well as the ongoing feedback cycles also clearly show the agile background that DevOps has.

3 DevSecOps

DevSecOps extends DevOps with security and has the goal to develop more secure applications. A simple premise of DevSecOps is that everyone involved in the software development life cycle is also responsible for security. Security tasks are not only done manually but are also integrated into an automated DevOps workflow. This could be for example an automated security code analysis (e.g. OWASP analysis tools) as part of the build process.

The individual phases of DevOps are extended by security. In the planning phase security requirements must be identified and considered. In the Create phase security is implemented and in the Verify phase automatic security tests are carried out. Packaging and Release are extended by checks, Configuration and Monitoring will be extended to include further steps such as automatic threat detection and attack prevention.

DevSecOps was developed for several reasons: increasing IT threats, increasing importance of cybersecurity and additional external requirements such as compliance with ISO2700x, FIPS, CSA-CCM, PCI-DSS or GDPR. All these aspects must be considered during application development, which is why security is becoming an integral part of DevOps in some areas.

3.1 DevSecOps cycle

In addition to the enrichment of the DevOps phases, there is also an own DevSecOps cycle, which concentrates exclusively on security:

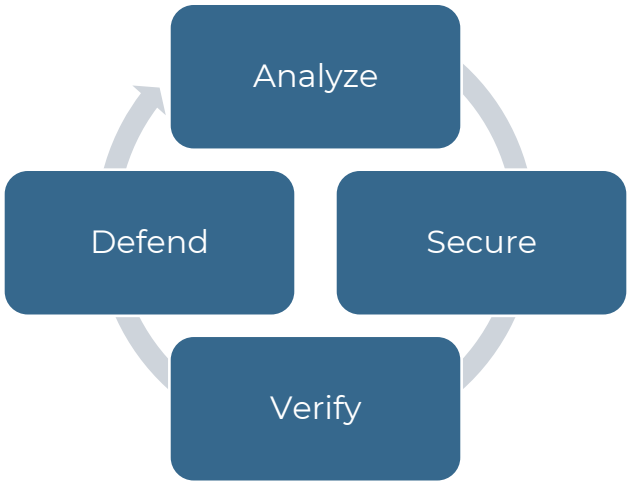

Figure 2: DevSecOps cycle

- **Analyze**: Identification of the most critical security challenges
  Selection of the most important security challenges based on architecture reviews, problems encountered, risk analysis and similar metrics
- **Secure**: Implementation of a Defense Strategy
  The Defense Strategy includes not only the implementation in the code, but also regular controlling, adjustment of support processes, execution of trainings, background checks, creation of a communication strategy, etc. The following 4 areas can be used as base to describe the defense strategy:
  - **Challenge Description**: What happened and what is the challenge about? Why is it relevant? Which systems are affected? What is/can be the impact?
  - **Defense Story**: How does defense work? How is the threat dealt with and why does this defense effectively solve the problem?
  - **Operational Guidelines**: How is the defense configured and operated? Who needs training? Who is affected by the change? What needs to be considered in the future?
  - **Security Testing Approach**: How is it verified that the solution actually works?
- **Verify**: Automation of Security Testing
  Not all security testing can be automated, but this is attempted as often as possible. This raises questions such as: What should be tested, how is it tested, use of positive/negative test, which platforms/tools to test etc.?
- **Defend**: Detect attacks and prevent exploits
  Defence through the use of various tools, such as Runtime Application Self Protection (RASP), Web Application Firewalls (WAF), Network Intrusion Detection and Prevention Systems (NIDS/NIPS), Security Information and Event Management (SIEM)

3.2 Integration of Security into DevOps - SAMM

DevSecOps can be integrated at different levels in the field of software development. The Open Web Application Security Project (**OWASP**) uses the Software Assurance Maturity Model (**SAMM**) to define four business functions to cover the core areas of software development:

- **Governance**: Governance around the processes and activities in the company related to the software development process and delivery.

- **Construction**: Development activities including processes and activities such as product/project management, requirements engineering, architecture, design and implementation.
- **Verification**: Checking the packages created in software development (code review, testing, security reviews, etc.).
- **Deployment**: Commissioning and operation

Each business function contains 3 security practices, which can be viewed independently.

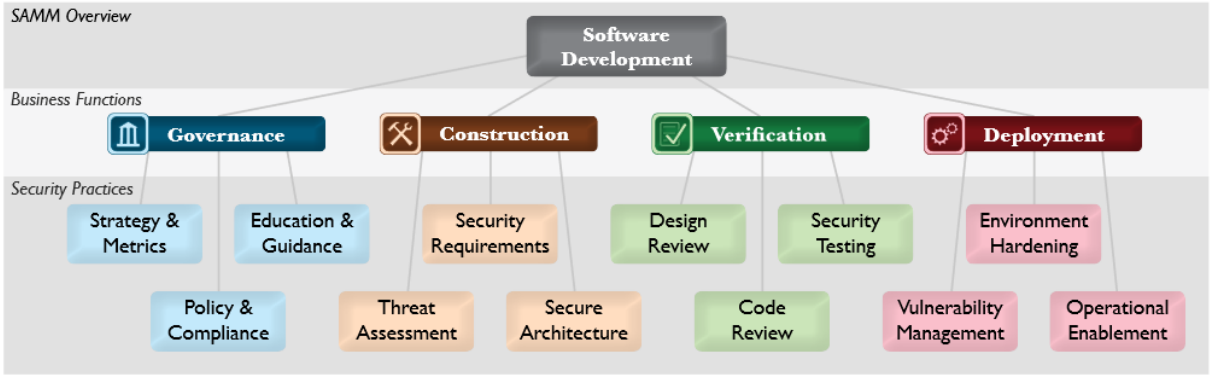

Figure 3: Software Assurance Maturity Model (OpenSAMM, 2009)

3.2.1 Governance

The organizational business function is divided into the following three security practices:

- **Strategy & Metrics**: The framework and overall strategy for the Software Security Assurance Program. Identification of security goals and possible risks as well as the definition of metrics to measure success.
- **Policy & Compliance**: Setting up security and compliance requirements. All requirements are collected - both external (regulatory requirements) and internal (internal security standard). This includes, for example, the GDPR or ISO 27001. Definition of security design best practices, identity and access management, security requirements, cryptography, key management, session management and similar.
- **Education & Guidance**: Training and support for all persons integrated in the software lifecycle. Training of persons in the areas of security and compliance, setting up knowledge bases, FAQs, etc.

For DevSecOps the following requirements may arise:

- Compliance with GDPR (how are deletion requests handled?)
- OWASP SAMM self-assessment
- Creation of Security Guidelines, IT Policies and others
- Integration of secure coding automation tools into the development process
- Threat intelligence security monitoring
- Create a secure design knowledge base for development
- Integration of security testing tools into Quality Assurance

3.2.2 Construction

Construction includes not only the software development itself, but also processes and activities before, after and during the development such as product management. Security practices in this area are:

- **Threat Assessment**: Define and prevent possible risks and dangers. Use STRIDE for the threat assessment which stands for:
  - **S**poofing: Disguising/pretending another identity, including phishing, IP spoofing or URL spoofing,
  - **T**ampering: falsification/sabotage of data e.g. from websites (exchange of parameters, manipulation of form data etc.)
  - **R**epudiation: denying of actions by assuming the identity of other users or deleting log data.
  - **I**nformation Disclosure: Unauthorized access to data, e.g. database hacks and theft of data from millions of customers or credit card data.
  - **D**enial of Service: Prevention of services, e.g. DDoS attacks.
  - **E**levation of Privilege: Unauthorized use of increased system rights

  There are other assessments like the Privacy Impact Assessment (PIA), which can be used for GDPR related topics. It helps to identify which components collect which data, how those access the data and how these are secured.
- **Security Requirements**: Security requirements depend on various factors such as business itself (e.g. banking), legal requirements (GDPR, PCI-DSS) or security compliance. Based on the requirements, the IT processes (e.g. release management) must be adapted accordingly.

  OWASP Application Security Verification Standard (ASVS) defines 3 levels of security requirements:
  - **ASVS Level 1:** minimum requirements, apply to each software *This is the compliance with basic guidelines such as OWASP Top 10.*

    - **ASVS Level 2:** Applications that contain sensitive data *Components (libraries, modules, external systems) have to be identified and documented, security libraries have to be implemented centrally, components are separated (network segmentation, firewalls, …), layer architecture is used, no critical business logic or keys on the client side, etc.*
    - **ASVS Level 3:** critical applications which contain sensitive data or need a high level of trust. This includes applications in the financial or medical sector.
      Threat model exists (STRIDE), secrets, keys, passwords are not found in source code, access control is managed centrally, master secrets use virtual or physical hardware security modules (HSM), random numbers are generated with sufficient entropy even under heavy load.
- **Secure Architecture:** Includes proactive action to design and develop secure software. This includes documentation of used frameworks, using security by design and the identification of security critical modules, services and infrastructure. Also, the identification of shared infrastructure and services within the organization and the accesses must be considered.

All the mentioned points and their results should be documented as it is crucial to consider security from the beginning and to plan the application according to those requirements. The principle Security by Design is applied in DevSecOps and must be considered in all phases, but especially in the construction phase.

3.2.3 Verification

Verification includes all activities to check the artifacts generated before the application is delivered. Automated tests, regression tests, verification of the deployment packages, documentation and all other deliverables are considered in this phase. From a security perspective, there are three areas to include:

- **Design review: I**s intended to ensure that the software is securely designed and that appropriate security guidelines are observed and checked. The earlier you recognize problems in the architecture through a review, the sooner you avoid high costs through refactoring. The following resources can be used for the review:
    - Security Compliance Checklist
    - OWASP ASVS

    - OWASP Top 10
    - CWE / SANS Top 25 Most Dangerous Software Errors
    - ATT & CK adversarial tactics, techniques and common knowledge
- **Code Review**: Review of the code considering secure coding, checking third-party components, checking the configuration, etc.
- **Security Testing**: The aim is to check whether the application fulfills the security requirements, industry standards, customer requirements and legal requirements. To do this, the following information and tools can be useful:
    - **Security Release Criteria**: Minimum requirements that must be met for the software to be released.
    - **Security Testing Plan/Cases**: Test plan for security, e.g. OWASP testing guide (https://www.owasp.org/images/1/19/OTGv4.pdf).
    - **Automation Testing Toolkits**: Integration of automatic security testing tools in the CI/CD process. This includes both code analysis tools that check for security and test tools such as vulnerability scanners (https://owasp.org/www-community/Vulnerability_Scanning_Tools), tools that are developed in the company or other existing tools – e.g. Kali Linux Testing (https://tools.kali.org/tools-listing).

  So-called release gates should also be introduced. These define various criteria's that must be met before the software is released.

It is relevant for DevSecOps to insert appropriate review processes. There are various tools for automation that perform automatic code analyzes (Static Code Review, Maintainability Index, Cyclomatic Complexity, Code Coverage, Secure Code Analysis) or perform automatic tests. The tools differ depending on the technology used for development (programming language) and the technology used for the DevOps stack (Azure DevOps, Atlassian, ...). A list of tools for different technologies is e.g. can be found at OWASP:

- .NET: https://www.owasp.org/index.php/Category:OWASP_.NET_Project
- Java: https://www.owasp.org/index.php/Category:Java
- Perl: https://www.owasp.org/index.php/Perl
- PHP: https://www.owasp.org/index.php/PHP
- Python: https://www.owasp.org/index.php/Python
- and many more

There is also a general list of tools for automated source code analysis:

- https://www.owasp.org/index.php/Source_Code_Analysis_Tools

- https://github.com/mre/awesome-static-analysis/

Language-independent tools include: SonarQube, DREK, Graudit, VisualCodeGrepper

3.2.4 Deployment

Deployment includes all steps for the actual delivery of the software. The three security practices in this area are:

- **Vulnerability Management**: Contains the definition of possible security incidents and possible vulnerabilities. NIST SP 800-61 - "*Computer Security Incident Handling Guide*" describes some processes and procedures such as Incident Handling (Preparation - Detection & Analysis - Containment Eradication & Recovers - Post-Incident Activity). In the event of a problem, there are different levels: Vulnerability received - internal/external communication - Root Cause Analysis - Mitigation - Deployment and Verification. DevSecOps is involved in all areas, especially from the root cause analysis.
- **Environment hardening**: Securing the environment through secure configurations, ongoing monitoring or thread detection systems. Checking Common Vulnerabilities and Exposures (CVEs - https://cve.mitre.org/)
- **Operational enablement**: Contains the interactions between the development team and operations team. This includes deployment, testing, running security checks or handover and secure the configuration. When a software is delivered it should at least fulfill the following points:
    - Code signing to ensure integrity and authenticity.
    - Application communication ports matrix so that as few ports as possible are opened for as few applications as possible
    - Secure application configuration

In the DevSecOps Cycle, you must ensure that the risks listed do not occur and that appropriate processes and tools are implemented, e.g. to prevent the deployment of a manipulated software package. You should also be prepared for any problems and continuously check the productive systems.

4 Integration of Security into the Development

The previously described model - OWASP SAMM - can be used to generally integrate security into DevOps. The Microsoft Security Development Lifecycle Model (**SDL**) is recommended to integrate security into the software development itself. It includes different phases of development and describes which security activities should be integrated into the individual phases.

4.1 Security Development Lifecycle (SDL)

Phases	Description, To-Dos
Training	• **Core Security Training:** Conducting training for everyone (developers, testers, IT, program managers) on the topics of secure design, development and testing, privacy
Requirements	• **Establish Security Requirements** • **Create Quality Gates/Bug Bars:** Define quality criteria, define thresholds with the aim of preventing errors during development. • **Perform Security and Privacy Risk Assessments**
Design	• **Establish Design Requirements:** Define design requirements so that time and resources are available for security (timeline, budget) and the design already contains certain requirements (e.g. cryptography). • **Perform Attack Surface Analysis/Reduction:** Analysis of possible attack scenarios and consideration of these in the design (principle of least privilege) • **Threat modeling:** Define potential threats and prepare for action
Implementation	• **Use approved tools:** Use and maintain a list of tested tools and security checks. Update tools and libraries. • **Deprecate Unsafe Functions:** Analysis of all functions and APIs and exclusion of all unsafe or potentially

	dangerous or risky interfaces or replacement with safe alternatives. • **Perform Static Analysis:** Security Code Review and other code analysis tools
Verification	• **Perform Dynamic Analysis:** Check the running software, monitor the application (memory corruption, user privilege issues and other security issues) • **Perform Fuzz Testing:** Tests with manipulated, random or other „unusual" data. • **Conduct Attack Surface Review:** Review of attack surfaces, which interfaces are there to the system, which entry gates are there etc..
Release	• **Create an Incident Response Plan:** preparation for possible problems, establishing security contacts, security plans etc. • **Conduct Final Security Review:** Final security review of threat model, tools, performance, quality guidelines etc. Potentially run penetration test again. • **Certify Release and Archive:** Certify the software before release and archive all artifacts (specifications, source code, binaries, threat models, documentation, emergency plans, licenses, third-party tools, etc.)
Response	• **Execute Incident Response Plan**: Implementation of the Incident Response Plan, which was defined in the release and execute the activities as well as ongoing monitoring.

5 Practical Example: First steps in DevSecOps

The following first quick example shows how DevSecOps could work in case of a security issue. For this example, let's assume that a simple web application was created as part of a DevOps project. After a while there is a security problem in the web application. It turns out that SQL Injection is possible and was carried out by an attacker. How can security be integrated into the DevOps process?

1. **Plan** - Implement a Defense Strategy
   The strategy in this example could be to prevent SQL injection by input validation and switching to queries with parameters. The necessary steps are divided into several backlog items so that the implementation can start.
2. **Create**: Implementation of the defined backlog items.
3. **Verify** - automation of security testing
   To prevent such problems in the future, security testing will be integrated into the verification process. For this purpose, a corresponding tool is installed in the development environment of the developer. Furthermore, this tool is integrated into the CI/CD process. As soon as the tool detects a security issue during the build, the build fails and deployment is prevented.
4. **Release**: Release of the new software
5. **Monitor** - monitoring of the productive system
   As soon as the new web application is online, it is continuously checked whether there are new security problems and whether SQL injection is used for other purposes. Furthermore, the monitoring can be expanded to automatically generate a warning when using SQL Injection.
6. **Analyze / Plan** - identify the next steps
   As usual in an iterative environment, the application is tried to be further improved. Problems are solved one after the other. SQL injection has been fixed. The next step could be the extension of the security testing tools to prevent cross-site scripting.

6 Implementation plan: Integrate Security into DevOps

The following describes how DevSecOps could be implemented in an existing company. It is assumed that DevOps has already been implemented and is now being expanded to include security. The following steps are basically generic, but in some cases, it is assumed that the company develops software with .net Core and Microsoft Azure. The steps and resources listed may therefore be adapted for other technologies.

Since DevSecOps is strongly based on agile approaches, the following plan consists of sprints. The number of sprints and the length could highly vary based on the capacity, the number of identified issues and the maturity of the system and the team. In Sprint 1, an OWASP Top10 check is already being carried out. If an extraordinarily large number of errors are found here, this by sure affects the next sprints.
The sprints are designed in such a way that the first ones are more precisely defined and the latter less precisely, since the tasks and requirements for the next sprints arise in the individual sprints. The specified sprints and their tasks serve as a guide for the implementation of DevSecOps in an existing company.

6.1 Sprint 1 – Requirements & first steps

Goal: At the end of the sprint, there should be an awareness of security and its importance. Everyone should feel obliged to security and have gained basic knowledge in this area. Likewise, all security and compliance requirements must be defined at the end of the sprint.

Tasks	Responsible
DevSecOps detail planning Plan the sprints, define base requirements and extend this plan if needed. This is a permanent task in an agile environment, so don't plan too much.	CISO (Chief Information Security Officer)
Collect Security & Compliance requirements Capture all requirements in coordination with the CEO, CIO, compliance and the legal department.	CISO

Prepare/execute Secure Coding Trainings for DevOps Run Secure Coding Trainings with partners/training providers.	CISO, DevOps
Setup Security Knowledge Base Setup a KB for the Software Development and IT which must be maintained and extended regularly.	IT
Hang up .Secure Coding Sheet so that it is clearly visible e.g. https://www.owasp.org/index.php/.NET_Security_Cheat_Sheet	DevOps
Find and define the first security tools for the DevOps process Selection based on: https://github.com/mre/awesome-static-analysis/ or https://www.owasp.org/index.php/Category:OWASP_.NET_Project Potential Tools: • OWASP AntiSamy .net • Security Code Scan for .net (https://security-code-scan.github.io/) • SonarLint: https://www.sonarlint.org/visualstudio/	IT-Architect
Carry out an OWASP Top 10 Review Review the applications to prevent common OWASP Top 10 issues	IT-Architect

6.2 Sprint 2 – First security improvements, architectural review

Goal: At the end of the second sprint, the first identified security issues should be remedied and the first tools are integrated into the CI process. The architecture is reviewed and documented so that all components of the system are known. The result of the review also serves as input for the next sprints.

Tasks	Responsible
Integrate first Security Tools into the DevOps process Integrate the tools identified in Sprint 2 into the DevOps process. This could be that Static Code Analyzers are integrated into the IDE and the build process.	IT

Fix occurring warnings The newly integrated tools will show a lot of warnings which should be fixed. These fixing will start in the second sprint and could take multiple sprints, depending on the number of warnings.	Development
Resolve OWASP Top10 issues The issues identified in Sprint 1 should be fixed. If there are too many issues, then those should be prioritized and spread across the next sprints.	Development
Add fixed issues to automated Test Extend the automated tests so that the identified security issues are covered by the test cases. If this is not (yet) possible, document those cases in the test management so that they are at least covered and documented by manual tests.	Tester, Development
Security Review of the architecture Review the architecture based on the OWASP ASVS. To do that, define the level which you should already know from Sprint 1. After that, go through the checklist and create security issues (https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf).	IT-Architect
Documentation of the architecture Document the architecture including all components, connections, interfaces etc. This should be a result of the security review of the architecture.	IT-Architect
Fill Security Knowledge Base, define Guidelines Extend the Security KB, Definition of Guidelines – e.g. Cryptography (Hashes, Algorithms, ...), Session Management, Identity Management, etc.	DevSecOps

6.3 Sprint 3 – Improve Security, Configuration & Security review

Goal: At the end of the third sprint, the security level of the applications should be further improved. The configuration is reviewed from security perspective and changed accordingly. External tools (3rd party libraries and co) are documented and checked.

Tasks	Responsible
CWE/SANS Top25 Test run After reviewing and fixing OWASP Top10 issues, it's time to further check the applications. This can be done by using the „25 Most Dangerous Software Errors" (CWE/SANS Top25)	IT-Architect
3rd party tools review & documentation Review and documentation of all 3rd party libraries and tools. Those must be known and checked for security issues.	IT-Architect, DevSecOps
Define Metrics for Security A good starting point are the Common Criteria for Information Technology Security Evaluation with its Evaluation Assurance Levels (EAL) 1 – 7. Others are the Orange Book (TCSEC), SSE-CMM, NIST FIPS-140 series or NIST SP 800-55. (see 7 - Further information & Tools)	CISO
Check Configuration for Security Check, where and how Keys, Passwords, Connection String and other confidential information is stored. Are they in the source code? Do you really need to store them? What about production environment configuration? Document the issues and add them as backlog items.	DevSecOps
Release Management Security Review Check the current release management process for any issues, missing verifications and others and document it.	IT
Define Security Testing Plan Develop a plan about how you are going to test the security of the system (roadmap, deliverables, activities, timeline,	IT-Security

resources ...). This plan is the base for security tests executed later.	

6.4 Sprint 4 – Next level Security, remediate risks by 3rd party tools/libraries

Goal: At the end of the sprint, the overall security should be greatly improved by fixing the identified CWE/SANS Top25 issues, as well as problems in the configuration. The definition of quality gates creates the basis for sustainably improving the release quality in the next sprint.

Tasks	Responsible
Fix identified CWE/SANS Top25 issues and extend the Verification process Resolving the problems encountered and integrating them into the CI process, if possible. If this is not possible – add verification tests in the test management.	DevSecOps
Secure configurations Securing Connection Strings, Keys and others. Better is to completely remove them by using e.g. managed identities.	DevSecOps
Define Release Gates/Quality Gates Definition of metrics and quality guidelines that must be fulfilled before the software is released.	CISO
Define Incident Handling & Incident Response Plan Define and document how security incidents are handled and how to act in case of an incident. Define emergency contacts, which must be involved. Go through the NIST SP 800-61 - "Computer Security Incident Handling Guide".	CISO
Replace 3rd party Tools If there were any issues with 3rd party tools or libraries discovered in the last sprint, they may have to be replaced.	Developers

This could take longer than just one sprint, so plan it for the next sprints if needed.	
Research & Select Monitoring Tools Researching tools that can be used to monitor security. Drawing up a list and select the first tools.	IT-Architect/IT-Security

6.5 Sprint 5 – Monitoring, Threat Assessment & internal Training

Goal: After this sprint, the security of the applications are tested, and identified issues are documented. The applications are now monitored so that attacks can be detected. All members of the organization are familiar with security and have finalized security awareness trainings

Tasks	Responsible
Setup/integrate Monitoring Tools Setup or integrate the monitoring tools identified in the last sprint.	IT
Threat assessment according to STRIDE Do Threat Assessments according to STRIDE	CISO
Do internal Security Test Do an internal security test and document all identified issues.	IT-Security
Conduct IT-Security Awareness Training The whole organization must be trained and get knowledge about security. This should also prevent one of the main threats which is “Social Hacking”.	CISO
Implement Release/Quality Gates Implementation of the quality goals that were defined in the last Sprint. These quality gates should be automatically checked. If that’s not possible, define some manual checks that must be done for each major and minor release.	DevSecOps

6.6 Sprint 6 – Preparation of the final Security/Penetration Test

Goal: As many security problems as possible should be fixed in this sprint. This applies in particular to those who appeared in the internal security test. The penetration test should be prepared.

Tasks	Responsible
Resolve all identified issues Elimination according to priorities, preparation of the application for the external security test. Extensions to the automatic verifications to automatically check all problems that occur. Consider the results of the threat assessment.	DevSecOps

6.7 Sprint 7 – Security & Penetration Test

Goal: At the end of the sprint, all weaknesses in the IT systems and in the organization should be known so that they can be remedied in the next sprint. This test must be carried out by an external partner and should, if necessary, be in preparation for a possible certification. The results are therefore not only the weak points, but also necessary adjustments for the acquisition of a certification.

Tasks	Responsible
Perform external Security & PenTest Run a penetration test which checks the technical systems. Do security audits, phishing campaigns and crisis scenarios to also check organizational measures and the human factor (employees).	External Security Partner
Extend the IT-Security Policy Extend the security policy by review processes etc.	DevSecOps

6.8 Regular Tasks

During the sprints there are certain tasks that should be carried out continuously. In general, everyone in the team is responsible for security and the following list does not include everything by far but should serve as a rough checklist so that certain points are not neglected.

Tasks	Responsible
Fix Code Analyzer (Security) Warnings Warnings that occur must be fixed regularly as part of daily job.	DevSecOps
Increase Code Coverage Increase unit-test code coverage as well as security tests to improve the quality.	DevSecOps
Go through the DevSecOps Cycle Analyze – Secure – Verify – Defend: These steps are repeated within the sprints whenever security problems are identified. Issues are documented, added to the backlog and prioritized.	DevSecOps
Update the Security Knowledge Base The security knowledge base is a living system and is constantly extended. Share information about security within the company.	DevSecOps

6.9 Outlook

Security is basically never done and can always be improved. Some ideas are:

- Run Penetration Tests regularly
- Run Crisis Simulations
- Review policies and other guidelines
- Do Security Awareness Trainings (consider online and offline trainings)
- Run simulated attacks (phishing attack, DDoS, …)
- Hack your own system
- Start a bug bounty program
- …

7 Further information & Tools

ATT&CK – Adversarial Tactics, Techniques and Common Knowledge: https://attack.mitre.org/

Awesome AppSec: https://github.com/paragonie/awesome-appsec

CAPEC – Common Attack Pattern Enumeration and Classification: http://capec.mitre.org/data/definitions/1000.html

Cloud Security Alliance - Consensus Assessments: https://cloudsecurityalliance.org/working-groups/consensus-assessments/

Common vulnerabilities and exposures (CVEs): https://cve.mitre.org/

CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/

CVSS – Common Vulnerability Scoring System: https://www.first.org/cvss/

GDPR Privacy Impact Assessment: https://gdpr-info.eu/issues/privacy-impact-assessment/

Microsoft Elevation of Privilege Card Game: https://www.microsoft.com/en-us/SDL/adopt/eop.aspx

Microsoft Threat Modeling Tool: https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool

NIST Cyber Security Framework: https://www.nist.gov/cyberframework

NIST Information Security Handbook – A guide for Managers: https://csrc.nist.gov/publications/detail/sp/800-100/final

NIST SP 800-50: Building an Information Technology Security Awareness and Training Program: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-50.pdf

NIST SP 800-61 – „Computer Security Incident Handling Guide“: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf

Open Reference Architecture for Security and Privacy: https://security-and-privacy-reference-architecture.readthedocs.io/en/latest/

Open Security Architecture Patterns: http://www.opensecurityarchitecture.org/cms/

OpenSAMM Self-Assessment Worksheet: http://www.opensamm.org/downloads/resources/20090925-SAMM-Assessment-v0.4.xls

OWASP ASVS checklist for audits: https://github.com/shenril/owasp-asvs-checklist

OWASP Code Review Project: https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

OWASP Cornucopia: https://www.owasp.org/index.php/OWASP_Cornucopia

OWASP Security Knowledge Framework: https://www.owasp.org/index.php/OWASP_Security_Knowledge_Framework

OWASP threat modeling cheat sheet: https://www.owasp.org/index.php/Threat_Modeling_Cheat_Sheet

PCI-DSS – Payment Card Industry Data Security Standard: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf

SAFECode Security White Papers: https://safecode.org/publications/

SEI CERT Coding Standards: https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards

Static analysis tools for all programming languages: https://github.com/mre/awesome-static-analysis/

8 References

Drinkwater, Doug (2018): What is DevSecOps? Developing more secure applications, in: https://www.csoonline.com/article/3245748/devops/what-is-devsecops-developing-more-secure-applications.html, last checked: 10.11.2018

Gualtieri, Mike (2011): I Don't Want DevOps. I Want NoOps., in: https://go.forrester.com/blogs/11.02-07-i_dont_want_devops_i_want_noops/, last checked: 10.11.2018

Hüttermann, Michael (2012): DevOps for Developers, Berkeley: Apress

Hsu, Tony (2018): Hands-On Security in DevOps, Birmingham: Packt Publishing

Jendrian, Kai (2012): Sicherheit als Qualitätsmerkmal mit OpenSAMM, DuD – Datenschutz und Datensicherheit, 4/2012, online: https://www.secorvo.de/publikationen/opensamm-jendrian-2012.pdf

Kharnagy (2016): A visual representation of the DevOps workflow, in: https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg

Microsoft (2018): Security Development Lifecycle, in: https://www.microsoft.com/en-us/sdl, last checked: 10.11.2018

OpenSAMM (2009): Software Assurance Maturity Model, in: http://www.opensamm.org/downloads/SAMM-1.0-en_US.pdf

OWASP (2018): OWASP Application Security Verification Standard 3.0.1, in: https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf, last checked: 10.11.2018

Wikipedia (2018): DevOps Toolchain, in: https://en.wikipedia.org/wiki/DevOps_toolchain, last checked: 10.11.2018

Williams, Jeff (2018): Introduction to DevSecOps, in: https://dzone.com/refcardz/introduction-to-devsecops, last checked: 11.11.2018

Willis, John (2010): What Devops Means To Me, in: https://blog.chef.io/2010/07/16/what-devops-means-to-me/, last checked: 11.11.2018