Tuomas Tiensuu

DevSecOps adoption
Improving visibility in application security

Master's thesis

Master's Degree Programme in Cybersecurity

2022

XAMK
South-Eastern Finland
University of Applied Sciences

XAMK
South-Eastern Finland
University of Applied Sciences

Author (authors) Tuomas Tiensuu	**Degree title** Master's Degree Programme in Cybersecurity	**Time** 2022
Thesis title DevSecOps adoption: Improving visibility in application security		63 pages
Commissioned by Vilma Blomberg		
Supervisor Kimmo Kääriäinen		

Abstract

For organizations to be able to build digital products that are as secure as possible for their customers, security must be implemented in every phase of the software development life cycle. Good application security management and security improvements require good visibility of security activities in the SDLC. This research studied visibility in application security, and what factors are important to consider when aiming to improve visibility.

The action research method was used in this study. The theoretical part consists of an introduction to modern software development, DevOps practices and security automation, where visibility is needed. The section also demonstrates the standards and certifications widely used in the field, as well as various activities during the secure software development lifecycle.

The primary goal of this study was to amplify the most important issues that should be considered when developing application security visibility. The secondary goal was to define the key roles in the organization that need visibility, so that software development could be performed securely and following best practices.

The research showed that when improving application security visibility, it is necessary to pay attention to the impact of the security findings provided by the visibility, and how the situation can be enhanced during the entire software development life cycle. It is very important to provide visibility to the various stakeholders in the organization, so that any actions can be taken to improve application security. However, the focus should be on the business impact, the most accurate situational awareness, and clear guidelines, that can be used to improve application security.

Keywords

devsecops, application security, visibility, security tooling, security automation, paved road

Contents

LIST OF ABBREVIATIONS

API	Application Programming Interface
CD	Continuous Deployment or Continuous Delivery
CI	Continuous Integration
CSRC	Computer Security Resource Center
CVMS	Centralized Vulnerability Management System
CVSS	Common Vulnerability Scoring System
DAST	Dynamic Application Security Testing
DevOps	Development and Operations
DevSecOps	Development, Security, and Operations
EASM	External Attack Surface Management
IAST	Interactive Application Security Testing
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
NIST	National Institute of Standards and Technology
PO	Product Owner

QA	Quality Assurance
SAST	Static Application Security Testing
SCA	Software Composition Analysis
SDL	Software Development Lifecycle
SSDL	Secure Software Development Lifecycle

1 INTRODUCTION

In the modern software development world security should be in products DNA, and it should not be an afterthought. That means we think of and try to solve security issues early in software development lifecycle (SDL) and try to implement different security practices and activities in each phase of the SDL. These activities may include factors such as threat modelling, risk analysis, static and dynamic application security testing, software composition analysis and infrastructure security checks, to name only a few. (Zeeshan 2020.)

However, as especially bigger companies have usually many different software engineering teams and these teams get input and data from many different tools – security and other development and operations wise critical tools – there is often a problem with the amount of information development teams and security teams get from different sources. This makes security very hard in the modern application engineering world where DevOps methodologies are used, as without proper automation and workflows in place there are simply too much noise, various alerts and findings from security and other sources from the toolchain. Developers should not have to go through multiple external security scanners and read vulnerability reports from penetration tests, such as SAST, DAST, or SCA tools separately. Also, security engineers should not need to go through every single security tool report with each product team one by one since this is not a scalable way of working in the long run. In fact, it is almost impossible due to a lack of application security resources compared to product development resources (Sonatype 2020).

It is also unfair to expect that the members of the development or operation teams understand security issues at the same level as the information security professionals. Security issues are not included in a job description of software developers and often they are not trained for it, either. Instead, the current trend and consensus has turned to organizations using application security teams to produce secure frameworks, libraries, and secure defaults for developers to use. The lack of security visibility and insufficient security checks in the developer workflow are issues that organizations have struggled with for a long time, and as one of the ways to solve the issue, security teams have started adoption of so-called security guardrails. This adoption tries to eliminate the friction between development teams and security teams ensuring development teams can produce software quickly and also securely. With the help of these automated guardrails, development teams can achieve the greatest possible autonomy while the most important security controls come into use with minimal bottlenecks from the security team and the security team does not act as a gatekeeper while new functionalities are released into production (DZone 2022).

Several different parties benefit from good visibility to application security. Lack of visibility makes it difficult for product owners to make decisions in the software development lifecycle because there is not always a proper understanding of security risks. For both application security and incident response team, the lack of visibility prevents from getting an overview on potential impacts and where to look for malicious actors. In large organizations there is need for good compliancy and risk management. Lack of visibility makes it hard to provide, e.g., auditors evidence on how controls are being implemented. In the target organization that is a particularly important factor because one of the goals in improving visibility is to get an ISO 27001 certificate for the company's digital products. Without aggregated views on the security risks of the whole environment, organizations cannot make a proper risk analysis and decisions to mitigate or accept the risks.

In December 2021 a severe vulnerability was found in Java library called Log4j. This started a chain of large investigations in most companies. One thing that internal security people were struggling with was that in many cases they did not know how many systems were impacted and what applications were using Java and this particular library. As DevOps and Agile development practices emphasizes a working software over exhausting documentation, the documentation created is often made only for team itself to use and supporting functions in the company have to find other ways to collect important and up-to-date information about the systems. Security tools in the target organization were also not very functional for this purpose and did not offer proper visibility to solve the problem very effectively.

The goal of this study is to help product development teams, application security team, and other stakeholders by providing visibility to security status of the different products and applications and make application security practices a little bit easier by bundling a security toolchain of many different testing tools and how these capabilities could be used to improve visibility in a meaningful way. There is also a greater need for security visibility towards different stakeholders since security issues can have a detrimental effect on the business and especially if the company is not aware of the security risks that can have a major impact. This study will also focus on how to enhance the visibility for risk management and compliancy purposes. (DevSecOps Community Survey 2020.)

1.1 Thesis topic

The objective of this thesis is to study security visibility in the modern software development lifecycle where development, security and operations try to work together seamlessly to achieve a common goal – usually referenced as DevSecOps - as well as what is the visibility status currently in commissioner's application security domain and how it could be improved. This thesis focuses on the visibility of security in large corporations that have shifted mainly to DevOps tools and practices in their engineering and development work. In this context, the main focus will be on the R&D functions of the target organization, although the research can also be continued to cover other functions, such as Enterprise IT. The reason why this topic is worth researching is because modern software engineering has taken significant leaps forward past years and at the same time as the complexity has increased, so has the need to understand the technologies and the security risks associated with them. One of the most challenging factors from security perspective is variety and number of programming languages, frameworks, tools, and platforms that different product development teams use. Even though DevOps principles strive for seamless collaboration between development and operations teams, and even unite those two, it is often the case in software engineering that teams do a lot of work in silos. This is a reason why security teams find it difficult to actually improve the level of security in software development lifecycle because they do not always know what and how things are done in the specific team. (Ribeiro 2022.)

1.2 Background of the commissioner

The commissioner of this work is a Finnish engineering company in the machinery industry. As many other engineering and traditional industry companies, the commissioner has also started digitalization journey a few years ago meaning that new technology has been taken in the use to support its new products and services. The commissioner has been developing new digital services to bring new added value to its customers and their end users. In an increasingly digitalized world, we see increasingly traditional equipment connected – the Internet of Things – and securing these new digital services is one of the top priorities. With the traditional equipment being connected with cloud technologies and applications, good cybersecurity practices are coming a necessity.

In the commissioner's business, customers often have strict requirements for cybersecurity. Software development must be secured throughout its life cycle, from the design and planning phase to maintenance. Visibility to security practices helps to show customers that security is taken into account at every stage of development.

2 RESEARCH PROBLEM AND RESEARCH QUESTIONS

Often, large organizations have need for a high-quality and clear application security guardrails, policies, and guidelines. Naturally, having a good application security program can protect an organization's business assets and property, but there may also be external requirements for companies to adhere to strict standards and have security policies followed. However, from the application security team's perspective, it can often be the problem that it is unknown how different teams do their day-to-day work and how security guidelines are followed in different software development teams. It is not necessarily only a matter of controlling the working methods and the technologies used, in order to operate with them as securely as possible, but rather to gain an understanding of how the security of development work could be supported the best way, and what are the factors in visibility that need to be improved so that development teams can take security into account, while in parallel, building functional solutions to customers to achieve important business goals. It is not uncommon for large corporations to have twenty different product and software development teams, hundreds of applications and only one application security team trying to take care of security practices. To track what is the security status in each team, product and application is a major challenge. (Ribeiro 2022.)

This problem can be divided into the following research questions

RQ1: *Which factors should be considered effective to improve application security visibility in a large organization?*

The first goal can be set as trying to find important factors that affect visibility in application security and how things could be improved. This also requires an investigation of the current state of the organization to gain a generic understanding of how application security practices are implemented with different teams. From this, the research can be continued with the following research questions:

RQ2: Who are the stakeholders in the large organizations that need visibility to application security and how the organization benefits from it?

The purpose is to find out which stakeholders are the ones who benefit from increasing the visibility of application security, and who can effectively affect prioritization if the visibility is improved.

RQ3: Which application security metrics must be visible to different stakeholders?

Which metrics are important to bring to the attention of different stakeholders so that development can be expected to take place? Which metrics are essential in order to give different stakeholders visibility into the current state of the organization's application security so that risks can be reduced, and security be improved?

2.1 Research method

The research was studied by using the action research approach. The action research is usually carried out as a participatory and cooperative study. All community members act as active participants and influencers in change and research processes (Lapan 2011). Information and data related to the study is gathered and analyzed using observation, surveys and reports from security, orchestration, and reporting systems. One reason behind choosing an action research method is Jean McNiff’s idea that if we are not happy with current practices, we should strive to influence and push strongly towards change, and always to challenge the current understanding of the situation (McNiff, 2002). In the longer term, the research's main idea is to influence things that are not at the level they should be.

The different phases of action research provided by McNiff (2002) are listed below:

- Surveying the situation and finding out what are the starting points
- Ideation and conceptualization of the activities
- Initiating and implementing the activity
- Monitoring the effects and making observations
- Evaluation
- Possible implementation or correction of a new form of activity

Research methods may change along with the study if new and better ways are found to be useful. The first half and theoretical part of the study includes explanation of the terminology and an introduction to basic concepts of

DevSecOps and modern application security engineering. At this stage, a methodology assessment for application development teams was also carried out. This assessment included 25 different questions that were divided into 3 categories: culture and collaboration, velocity and process efficiency and tools and automation. Third party assessment tool GitLab’s DevSecOps Methodology Assessment was used to help gather questions and categorize them accordingly. This part of the thesis is going to try to cover as recent literature of the topics as possible. During this phase problems are identified, and data is collected using quantitative research methods.

Information for this research phase was gathered from books in the information technology field, electronic publications, Internet articles and studies and surveys from global IT and security companies. The majority of the books for this study were acquired from O’Reilly online book service as they offer up-to-date material and most of their writers are well-known pioneers in their field. Modern application security engineering is moved on at such a pace that most books released a few years ago are clearly and irretrievably outdated. (Farley 2021.) The second part of the work includes empirical research using empirical evidence to find answers to research questions. This stage involves analysis and process development itself, including a survey that was sent to people in different positions in the IT and software development field:

- Software Developers / Software Engineers
- Operations Leadership
- Technology Executives
- Devops Leadership
- DevOps Engineers
- Software Architects
- Cloud Architects
- Security Leadership
- Product Owners

The study is limited to large businesses with several different software engineering teams developing new solutions. This work also focuses only on companies whose software development is meant to be done using modern DevOps methods and Agile principles.

In recent years, several studies have been completed related to DevSecOps adoption and security challenges as Agile and Lean methods evolve. Most studies' results were analyzed by using qualitative methods. (DevSecOps: A Multivocal Literature Review, 2017.)

2.2 Previous research and literature review

Visibility is an important term in this study and will be closely examined during this study. Visibility is often defined as "the state of being able to see or be seen", and in this research context it can be defined as the ability of a process and system to produce high quality information for the needs of different stakeholders and is always available, regardless of time and place. The goal of good visibility is to have as complete picture of security status as possible. The probabilities of security success are reduced when there is no precise visibility into security activities and therefore security issues cannot be effectively addressed or developed. It is impossible to control or protect devices, applications, data, or processes related to these if visibility is not enforced. Thomas and Tabassum concluded that security training for software developers helped create visibility into an organization's security issues, although the training was not otherwise considered relevant (Thomas & Tabassum 2018). Studies have shown that maturity and methodology assessments have improved security visibility in various organizations and thus positively influenced change and risk management. (Mohammed 2015.)

The search string "application security visibility" or "security visibility" did not directly give that many results. In several books dealing with application security or cloud security though, there were chapters that discussed how visibility can be improved. However, these were mostly related either to the visibility offered by the tools, which is a very relevant topic, or to security operations, which this work does not really focus on. It seems that application security from the visibility point of view has not yet been studied at least very widely, so this study feels very timely.

2.3 Visibility Trends in application security

There are several concerns related to visibility in application security context. The adoption of API centric software development weakens the visibility in traditional sense compared to the previous development model, when you could still test the functionalities of the application in practice by going to check the application running in production and testing it dynamically. With an API centric approach or single page architecture, gaining appropriate visibility to running applications in production environments becomes more complex. However, good management of APIs can improve the visibility and comprehensibility of systems. If the essential assets are known and the architecture is clear, a well-planned and documented, visibility into the functionalities can even improve and thus the needs and possible gaps of security are easier to understand. Today, API-based integrations are de facto between different IT applications, be it a client-server relationship or process-to-process communications. Modern companies rely heavily on API’s and microservices not only to build but also to connect applications and data flow. (Chatterjee 2021.)

The structure of architecture always depends on the organization and the individual solution but grouping and sharing APIs between different domains can be divided into layers as shown in Figure 1.

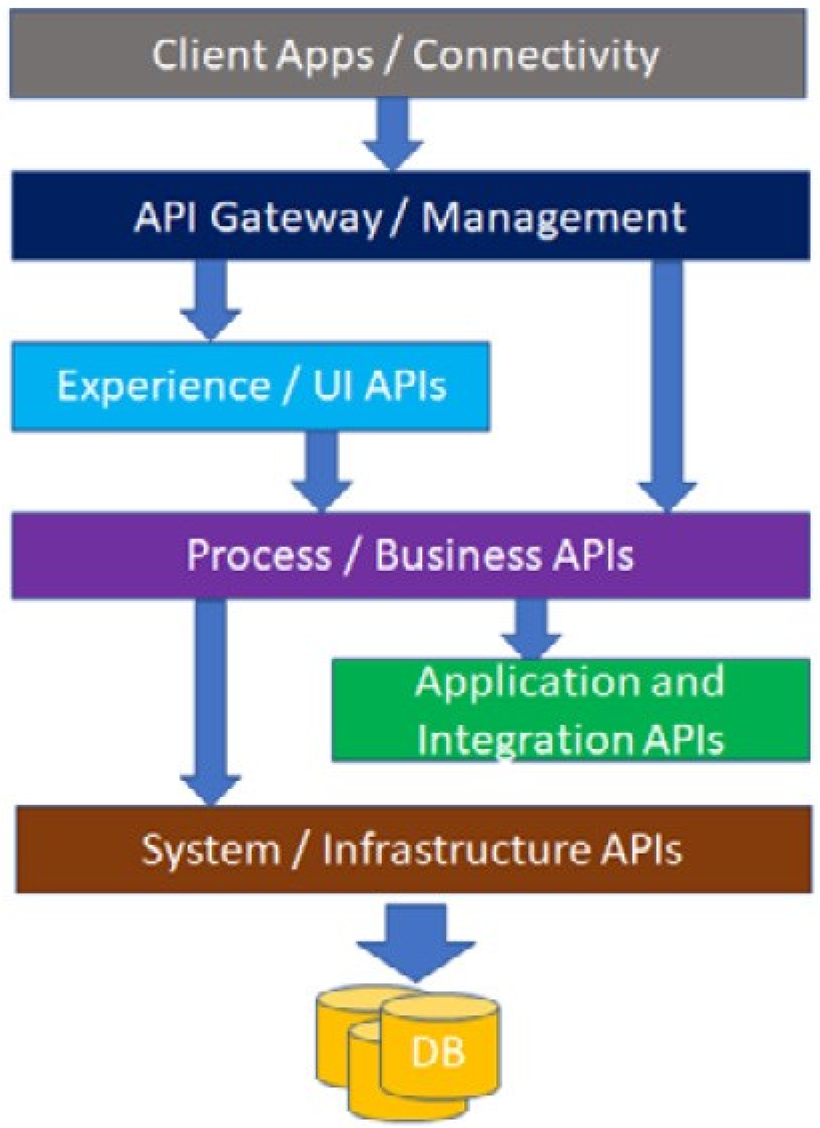

Client Apps / Connectivity
API Gateway / Management
Experience / UI APIs
Process / Business APIs
Application and Integration APIs
System / Infrastructure APIs
DB

Figure 1. Designing API-first Enterprise Architecture (Chatterjee 2021)

Another major trend in software development and the cloud native approach is Infrastructure-As-Code (IaC). In the IaC methodology, operations workload is largely automated, and the configuration of the environments is handled in roughly the same way as the application code. IaC configurations are usually defined according to template file and with the help of these files, the information about environment variables can often be seamlessly transferred to other systems by using various integration methods. This improves security visibility because security issues related to misconfigurations can be easily detected through these files, and automatic testing targeting these files is therefore a relatively reliable way to make sure that the environments are sufficiently secure.

According to Podjarny (2021), Infrastructure as Code came roughly in two different phases. At first, the transition was led by tools that enabled commanding several different servers and other systems, usually virtual machines, at the same time and set different types of information security settings and controls easily for many computers at once. Examples of these tools are e.g., Puppet, Chef and Ansible. The second wave was led by the "cloud native" transition and the configuration of cloud services and the setting up of cloud infrastructure. This wave was led by Terraform, which made it possible to tune the infrastructure to match the applications used in it. The tools of the first wave later developed to meet the needs of the second wave, and today popular and widely used solutions are tools such as Kubernetes Helm charts, AWS CloudFormation and Azure ARM. Infrastructure as Code brings new challenges to security professionals because new tools and existing IaC syntaxes must be learned. However, it also enables security automation and potentially improves visibility into the security of the infrastructure because it is possible to observe and scan the configurations directly from the template files that define the infrastructure settings.

Along with infrastructure-as-code, other as-code methods have started to be widely used. Methods such as policy-as-code and everything-as-code are getting more popularity, which in its brevity means more automation.

Security teams are trying to solve problems regarding lack of asset management and securing services what their organization is exposing while not making too much noise and maintaining high accuracy testing on their digital assets. Organizations need to concentrate more on the complete attack surface management, discovering and assessing their Internet-facing assets and scanning for known vulnerabilities or anomalies. In particular, by combining the DAST scan during software development and attack surface management, the overall visibility can be significantly improved, and an understanding of the correct attack surface can be gained, which is not always seen from within the organization, but actually from outside it. Attack surface management is considered as top challenge in the application security field in 2022 and is rising to be even more relevant as companies shift to greater use of public cloud. Security teams must continuously analyze outputs, define severities for security defects and prioritize and govern remediation efforts while all these vulnerabilities and defects are discovered from various sources. (Detectify 2022.)

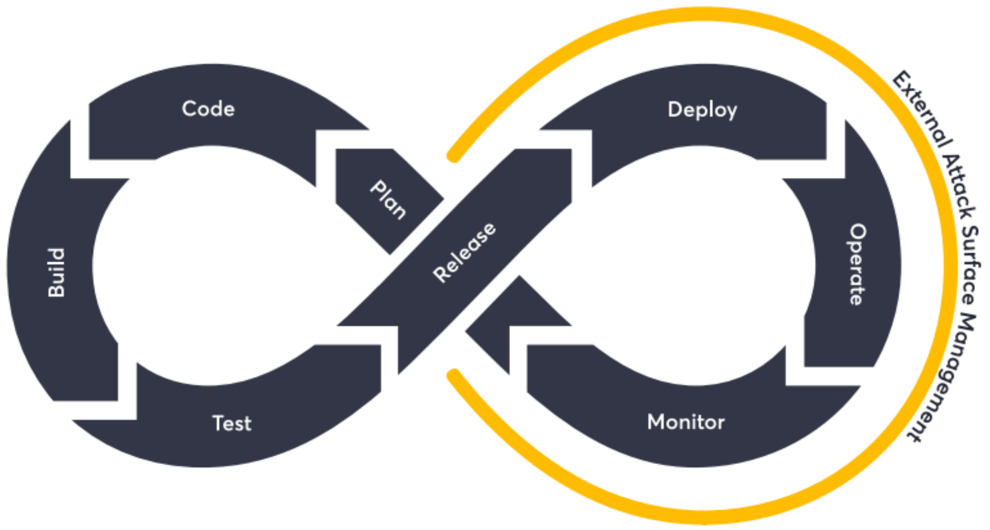


Figure 2. EASM in existing workflows (Detectify 2022)

3 APPLICATION SECURITY PIPELINE FOR CONTINUOUS VISIBILITY

It should be considered that it is possible to improve security visibility at all stages in the secure software development pipeline. Understanding the true security posture of applications requires that we be aware of the need for various security controls in the plan, code, build, test, release, deploy, operations and monitor phases in the lifecycle. To achieve this, we need to create a shared security responsibility culture among several different stakeholders, by providing continuous and automated visibility. (Chargebee 2022.)

3.1 Security visibility

With security visibility the goal is to obtain a complete picture of a company’s security activities and the ability to execute them. In general terms, security visibility provides a holistic picture of an organization’s security status and ability to respond to risks and threats. Improvement of security visibility can be achieved by cultural movement but also by technological investment. A good threat management platform is able to present a versatile view of the attack surface, showing all the organization's exposed assets and the risks and vulnerabilities that apply to them. The most important aspects regarding security visibility are visibility into the security posture, how the organization's assets compare to benchmarks, and automatic workflow when a security event occurs. (Cooper 2020.)

In this thesis, visibility mostly means visibility in the context of application security and not in the broader context of cybersecurity, although the same principles and techniques presented in the work can also help in achieving wider visibility in the organization.

3.2 DevSecOps

DevSecOps is a cultural and technological movement that aims to merge development, security, and operations in software development lifecycle. It adds security in every activity in software development process while maintaining and respecting DevOps principles and its manifesto. These activities can be divided into four different phases: requirements, design/development, testing and deployment. As DevOps aims to improve speed of development and deployment of an application, DevSecOps ends up helping overall security of the application since security testing will be partly automated, security bugs can be fixed, and updated software deployed much faster. More than anything else, DevSecOps aims to remove the human where manual intervention is not needed. As soon as you get the person out of the way, human errors such as typos, mistakes, jumping ahead of instructions etc., disappear from the processes. Of course, not all mistakes are made by humans, but by switching to DevSecOps, resources can be shifted to where they are most effective. (Dang 2021.)

Although DevSecOps aims to combine people, processes, and tools into one well-functioning entity, the term often refers to only one domain of this principle, e.g., testing tools and how the integration of different application security tools and other DevOps tools is done smoothly and without disrupting the developer’s workflow.

3.3 DevOps

DevOps refers to a methodology and culture that aims to integrate development and operations to improve the overall level of quality in software development lifecycle. DevOps focuses largely on automation and testing, and one key principle is to develop processes by automating as many repetitive tasks as possible. The main idea of DevOps is often that one team is responsible for the software throughout its entire life cycle, from the design phase all the way to production and maintenance. In the traditional model, development and production have their own individual goals. The DevOps model is closely linked to agile software development. DevOps is a cultural and technical way of working that enables the ideal of agile development: the rapid development and release of new software versions, reacting to possible changes in the operating environment. As much as anything else, the move to DevOps or DevSecOps is about getting humans out of the middle of processes that are simply better handled by software. (Dang 2021.)

As Podjarny (2021) states, DevOps has changed working methods and technologies during the software development lifecycle. It is predicated on independent and autonomous development teams, who are responsible to own the application end to end, meaning in the end of the day, they are also the ones who are accountable for security for their own products. Today, in many contexts, DevOps also is used identically to DevSecOps, i.e., application security activities are already built into everything we do, together with development and operations, in smooth cooperation. This co-operation model is shown in Figure 3.

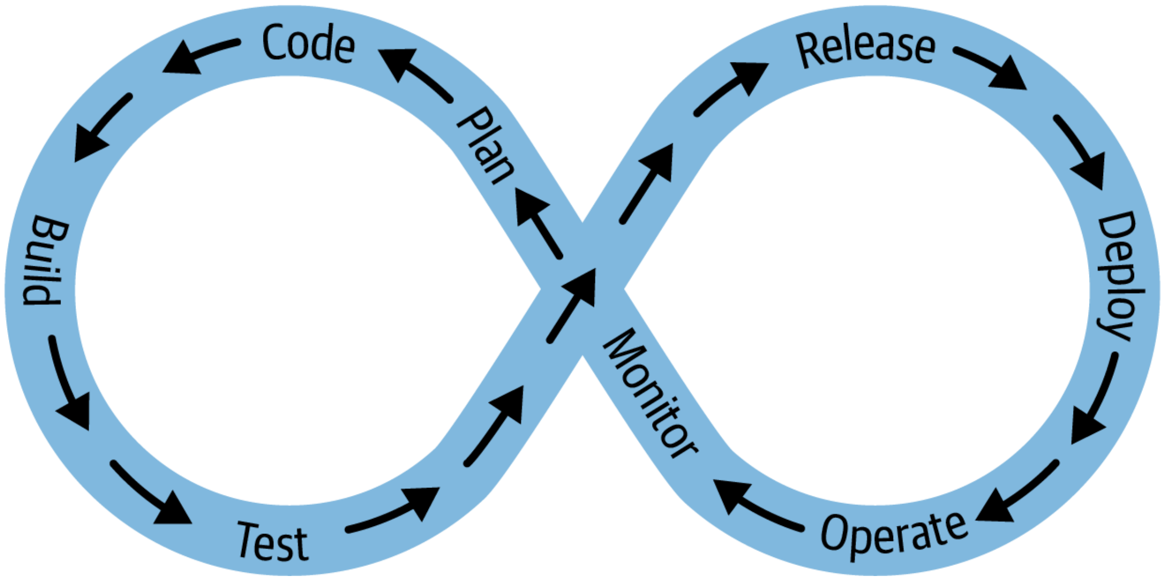


Figure 3. A continuous DevOps practice (Podjarny 2021)

Improving visibility and making everyone's work as transparent as possible is inherent to DevOps. When tools, people and processes work well, we also make our work highly visible to everyone and that enhanced visibility at the same time also improves our security capabilities and maturity. One way to make a team’s work more visible is the use of different DevOps techniques and practices, like visual work boards such as Kanban, as shown in Figure 4, or sprint planning boards where everybody can present their work in either physical or digital cards. The use of different tools and methods always depend on the way a specific team is working. Although it is questionable whether these solutions actually help the organization's transition to agile models, they can still improve the visibility of work tasks, often called as backlog items.

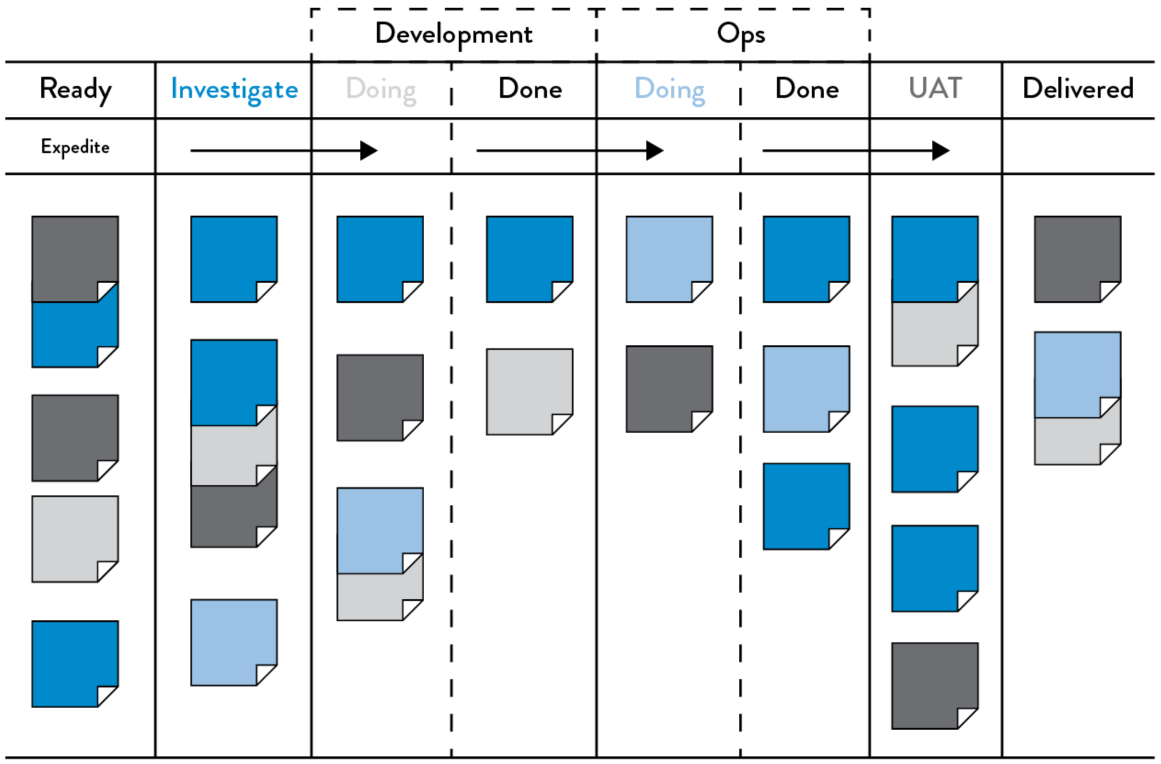


Figure 4. An example of Kanban board (Kim 2016)

3.4 Cloud native

Products that are built using modern software development techniques and technologies like microservices, containers, CI/CD, declarative APIs and immutable infrastructure are known as cloud native applications. The name suggests that these applications are really born in the cloud and are also managed and maintained there as they try to leverage the cloud capabilities to the max.

According to Podjarny (2021), it should be noted that with cloud native applications, the development teams have greater responsibility for the different layers of application development. Layers such as open source libraries, application code, cloud or SaaS services, and containers are the responsibility of the development team. Before, when development and operations were divided more siloed way into application development and IT operations, the responsibility model looked very different. As shown in Figure 5, older applications are made up mostly of code and open source libraries. They also rely on either centralized IT or maintenance team that supports them with infrastructure, such as hardware, VMs, network access etc. Now technologies like Infrastructure-as-code (IaC) changes that approach entirely as teams would spin up the environments from IaC file templates.

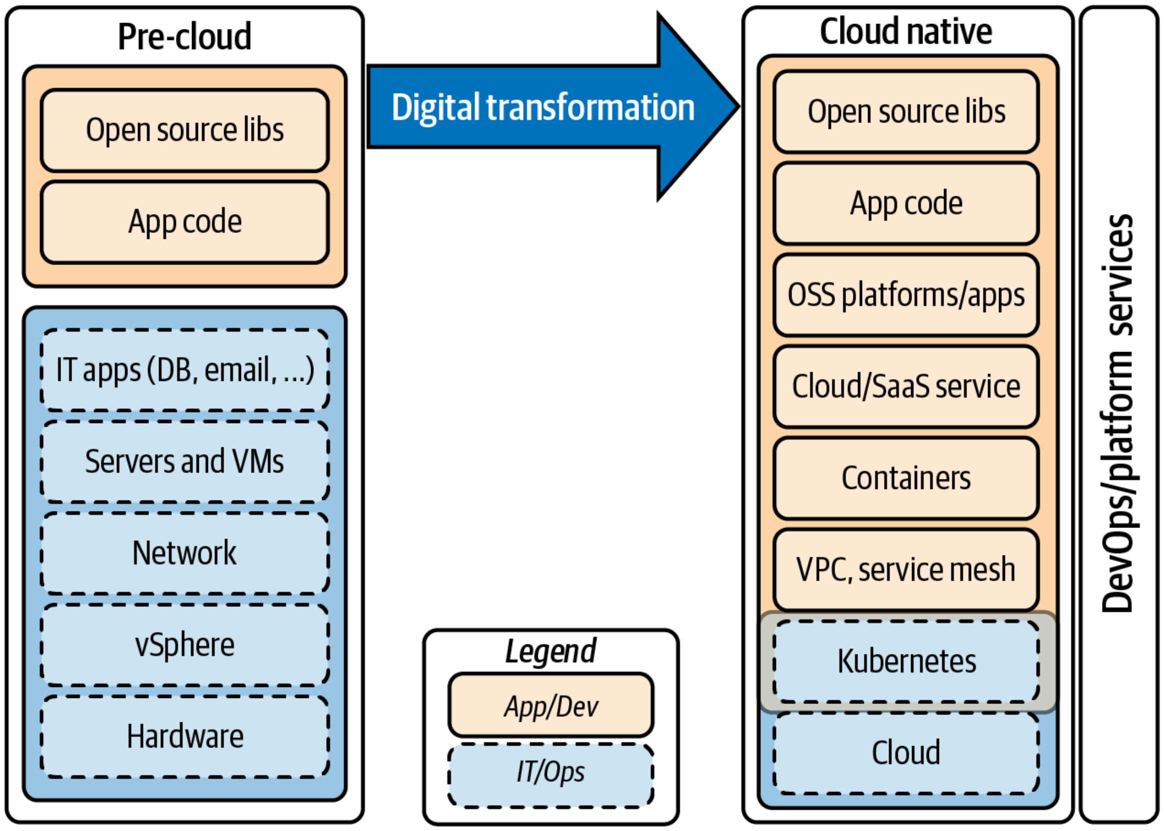


Figure 5. Cloud native and pre-cloud comparison (Podjarny 2021)

3.5 Secure Software Development Framework (SSDF)

The purpose of secure software development practices is reducing the amount and severity of vulnerabilities in software development lifecycle and gain the means to catch the security issues early in the lifecycle before the software is released. The Secure Software Development framework is very important set of practices that organizations should follow and implement. It also provides a common language that all practitioners in the field can understand and follow the same terminology. Several DevSecOps tools use SSDF as a help when checking against a certain reference framework, whether it is SAST, DAST or Threat modeling tools. National Institute of Standards and Technology (NIST) defines their SSDF in the following way (National Institute of Standards and Technology, 2020):

The Secure Software Development Framework (SSDF) is a set of fundamental, sound, and secure software development practices based on established secure software development practice documents from organizations such as BSA, OWASP and SAFECode. Few software development life cycle (SDLC) models explicitly address software security in detail, so practices like those in the SSDF need to be added to and integrated with each SDLC implementation.

3.6 Security metrics

Security metric is a system that allows us to compare something against a standard. It can be defined as a measurement in relation to one or more points of reference in addition to which "metrication" can be defined as the process of selecting and applying metrics to improve the management of something (Brotby & Hinson 2016). These metrics can be used to improve information security for strategic or tactical reasons, and for compliance when there is a need to act in accordance with certain security frameworks. They help various different stakeholders to understand the security situation and awareness as well as the implementation of controls. (Brotby & Hinson, 2016:18-23.)

Some suggestions for security metrics can be:

- Mean-Time-to-Detect and Mean-Time-to-Respond
- open security findings
- number of systems with known vulnerabilities
- number of policy violations
- % of systems with formal risk assessment
- % of systems with tested security controls
- number of identified risks and their severity
- average days to resolution
- severity level of vulnerabilities

- code metrics
    - lines of code
    - source files
    - code churn
    - finding density

The research aimed to find out reasonable metrics that would benefit different stakeholders and improve the visibility of the application security posture. It is important to understand the purpose of collecting metrics and for which groups certain metrics are interesting and useful.

3.7 Standards and best practices

In the industrial sector, cybersecurity operations are governed by several different standards and frameworks. Especially when it comes to IoT devices, it is important to stand out from competitors by certifying products according to certain standards. Some of the standards are also forced through regulation, depending on where in the world the company operates. The visibility of the certification process to information security operations and activities is very important, and without that visibility, it is difficult to show during the audit whether the company meets the requirements set by the standard or not.

3.7.1 NIST DevSecOps Framework

The US National Institute of Standards and Technology (NIST) provides the standards, frameworks, and information security guidance for public organizations, private companies and businesses any size. NIST has several different security frameworks and guidelines, especially comprehensive cybersecurity framework, which is presented in Figure 6. This work mainly focuses on its DevSecOps framework and best practices that are presented in it. In their Computer Security Resource Center (CSRC) NIST has DevSecOps framework explained as follows (National Institute of Standards and Technology, 2021):

DevOps brings together software development and operations to shorten development cycles, allow organizations to be agile, and maintain the pace of innovation while taking advantage of cloud-native technology and practices. Industry and government have fully embraced and are rapidly implementing these practices to develop and deploy software in operational environments, often without a full understanding and consideration of security.

The framework also defines four different value adding capacities (National Institute of Standards and Technology 2021). DevSecOps:

- **reduces vulnerabilities, malicious code, and other security issues** in released software without slowing down code production and releases. This is an important factor when building a security paved road in the organization.

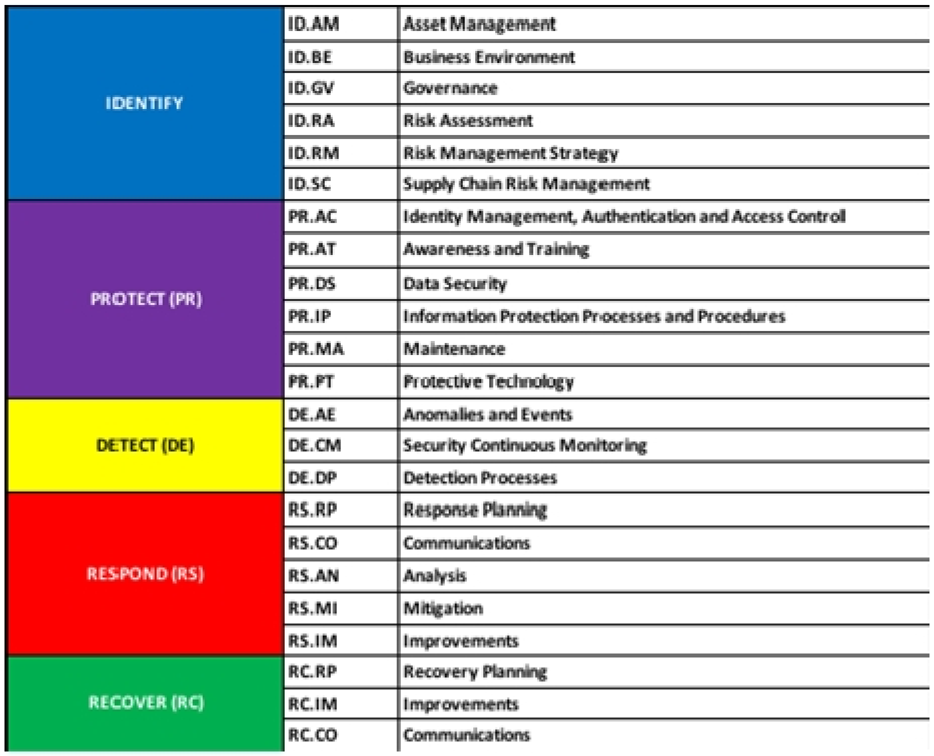
- **mitigates the potential impact of vulnerability exploitation throughout the application lifecycle**, including when the code is being developed and when the software is executing on dynamic hosting platforms.

- **addresses the root causes of vulnerabilities to prevent recurrences**, such as strengthening test tools and methodologies in the toolchain and improving practices for developing code and operating hosting platforms. This is very important in the transition to security automation and DevSecOps. Finding root causes helps to configure tools and pipelines so that only the most important and relevant findings end up in the developers' backlog.

- **reduces friction between the development, operation, and security teams** to maintain the speed and agility needed to support the organization’s mission while taking advantage of modern and innovative technology.

In addition, NIST has published many other guides such as:

- NIST SP 800-82; a very extensive document for securing ICS (Industrial Cyber Security)
- NIST SP 800-53, a database for security and privacy measures and controls
- NIST SP 800-30, guidelines about risk assessment for IT systems



IDENTIFY	ID.AM	Asset Management
	ID.BE	Business Environment
	ID.GV	Governance
	ID.RA	Risk Assessment
	ID.RM	Risk Management Strategy
	ID.SC	Supply Chain Risk Management
PROTECT (PR)	PR.AC	Identity Management, Authentication and Access Control
	PR.AT	Awareness and Training
	PR.DS	Data Security
	PR.IP	Information Protection Processes and Procedures
	PR.MA	Maintenance
	PR.PT	Protective Technology
DETECT (DE)	DE.AE	Anomalies and Events
	DE.CM	Security Continuous Monitoring
	DE.DP	Detection Processes
RESPOND (RS)	RS.RP	Response Planning
	RS.CO	Communications
	RS.AN	Analysis
	RS.MI	Mitigation
	RS.IM	Improvements
RECOVER (RC)	RC.RP	Recovery Planning
	RC.IM	Improvements
	RC.CO	Communications

Figure 6. Structure of the NIST Framework (NIST 2021)

Many organizations and teams use NIST frameworks even without knowing it, because a large part of the security tools map security findings according to the NIST framework. The different test cases that the tools perform are based on these frameworks and the best practices presented in them.

3.7.2 ISO 27001

The ISO27000 family of standards is very well known and respected in the world, which is why companies often aim to achieve ISO certification for their selected products. The standard is very comprehensive, and the preparation of the certification is often a very tedious project. In the commissioner organization, an ISO 27001 certification project was carried out for a large product family, as part of which application security visibility had to be improved, in order to know at what level the security capabilities and maturity of the product and teams were and what would be the possible problem areas that should be addressed well in advance of the audit. Information Security Management System (ISMS) is an essential part of ISO27001 implementation. The purpose of ISMS is to create risk management processes that ensure that risk, continuity, and information security management is at an appropriate level in the organization. This management system contributes to improving visibility and processes for secure working methods, which is not a one-time project, but a process subject to continuous change.

According to Calder (2020), the idea of ISO27001 certification process is to adopt controls relevant to you and for your organization. Those controls are divided in 14 different categories which are:

1.A.5 Information security policies

2.A.6 Organization of information security

3.A.7 Human resource security

4.A.8 Asset management

5.A.9 Access control

6.A.10 Cryptography

7.A.11 Physical and environmental security

8.A.12 Operations security

9.A.13 Communications security

10A.14 System acquisition, development and maintenance

11.A.15 Supplier relationships

12.A.16 Information security incident management

13.A.17 Information security aspects of business continuity management

14.A.18 Compliance

These 14 control categories consist totally of 114 different security controls that organizations can apply to improve their security posture. On a general level, an organization implementing the ISO 27001 standard must implement an information security management system, as well as maintain and improve it continuously in accordance with the requirements. This implementation must be easily demonstrable when requested by the auditor.

3.7.3 IEC 62443

Another important standard in the field of target organization is IEC 62443. As described by Flaus (2019), it consists of a set of instructions, requirements and guidelines intended for security personnel in the development and life cycle of industrial systems. Those requirements and guidelines are designed for people having responsibilities implementing them in the industrial automation and control systems’ (IACS) lifecycle and its different phases, from design and implementation to management. These people can have roles such as system integrators, system users, operators, or product suppliers. Mentioned roles and responsibilities are visually illustrated in Figure 7.

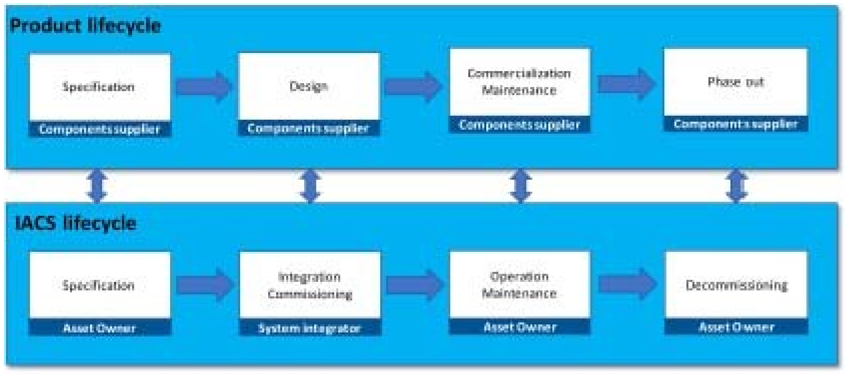


Figure 7. IACS lifecycle (Flaus 2019)

The purpose of the standard is to ensure the safe use of industrial automation systems. It has a sub-part 62443-4 which in general describes the requirements of product development, and system requirements for components but also includes a standard for secure product development lifecycle requirements. It specifies a set of secure development lifecycle requirements related to industrial automation and control systems environment, and also guidelines to meet the very same requirements. It is very important that the implementation of these controls and requirements are well visible because they are a key matter for the safety and security of the organization's products.

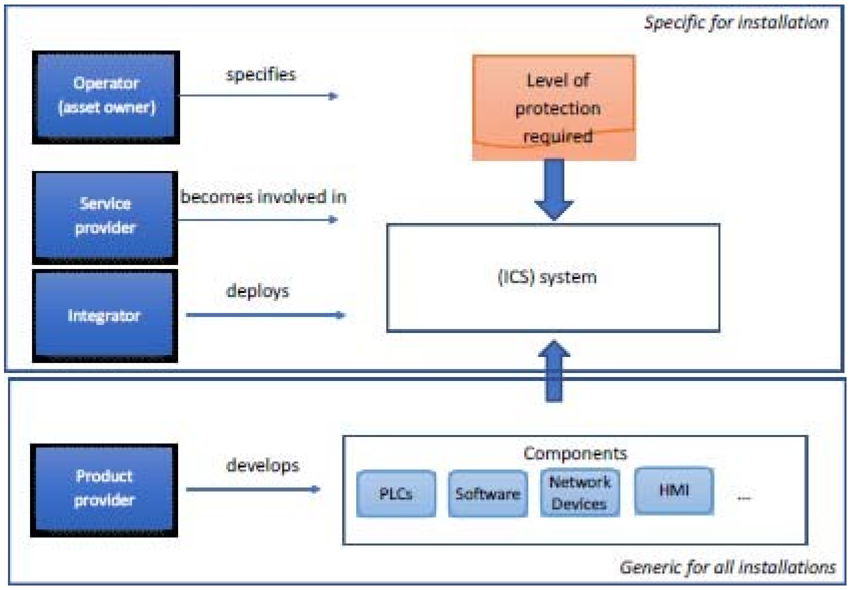


Figure 8. The target audience for the standard (Flaus 2019)

3.8 Automated security testing

In this context automated security testing, often called as an acronym AST, techniques refer to the security testing practices in the CI/CD pipeline. One of the key benefits of automating security testing is that it frees up time for actions that require human intervention, but also the fact that with the help of automation we can achieve a real time situation, where we have good visibility into the state of our application security posture. Automating different application security activities such as security testing in CI/CD pipelines, automated threat modeling and requirements management gives us improved visibility to application security status and that visibility helps organizations secure and understand their applications throughout the whole life cycle. In Figure 9, different testing methods are visualized during the software development lifecycle.

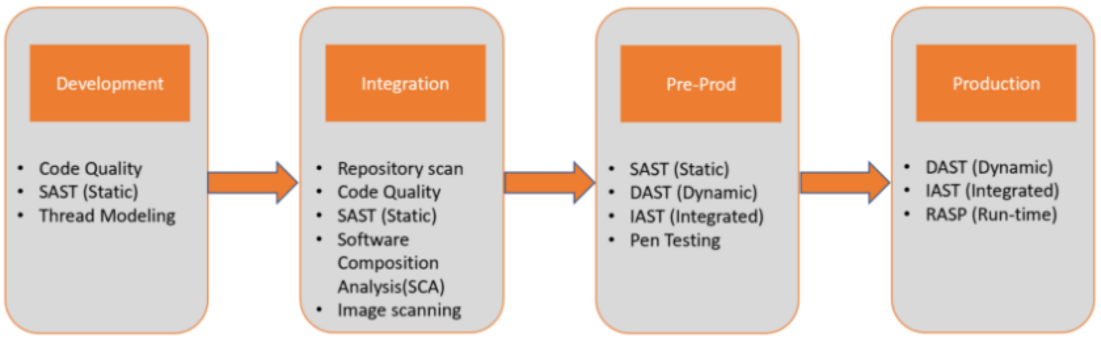


Figure 9. Different phases of security testing (Hsu 2019)

According to Hsu (2019) the purpose of security automation is to reduce the manual workload where human logic is not needed and save resources where they are most useful. Different test automation methods in the CI/CD pipeline are an excellent way to get a large test coverage in an application with a scalable way, without increasing the burden on people too much. Potential security flaws can exist anywhere, from the source code and third-party components to an insecure configuration or vulnerable infrastructure. In security test automation, it is necessary to be able to shift the automation focus where people and manual steps are not needed. As shown in Figure 10, use of automation transfers limited resources to fundamental analysis, understanding the security impact and internal communication in the organization, to get the greatest possible benefit from the findings of the automated tools.

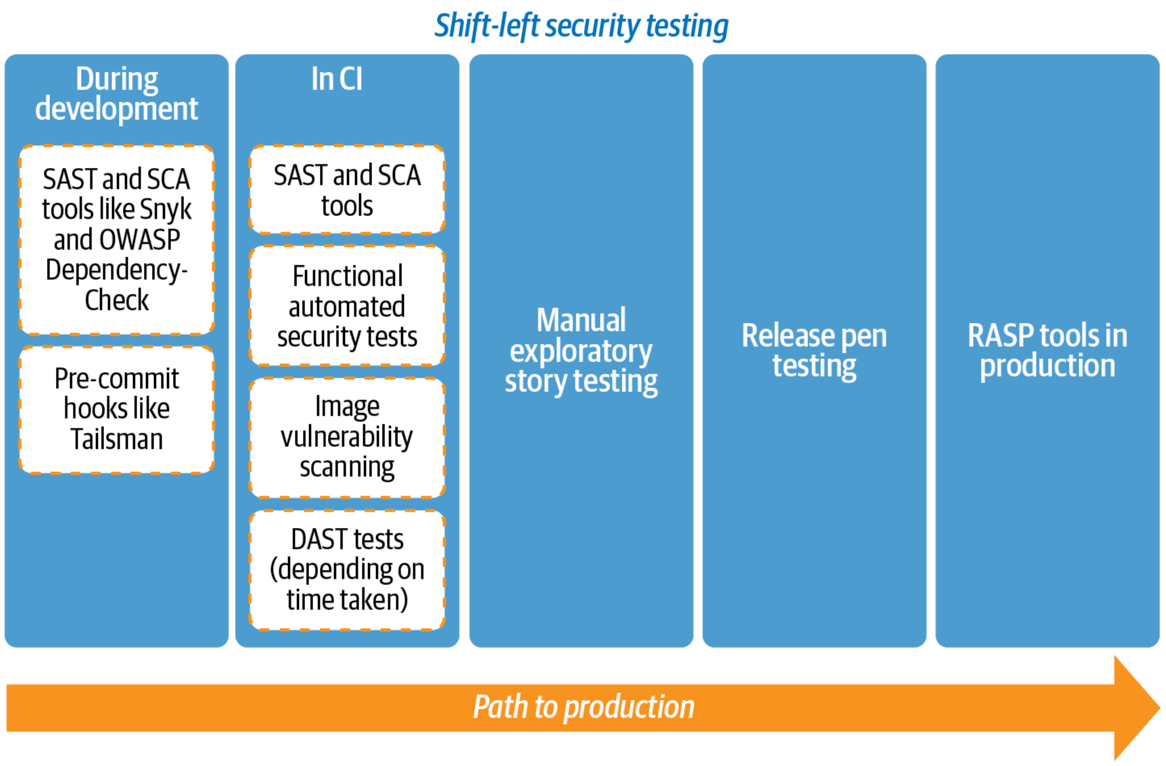

Shift-left security testing
During development
SAST and SCA tools like Snyk and OWASP Dependency-Check
Pre-commit hooks like Tailsman
In CI
SAST and SCA tools
Functional automated security tests
Image vulnerability scanning
DAST tests (depending on time taken)
Manual exploratory story testing
Release pen testing
RASP tools in production
Path to production

Figure 10. A shift-left security testing strategy (Hsu 2019)

3.8.1 Continuous Integration

Continuous Integration (CI) is a software development practice, and its purpose is to automate the delivery of changes of several different developers to the code base and centralized version control system. With CI, different team members can work on different things in parallel and combine new features when necessary, so that the whole application remains intact. The goal is to have a up and running software the functionalities of which each member of the development team can develop, change or modify without breaking the system and losing functionality. Continuous Integration often contains automated tests when changes are added to a larger code base. When talking about CI practice, the most well-known CI technologies and tools are also often discussed, such as Travis CI, Jenkins, Circle CI and GitLab CI.

3.8.2 Continuous Delivery and Continuous Deployment

CD in CI/CD means either Continuous Delivery or Continuous Deployment. The main difference here is the level of automation. As seen in the Figure 11, the continuous delivery allows development teams to push any code changes automatically to certain prepared state for a release to production. This can be non-production testing, or staging environment. It expands continuous integration by deploying any code changes to a testing environment and always having a deployment-ready build artifact that has gone through all the stages and tests in CI/CD process. Those tests can include not only UI testing, load testing, and integration testing but also automated security tests. Where continuous delivery requires a manual approval before changes are pushed into production, continuous deployment aims to automate the whole pipeline. Without manual human intervention, deployment to production happens automatically without explicit approval. (Amazon Web Services 2022.)

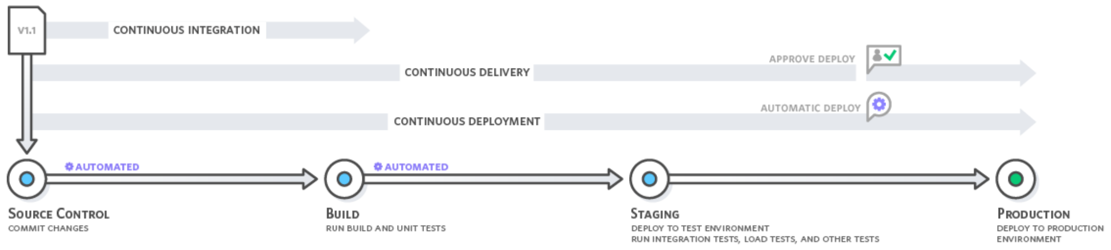


Figure 11. Continuous Delivery vs. Continuous Deployment (Amazon Web Services 2022)

3.8.3 SAST

One of the first automated CI/CD security technologies was SAST (Static Application Security Testing), also known as automated source code analysis. A SAST tool called Coverity has been implemented in the target company, and the goal is to use Coverity to scan the source code of every SRD application, automatically in the CI/CD pipeline, giving more visibility to security errors in the source code. One of the key benefits of SAST-tools is that they are relatively easy to setup and these scans usually do not take too much time to finish, making them ideal tools for agile teams and CI/CD pipelines. The downside of these tools is that they frequently produce false positive findings. SAST tools run security scans against source code, byte code, and assembled code for known vulnerabilities. SAST tools come in various forms, including plug-ins, libraries, and SaaS solutions (e.g., Snyk IDE plug-ins, Checkmarx SAST, Security Code

Scan), and can be integrated with CI pipelines natively to run against every commit. It is also possible to use SAST tools as part of an IDE (Integrated Development Environment), in which case the tool's plugin is installed directly in the code editor, and security checks are run at the same time as the application is being developed. This is the earliest stage when technical security testing can be brought into the software development lifecycle. SAST is a big part of shifting security left, as it helps you discover issues during development. (Gayathri 2022.)

3.8.4 DAST

DAST, i.e., dynamic application security testing, is a method of testing applications when the software is first built and a functional, running application is tested for various software security vulnerabilities. It differs from static testing in that, unlike with SAST scanners, there is no access to the source code, but dynamic testing examines the behavior of the software and how it responds to various inputs from the user.

Dynamic Application Security Testing (DAST) technologies are one of the most popular ways to ensure the security of products during application development. In this testing method, the security testers, examine the application while it is "up and running". It usually must be run in the test and production environment of the application in order to have most or all of the functionalities available. Dynamic security testing is part of black box testing, meaning that the tools try to study the software's behavior and response to simulated attacks by the testing tool. Based on the application's responses, the DAST tool aims to determine whether the application could contain security issues and be vulnerable to real cyberattacks. Due to the black box methodology, the DAST scanner doesn't really need to access the software code and in-app functions. In fact, it seeks to automate what a hacker would do in the "live situation" without excessive inside information about the target. Thus, finding a vulnerability usually means that the vulnerability can indeed be exploited. This also distinguishes dynamic security scanning from static application security testing (SAST) in that it does not produce as many so-called "false positive" findings.

When talking about DAST products, it is good to clarify what it means in which contexts. Dynamic testing can be talked about, for example, when performing largely manual penetration testing on the target system and launching a vulnerability scan with a testing tool such as Burp Suite or Nmap by a tester. In general, DAST is more often talked about nowadays when it refers to an automated security audit in the development phase of an application, e.g., in a CI/CD pipeline. An example workflow might look like it’s shown in the Figure 12.

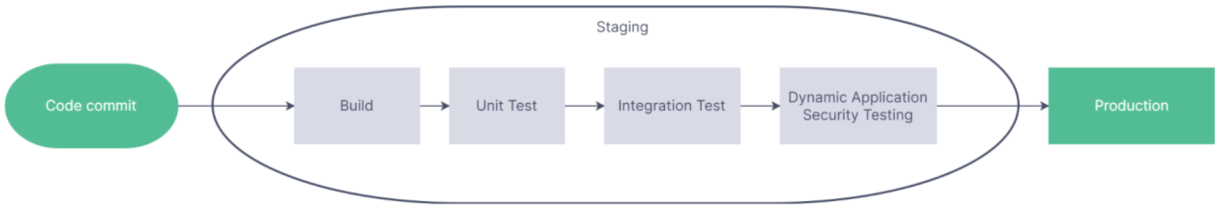


Figure 12. Example of DAST workflow

DAST is not a good solution for CI/CD environments in every situation. The downside of dynamic automated testing is that large-scale testing of an application can take several hours, making it very poorly suited to a DevOps workflow that is agile in nature. If DAST tools are intended to be used in the CI/CD pipeline, the scan settings should be configured carefully, so that the test stage includes only the essential and most important checks, while taking as little time as possible to complete.

3.8.5 SCA

SCA (Software Composition Analysis) is a testing method, with the purpose of checking the open source and third-party components used by the application and any known vulnerabilities hidden in them. SCA scanning is usually performed automatically in the CI/CD pipeline before the build phase of the application. SCA scanning can also be performed using separate plugins directly in the application development environment, i.e., IDE, or manually from the command line or the scanner's user interface. Similarly to SAST testing, SCA testing is a static method, and therefore does not require a running application to run the tests.

It is generally thought that SCA testing is the easiest automated activity to perform in a secure application development lifecycle and in a CI/CD environment, so it is often also the first step to automating application security testing. It has also been estimated that third-party components contain most of the vulnerabilities in the application, so it is logical to start such a testing activity where you can get the most benefits right from the start.

In terms of the security visibility, the checks and scans made by SCA tools are vital because they give visibility into which components the applications consist of. Most commercial SCA tools include the ability to generate an SBOM (Software Bill of Materials). SBOM is practically a "recipe" used for the application, i.e., a list of all the ingredients that the application consists of. Target company has implemented an SCA tool called Black Duck in SRD. The goal is that static composition analysis is connected to every application in their CI/CD pipelines, and their 3rd party components are scanned for known vulnerabilities automatically. This creates visibility into how applications are built, what their Software Bill of Materials is, and what different libraries they use. The majority of all application security vulnerabilities are reported from SCA tools, so this is an important view to have.

3.8.6 IAST

Interactive Application Security Testing (IAST) is a slightly more modern testing technology in the field of application security. When implementing IAST, an agent is installed in the environment that aims to investigate the normal use of the application and inject various tests to find vulnerabilities and security misconfigurations in the software's runtime operation. Due to its nature, IAST is capable of dynamic testing but at the same time able search for the root causes of vulnerabilities in the code base. According to Hsu (2018), IAST not only does DAST security testing but also can identify the root/cause at the source code level via a RASP Agent. In simple terms, IAST = RASP Agent + DAST.

Some of the advantages of IAST:

• Integration into the IDE enables earlier detection of vulnerabilities.

• Because of the detailed information available to the sensors, IAST can pinpoint sources of error accurately

• Because of the integration in the IDE, IAST can be part of agile development and the continuous integration/continuous deployment (CI/CD) pipelines.

IAST is the application security tool that most requires a genuine security culture in the organization. Especially from quality assurance (QA) team, they must have a real desire to also start security tests as part of the QA teams' manual and automated testing. IAST will not find anything if the application is not used as it is intended to be used, thus it is suitable for use side by side with, e.g., unit and integration tests, but not very well for use by the application security team. (Hsu 2018.)

3.8.7 External Attack Surface Management

External Attack Surface Management (EASM) is a technique that aims to map the attack surface and discover organization's Internet-facing assets for vulnerabilities. Visibility into assets and their proper inventory are key elements in securing applications, and EASM technologies can be used to better manage what the organization's applications visible on the Internet expose. Poor understanding of the organization's public-facing assets and the visibility they offer to outsiders can lead to serious exploits, such as severe subdomain takeover. EASM techniques aim to get answers to questions such as:

- What Internet-facing applications does the organization own?
- What vulnerabilities and anomalies do they contain?
- Where should the organization focus its attention?
- How are the identified vulnerabilities concretely fixed?

EASM solutions make it easier for security teams to focus on practical threats in the secure software development lifecycle. The technology balances testing throughout the life cycle, which is often focused on the left side, so to speak, i.e., the early stages of the life cycle.

3.8.8 AVC

Application Vulnerability Correlation (AVC) is a centralized vulnerability management system that gathers and combines security defects from different sources into a single security dashboard. Different sources of security defects can be static testing tools, dynamic scanners, IaC tools or penetration testing and auditing. In addition to the fact that AVC tools are able to collect various security defects from several different sources, with AVC it is also possible to build scanner integrations in a way, that management and orchestration of different tools is possible from the same platform.

3.8.9 Tool evaluation

When trying to solve problems regarding to improve application security capabilities and the visibility obtained with security tools, the whole toolchain and their markets should be evaluated regularly. Every security tool category consists of tens or even hundreds of different tools, and it is often difficult to understand what the best solution for a specific problem, team, or an organization at large scale would be. When trying to improve security visibility with the help of these tools we need to focus on how well they integrate to other critical platforms we use in regular basis. According to Hsu (2018) there are a few different considerations when evaluating DevSecOps toolchain:

- Usability. The target users of the code scanning tools are developers. The usability includes the capability to scan parts of the source code, differential scans, scanning reports, tracing back to original source code, and so on.

- Budget. If it is an IDE plugin commercial tool or for example SaaS based tool we need to consider how many concurrent users' licenses it will need.

- Programming languages support. We can do a survey of the programming languages used by in-house projects and prioritize the programming languages that are going to be supported

- Detection rate. It is common for any scanning tools to have false positive rates, depending on the scanning engine and rules. A high false positive is not a bad thing, and it can also mean the scanner takes a more conservative approach. Find the tool that best fits the projects instead of the most well-known. To evaluate the detection rate, we may use known vulnerable projects.

- Scanning rules update. It is important that the tool is constantly updated with rules and scanners. One of the key advantages of a commercial tool is that the tool will have up-to-date scanning rules.

- API capabilities. With modern application security tools, it is important to be able to communicate and integrate with them using APIs. The management of application security tools is a matter of whole chain of different solutions. Therefore, it is important, that the different systems are able to talk to each other, and the tools support the existing workflow, not the other way around.

3.9 Automated vulnerability management

One of the most important elements of any application security program is design of the good vulnerability management process and make sure that those vulnerabilities are triaged properly. Some of the clear advantages of automating vulnerability management are the following:

- It helps in driving security and compliance programs by bringing enhanced visibility of security status to different stakeholders.
- It helps in prioritizing security defects from different sources, based on for example the Common Vulnerability Scoring System (CVSS).
- It provides greater visibility to effectiveness of any security program.
- It improves visibility in real time with the faster feedback from CI/CD.
- Improved visibility provides needed proof to both internal and external auditors.

With automated vulnerability management it is possible to programmatically compile the data sources to provide effective vulnerability prioritization, validation and enhance visibility to application security status and defects. With this method we can save time for more important work that requires human intervention, for example improving organization’s security processes rather than focusing on vulnerability data dumps that can be easily observed by a machine. (Magnusson 2020.)

4 WAYS TO IMPROVE VISIBILITY

There are many ways to improve security visibility to certain targets such as products, development teams, processes, and cultural aspects. In the target company there were several different practices to enhance the visibility in the application security during the study, which will be covered in this chapter. Those ways included improving security toolchain and vulnerability management processes and technics, automated security requirements management and communication throughout the organization to improve security awareness between the security team, product development teams and other important stakeholders.

4.1 Automated security requirements management

The management of information security requirements is an important and often demanding task in any organization, and especially in a large international organization that may operate in several countries or continents, in which case the information security requirements should also take national legislation and standards into account. When talking about an organization's ability to comply with certain information security requirements, it often means the ability of several units, teams and products to meet these requirements. To see if the requirements are met for a specific application, product, unit or the entire company, visibility is needed into the information security status of several different parts of the organization. This requires good management of information security requirements.

The automation of information security requirements helps in such a situation to achieve the goal and can even create an up-to-date view of the information security status of different parts and products of the organization. In the case of the target organization, a platform called IriusRisk was used to automate the management of information security requirements and provide better visibility to commissioner’s application security posture. The automation of security requirements also makes sense when the organization develops digital solutions with several different software development frameworks, for different customer segments and possibly on a different cloud platform. Therefore, the exact security requirements against a certain framework can be retrieved automatically and the security team does not necessarily have to maintain enormously extensive inventory of documentation on how to use certain individual component or framework securely. An architecture image is created in the platform’s draw.io canvas, which contains the product's technical components. Figure 13 visualize the architecture and AWS components of a simple cloud native web application. Based on these components, the solution gets exact security requirements, which can be integrated again with different backlog tools, that the development teams use.

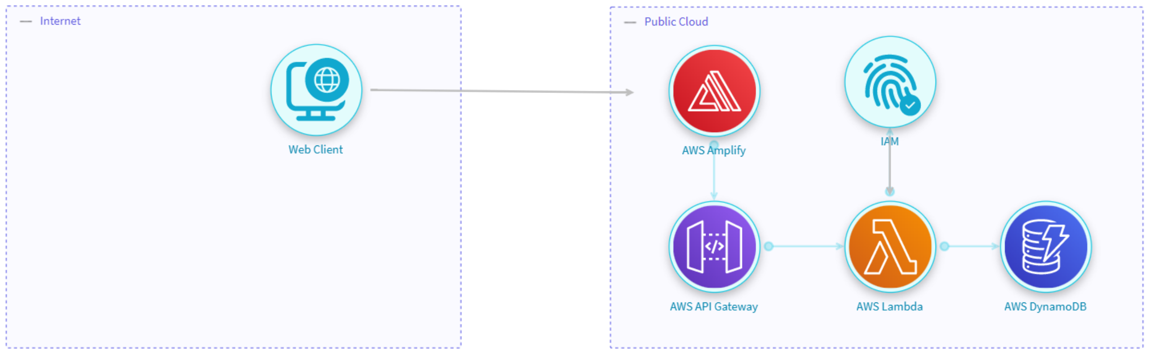


Figure 13. IriusRisk simple web application diagram

4.2 DevSecOps maturity assessment

The purpose of DevSecOps maturity assessment is to gain an overview of used technologies, tools and workflows in a particular team or a group of teams. The assessment helps understanding on what level of maturity a particular team or a group of teams have achieved in their DevSecOps practices and what are the key areas to be focused in the near future to improve the maturity and the security posture of a certain team and its product. Usually, a maturity assessment consists of 15 to 30 different questions divided into a few different categories to which the teams would give their answers and it can be done as a self-assessment. In the target organization, a maturity assessment was carried out for 27 different software development teams in order to improve security visibility and help the security team better understand the issues that the development teams need help with. After the questionnaire is completed and the answers are analysed the next step is building an action plan for improvement. (GitLab, 2020.)

4.3 Achieving full-stack visibility from development to production

Regarding testing tools, the plan is to create a unified pipeline that covers different types of tests and activities from the initial stages of development to production. The task of the testing pipeline is to cover all phases of the software development life cycle and, in addition, the entire attack surface of the product. This security testing pipeline cannot give very good visibility as it is, but for that purpose to improve visibility there is a need to create a separate platform from which the orchestration of tools takes place and security defects are mapped centrally. The commissioner has chosen CodeDx as a centralized vulnerability management and correlation system, which is currently owned by Synopsys.

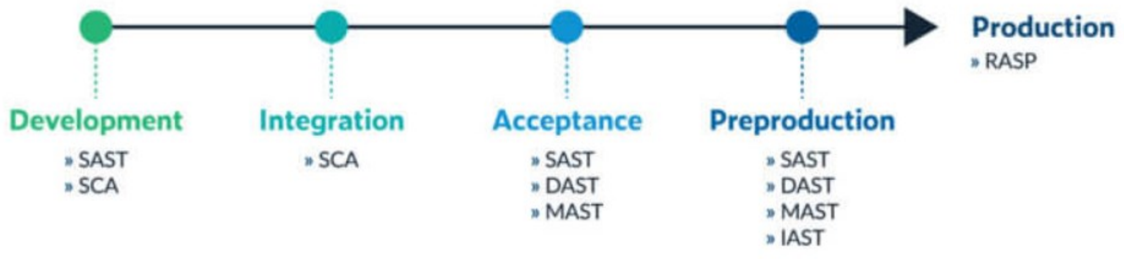


Figure 14: Security testing pipeline

With the help of the CodeDx platform, the overall visibility of the commissioner’s digital products can be significantly improved in terms of application security. The platform offers ready-made dashboards from which, with the help of graphic illustrations, the trend and evolution of application security defects can be seen over time. The platform collects data from different sources of security defects, such as SAST, SCA, DAST, penetration testing or bug bounty platform and compiles the data into one dashboard either for one team or application, or it can be viewed for the entire organization if there is a need to get a security view, for example, for the top management. Potentially, security visibility can be improved enormously with such systems, but the prerequisite for the effective use of these platforms is that the defects have first been triaged and it has been ensured that there are no so-called false positive findings that can easily distort data with dashboards.

4.4 Assessing DevSecOps objective and key results for development teams

The commissioner is undergoing an extensive transition to DevSecOps, and as part of this transition, certain OKRs were defined for the application development teams, the purpose of which is to get all teams to improve their cybersecurity capabilities and also to improve visibility regarding cybersecurity activities between different stakeholders. These OKRs included goals such as the introduction of basic cyber security trainings to the teams, the introduction and implementation of automatic security testing tools, sharing individual team's own operating models, used techniques and technologies and workflows with others and improving vulnerability management process.

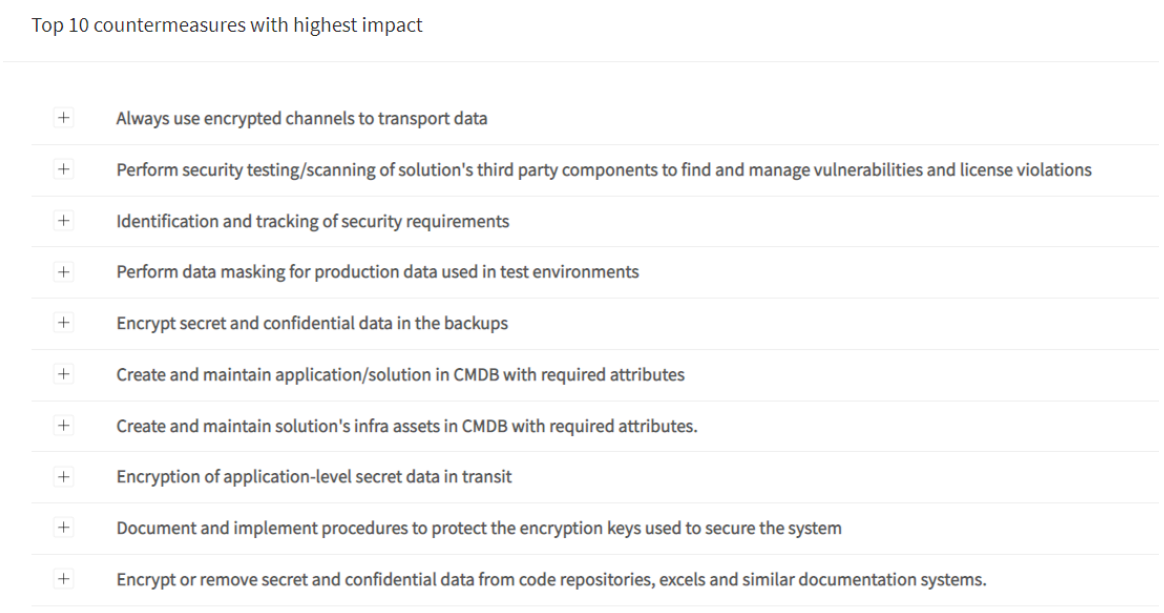
4.5 Improving visibility for ISO27001 project

As the commissioner's target has been to obtain ISO27001 certification for a few important digital products, and in the preparation of this certification process, it has been essential to improve the visibility of the teams' security activities, workflows and working methods. When evaluating working methods, cybersecurity plan was divided into different domains, which were

- application security
- asset management
- cloud security
- continuity
- governance
- human resource security
- identity & access management

- information protection
- information security event management
- network security
- secure configuration
- supplier relationships security
- threat vulnerability management

It is crucial in ISO 27001 certification to provide visibility into how comprehensively the information security guidelines and processes are defined. For this purpose, a platform called IriusRisk was used. IriusRisk is known for its ability to automate threat modelling but also to manage information security requirements in partly automated manner. Figure 15 shows how different countermeasures with the highest security impact are listed on the IriusRisk platform.



Top 10 countermeasures with highest impact

- Always use encrypted channels to transport data
- Perform security testing/scanning of solution's third party components to find and manage vulnerabilities and license violations
- Identification and tracking of security requirements
- Perform data masking for production data used in test environments
- Encrypt secret and confidential data in the backups
- Create and maintain application/solution in CMDB with required attributes
- Create and maintain solution's infra assets in CMDB with required attributes.
- Encryption of application-level secret data in transit
- Document and implement procedures to protect the encryption keys used to secure the system
- Encrypt or remove secret and confidential data from code repositories, excels and similar documentation systems.

Figure 15. Example of view of top 10 countermeasures

5 RESULTS

The objective of this study was to analyze the current status of commissioner's application security visibility and find ways to improve application security visibility at the product and also at the company level. A total of two different surveys were carried out in connection with this research. The first of which was a survey focusing on methodologies and working methods and the second survey on how the visibility of application data security could be further improved. Both questionnaires were built using Microsoft Forms tool and Microsoft Excel and PowerPoint was used with analysis and presentation of the research data.

5.1 Methodology assessment questionnaire

As one part of this thesis' work, a DevSecOps maturity assessment was prepared to gain information about the baseline of DevSecOps methodologies. Its purpose was to gain visibility into the working methods, technologies used and application security practices of different development teams. GitLab's DevSecOps methodology assessment was used as the basis of this questionnaire, from which the areas and groups of questions were selected to suit our own situation. The survey was divided into three different areas: culture and collaboration, velocity and process efficiency and tools and automation. There were 7-10 questions from each area and a total of 27 application development teams responded to the survey.

The first conclusions from the maturity assessment are that most of the development teams understand the importance of application security and that the company-level information security requirements and policies have been clearly communicated and enforced. Baseline security requirements are considered very important, and the existence of the application security team is known, and help can be requested when it is needed. In their own opinion, the development teams have moved away from the waterfall development model to agile development and DevOps ways of working, which is important for the application security team to understand because there will be more challenges for several application security activities in the software development lifecycle, and the importance of security automation must be taken into account. One aspect that is clearly worrying is, according to the survey, software developers practically spend most of their time implementing new functionalities, and there is not much time left for fixing bugs and technical debt. 22 out of 27 teams answered that they spend more time developing new functionalities than fixing things, for example related to application security. From business development perspective this can lead to profits in the short term, but in the long run, increases the security risks and business continuity to a large extent. This is also understandable because the target company is under great pressure to develop digital products for its customers and the industry is just at the transition of digitalization, so the innovation level is meant to kept very high.

Based on the survey, there is much room for improvement in the level of automation. 17 teams out of 27 answer that deployments and releases are automated either completely or partially, but 20 teams say that tickets are not created for security tests, nor the build is stopped when issues found or DAST analysis is performed after the build. In terms of visibility, a good solution would probably be to have the tasks from the security tests in the same place where the other tasks of the team are and where the teams' backlog is located, so that the security responsibilities and activities are not separated from the rest of the daily work.

5.1.1 Culture and collaboration

This section explains how much different teams have adapted DevSecOps to their practical working culture. It aims to study how well security issues are made known in the team, how important security actions are considered, and whether communication regarding security issues works clearly enough. The cultural aspect also takes a stand on the more demanding and advanced security activities, such as game days and chaos tests, which usually require a particularly active approach and the desire to do application security testing well.

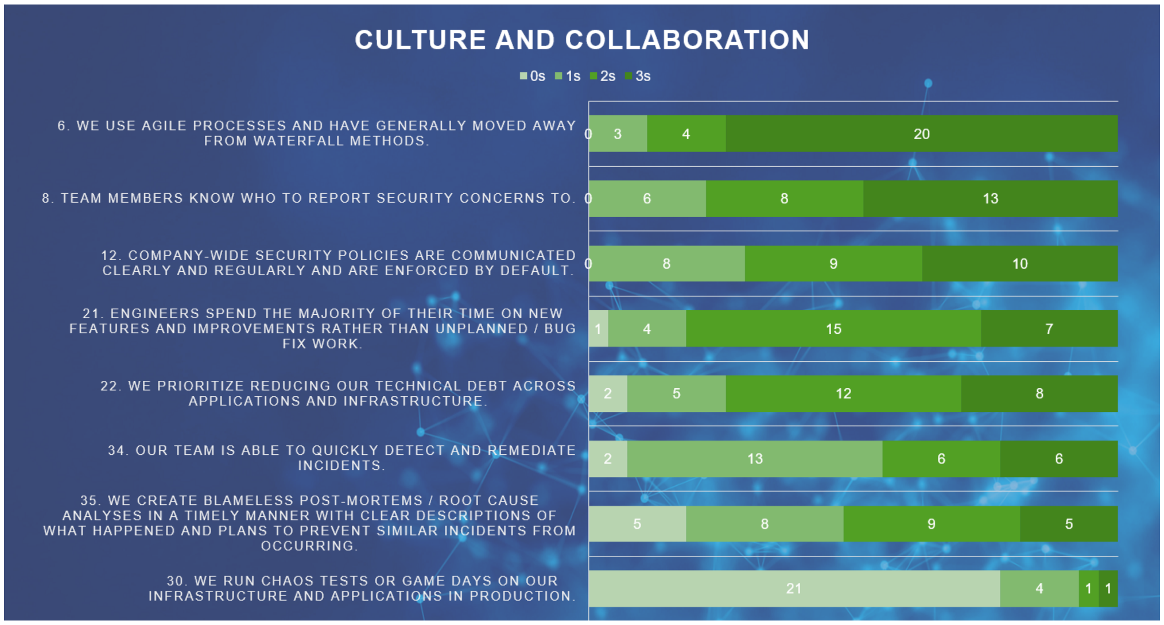


Figure 16. Culture and collaboration

As shown in Figure 16, a clear trend can be seen that teams have moved away from the waterfall model towards agile development methods. Most people seem to have an idea of who to report security incidents or concerns and where to ask for help. Security policies exist and they have been enforced to some extent, although not everyone had accurate information about what these policies are and where they can be found. Most of them have not yet adopted more advanced testing techniques, such as chaos testing.

5.1.2 Velocity and process efficiency

The Velocity & process efficiency category aims to practically measure how far the organization has moved to the left in security testing and secure application development design. The purpose is to investigate whether appropriate application security requirements have been collected for the piece of software, whether threat modeling has been done or static and dynamic security tests have been automated from the beginning in the CI/CD pipeline. It also examines how the processes work in practice and whether security is integrated into the processes in such a way that it remains involved even when new functionalities are introduced in the application.

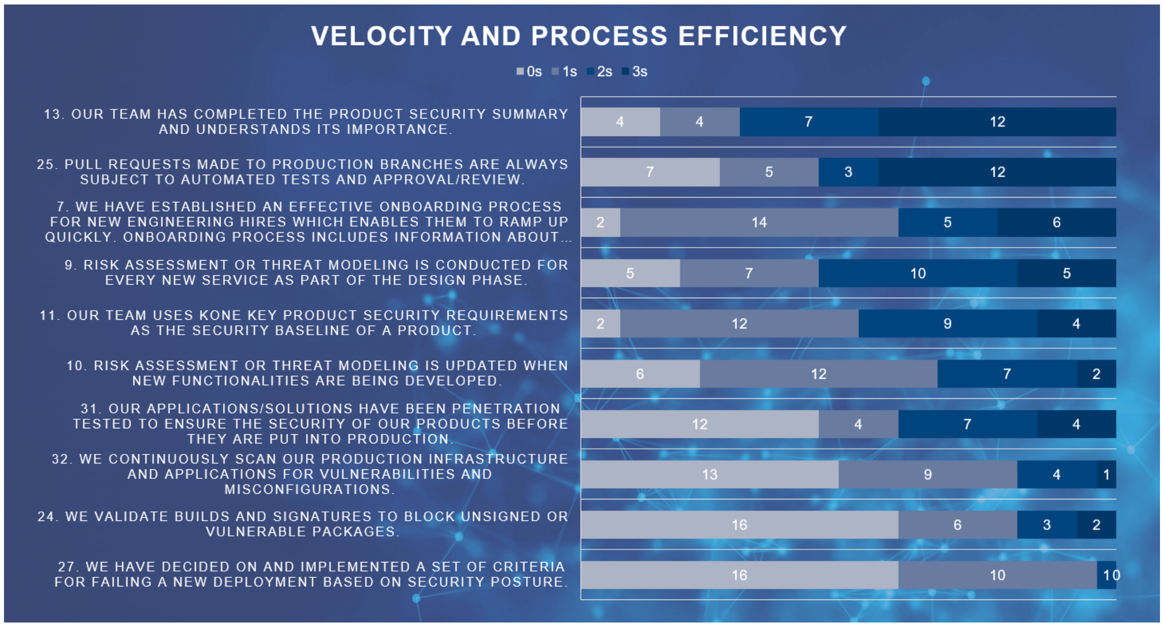


Figure 17. Velocity and process efficiency

From Figure 17 it can be stated that almost every team has some level of orientation process where information security topics and some basic activities from the application security area are also reviewed. Activities such as threat modeling and the collection of information security requirements are done quite actively. One clear point, however, is that the deployment to production is not forced to fail due to security findings, and the infrastructure is not very comprehensively and continuously scanned, at least not for security misconfigurations. Penetration testing is not done at all in half of the cases. At target company, a form called security summary must be filled in at the start of each project. This document is used to ensure certain security steps during development, such as security design, security requirements, risk assessment and security guidelines. It also includes things that may be necessary, depending on the project, such as supplier cybersecurity, which aims to ensure that partners also operate sufficiently securely. Most of the survey respondents had gone through this phase and several even updated the document as needed, which indicates that the security processes work well, at least in this respect.

5.1.3 Tools and automation

This category gathers the findings that have come from measuring the level of tools and automation. The category includes DAST, SAST and SCA tests and measures how automated deployments to production are. In practice, it is about whether continuous deployment or continuous delivery is used.

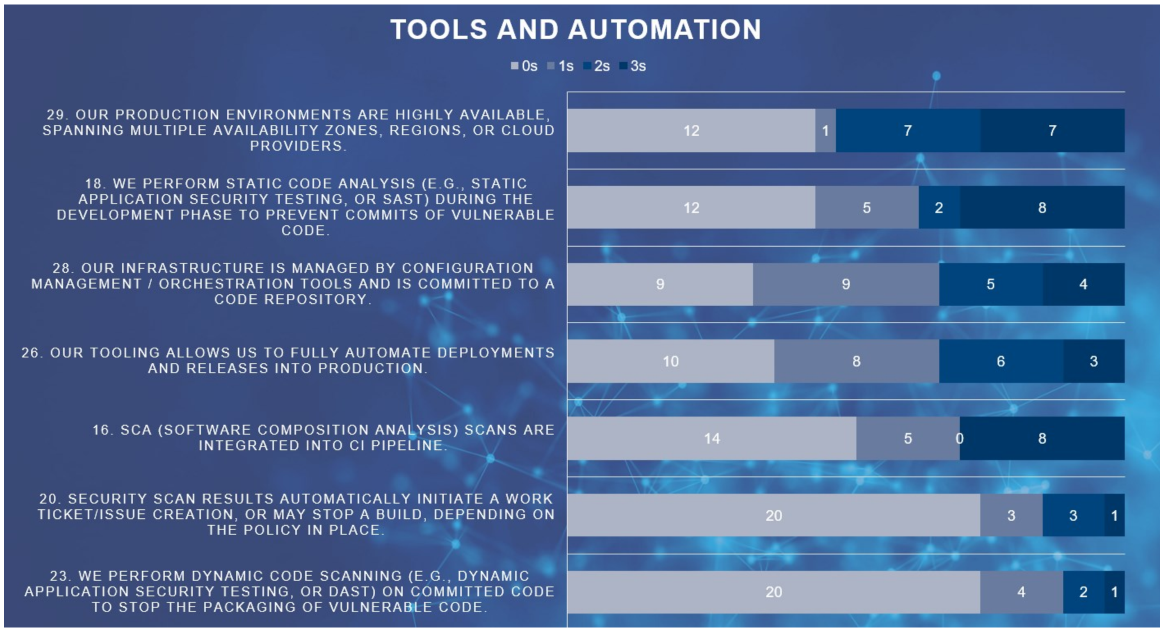


Figure 18. Tools and automation

Here, as shown in Figure 18, the purpose was to find out how automated workflows are in detecting security findings in the tool, and how they end up in the teams' backlog. In practice, the most significant observation from this category was that dynamic security testing is not done basically at all, and no hard failures are used in the pipeline. The reason for this may be that DAST scans, especially in the CI/CD pipeline, require the presence of the security team during the onboarding. To be an effective CI/CD test, the scan must complete its tasks in a few minutes. This requires configuring the scanner in such a way that it can crawl and spider the most important areas of the application and perform tests that are relevant to the target application, but not any additional measures that could delay the completion of the CI/CD pipeline. Also, the creation of tickets for backlogs is not automated, so if teams want security findings to go to the backlog to be fixed, they must be sent there manually. This is possibly the result of the fact that some security scanners give so many false positives and provide findings that are not fully understood. Consequently, if automated, this process could create a large number of tickets that nobody in the development team would not know how to solve.

5.2 Questionnaire for improving visibility

As the second part of this research a questionnaire was prepared to conduct research with the aim of getting views from the target company's key personnel and stakeholders and tackling the research questions presented at the beginning of the thesis in particular:

- *RQ1: Which factors should be considered effective to improve application security visibility in a large organization?*

  The first research question was set as trying to find important factors that affect visibility in application security and how things could be improved. This also requires an investigation of the current state of the organization in order to gain a generic understanding of how application security practices are implemented with different teams. From this, the research can be continued with research questions 2 and 3:

- *RQ2: Who are the stakeholders in the large organizations that need visibility to application security and how the organization benefits from it?*

  The purpose of RQ2 was to find out which stakeholders are the ones who benefit from increasing the visibility of application security and who can effectively change things if the visibility is improved.

- *RQ3: Which application security metrics must be visible to different stakeholders?*

  In more concrete terms, this research question can be divided to topics such as which metrics are important to bring to the attention of different stakeholders so that development can be expected to take place, and which metrics are essential in order to give different stakeholders visibility into the current state of the organization's application security so that risks can be reduced, and security be improved.

The survey was sent to total of 35 people in several different positions at the target company working mainly in the Research & Development (R&D) unit or global cybersecurity unit:

- Software Developers / Software Engineers
- Operations Leadership
- Technology Executives
- DevOps Leadership
- DevOps Engineers
- Software Architects
- Cloud Architects
- Security Leadership
- Product Owners

Survey included the following questions:

- What is your role or current job function? You can just add the position you are in or more detailed explanation of key responsibilities in your role.

- Which factors should be considered effective to improve application security visibility in a large organization?

- How the different stakeholders benefit from application security visibility?

- Who are the key stakeholders in the large organizations that need visibility to application security and why?

- Which application security metrics should be visible to different stakeholders? In this context security metrics can be measurable things like security tools in use, number of critical or high severity findings, % of systems with tested security controls, number of systems with known vulnerabilities, policy violations, etc.

- Are you familiar with the CVMS (Centralized Vulnerability Management System) platform?

- We have adopted the CVMS (Centralized Vulnerability Management System) to our use at target organization and have continuously added our digital products in the platform. Do you feel that the visibility into vulnerabilities provided by the CVMS platform helps to improve the security posture of your team/solution?

- Are you familiar with IriusRisk (automated threat modeling and security requirements management) platform?

- To provide more visibility in the status of compliance and security requirements we have started to add more solutions to IriusRisk platform. Do you feel that using IriusRisk has helped you or your team to gain better understanding of security posture of your solution and required security activities?

- Are you familiar with the monthly security dashboard provided by application security team?

- To provide more visibility to specific solution's/product's security posture, we have started to gather security data from R&D solutions to monthly security dashboard. In your opinion what kind of data would be the most important to get in the monthly security dashboard from organizational point of view?

- What type of data would you like to see in monthly security dashboard from point of your personal interest?

- Does monthly security dashboard help you understand security-related problems better?

- Does CVMS dashboard help you understand security-related problems better?

- Are you familiar with the services and tools provided by cybersecurity team?

6 DISCUSSION

The purpose of this chapter is to go through the results for the research questions, to try to solve at least partially the research problem and to discuss the topic in a general way. The detailed answers to the survey are not included in this work as such, because some of the answers contain information that is only intended for internal use of the organization, but the findings and conclusions from the answers are reviewed in a way that they are generic enough and do not violate the organization's information sharing policy.

6.1 Which factors should be considered effective to improve application security visibility in a large organization?

The main purpose of the study was to understand which factors are important to consider, when aiming to improve application security visibility. The answers from the questionnaire varied a lot depending on the roles of the respondents. More than a third highlighted the importance of processes and the fact, that visibility is not improved with any specific tool, dashboard, or system, such as CVMS, but by creating good processes and making sure that everyone knows what the expectations are. Processes and current security status must be communicated clearly between different stakeholders. Being able to present exactly what the security status is, what the potential impact of different threats are, and what exactly needs to be done to improve the situation seems to be most important factor. How the security posture is presented, and which technologies or tools are used to achieve visibility, does not seem to matter that much.

If the development teams are not given a sufficiently accurate picture of the security problems of their own applications, but instead driving them to take more responsibility for the investigations of vulnerabilities and their remedial measures, the tools must be easy enough to use and they must give clear instructions on what needs to be done to implement the security fix. 70% of respondents feel that the CVMS platform improves visibility and makes it easier to understand the security status of applications, but at the same time they feel that the view it offers is not accurate enough. A few problems that emerged in the answers were the poor prioritization of findings, their duplication and insufficient guidance on how to fix the issues. It was pointed out that the system still has a lot of potential, if it is developed into a more mature solution in the future. For software development teams, visibility must be created for the tools and processes that developers can use when building secure software by default. These must be very close to the developers' workflow, so that they are truly adopted and become part of the work culture. When providing visibility to application security, it is not enough to create a view of vulnerability list, and how many security issues each team has in their product. The security team must demand more from itself and focus on building security paved roads, which are designed secure-by-default. The security team cannot assume that the development teams will suddenly become security professionals and use a large part of their working time to understand findings and alerts from different sources. The security team must set clear expectations, guardrails, and step-by-step instructions on how to reach the most important goals and desired levels.

6.2 Who are the stakeholders in the large organizations that need visibility to application security and how the organization benefits from it?

Understanding stakeholder roles that need visibility is very relevant because at the end of the day it is people who make decisions, guidance, and show the direction in which the organization is heading, not tools or technologies. If security visibility is not made available to the right people, it will be very difficult to implement the change. Most of the respondents considered it important to give visibility to product owners or similar persons who have the authority to decide on the prioritization of the tasks in the development teams. Providing the right kind of visibility is especially important so that product owners understand the effect of security status on business impact. Therefore, we can conclude that generic dashboards do not work very well because each team should be able to have visibility to concrete security flaws that are relevant in the context of that application.

On the other hand, the answers also showed that when developers want to be given visibility into the application security, for example through security tooling, the tools should be good in terms of usability and the learning threshold should not be very high. This makes sense because developers already must learn several different tools and technologies and the time to learn and use application security tools is very limited. For this reason, the tools should be able to provide the information that is being sought rather quickly, and the security team should prepare certain secure frameworks that are configured secure-by-default. In this case, the developers' workflow is not broken, and the maximum possible benefit can be obtained from the toolchain. This is something that organizations should keep in mind when choosing new application security tools, because organizations do want to allow development teams to be autonomous with no bottlenecks from application security team.

In summary, visibility must be brought to the leadership so that the necessary resources can be given to manage security tasks. Relevant instructions and clear guardrails are important for developers so that they know what is expected of them.

6.3 Which application security metrics must be visible to different stakeholders?

The answers to the survey vary a lot, and the question of which metrics are the most necessary seems to depend entirely on the respondent's role in the organization. In general, all information was considered interesting and useful but practical metrics usually depend on application, hosting method, infrastructure used, policies, regulation and so on. One clear metric that stood out for several respondents was the criticality of security findings, and they were especially interested in the findings with the greatest possible business impact. Another clear metric that several respondents brought up was the number of days it takes to fix a certain security vulnerability after it is discovered. Inability to fix security vulnerabilities immediately after they appear reflects the organization's security culture and the precision at which the processes are designed. The answers also showed that metrics alone are not always perceived as very important, but one should be able to present what can be done about the security problems behind the metrics, and preferably at an accurate level. It can be assumed that until now the guidelines have been too imprecise or high-level guidelines, so they should be developed to better meet the needs and expectations of development teams.

7 CONCLUSION AND FURTHER RESEARCH

Visibility alone does not improve the application security if the visibility provided is not tied to the context. When improving application security visibility, it is necessary to pay attention to the impact of the security findings provided by the visibility, and how the situation can be enhanced during the entire software development life cycle. It is very important to provide visibility to the various stakeholders in the organization, so that any actions can be taken to improve application security. However, the focus should be on business impact, the most accurate situational awareness, and clear guidelines, that can be used to improve application security.

The research problem was the lack of visibility into the activities in secure software development lifecycle. That included working methods of different development teams, used tools and technologies, the utilization rate of security tools and the security posture of the products. During the research process, it was noticed that methodology and maturity assessments are effective ways to enhance visibility and get a high-level view of the DevSecOps practices of an organization, or specific part of an organization. Methodology and maturity assessment gives an understanding of the state of different sub-areas, and different trends that should be paid attention to. However, it alone does not help to improve and develop application security in a team level, because genuine improvement and development requires interest and attention to details. One thing worth noting in this or similar questionnaires are methods where “you get what you measure”, meaning that by using closed-ended questions and partially self-administered questionnaires, quality of these assessment may vary. In some answers, it was noticeable that the question was not completely understood, and therefore some of the answers can be considered a bit unreliable and some of them should perhaps have been interpreted between the lines. However, this did not apply to a large part of the answers, thus did not affect the results too much.

Almost every one of the respondents were familiar with the different security tools offered by the application security team, although not all of them had used these tools themselves. A conclusion can be drawn from this observation, that the visibility has been improved outside the security team as well, and the capabilities offered by the security team have been communicated successfully. This study helped indeed to improve the visibility of the target company's application security activities, and the security posture of digital products. Some clear shortcomings were noticed though, such as the fact that dynamic testing is not done very widely in R&D teams, or at least it has not been made visible. In modern application security, it is not enough just to implement shift left and move security testing to the beginning of the software development lifecycle. It is also crucial to be able to test production environment and applications in the state where customers use them, because that is exactly the view that potential malicious intruders and hackers get as well.

Visibility improves understanding of security status, tools, processes, and working methods in the organization. In the target company, the building blocks and security foundations are well in place, but the study shows clearly that security is not done in a context-dependent manner. Instead, the same security requirements, principles, and instructions are followed everywhere. This is not necessarily always a bad thing, and with the help of generic and good practice guidelines, we aim to clarify an otherwise very complex operating environment. However, development teams would greatly benefit from clearer security support tied to the context. One such idea is the so-called security paved roads, or guardrails, where the aim is to implement a secure-by-default framework for application development teams. The framework is a best practices model, that is always tied to the product and used technologies. It is intended to improve security visibility and reduce the need for the security team to act as a roadblock, but instead as an enabler. The research was carried out using the action research method, and Jean McNiff's model was used as the theoretical basis. The research project was iterative in nature, and because an open-ended questionnaire was partly used for data gathering, the analysis was sometimes challenging. However, it can be stated that the research work was successful, and answers to the research questions were found.

The topic has a lot of potential for further research. In particular, the correlation between the security findings provided by increased visibility and fixing the issues would be a very important area to study, which in this work received little attention because in large organizations, the development of cultural and working practices takes its own time. It remains to be seen how the development that took place during the study, affects the security posture of teams and products in the long term. Research on security guardrails and secure-by-default frameworks would be a logical continuation of this work because development teams clearly need more support and more technically precise instructions so that digital products can be developed securely in the future. There is a gray zone between where vulnerabilities are discovered and remedial actions are implemented, and in this zone, development teams and security teams must meet each other. Application security as an industry needs to raise its level by one step and offer developers a secure paved road to walk on.

REFERENCES

Amazon Web Services, Inc. 2022a. What is continuous delivery? WWW document. Available at: https://aws.amazon.com/devops/continuous-delivery/

Amazon Web Services, Inc. 2022b. What is continuous integration? WWW document. Available at: https://aws.amazon.com/devops/continuous-integration/

Brotby, W. & Hinson, G. 2016. Pragmatic Security Metrics. Applying Metametrics to Information Security. Boca Raton: Auerbach Publications.

Calder, A. 2020. The Cyber Security Handbook. Prepare for, respond to and recover from cyber attacks with the IT Governance Cyber Resilience Framework (CRF). Cambridgeshire: IT Governance Publishing.

Chatterjee, S. 2021. Designing API-First Enterprise Architectures on Azure. Birmingham: Packt Publishing.

Conklin, A. & Shoemaker, D. 2022. CSSLP. Certified Secure Software Lifecycle Professional. New York City: McGraw-Hill.

Dang, W. & Kohgadai, A. 2021. DevSecOps in Kubernetes. California: O'Reilly Media In.

Detectify. 2022. External Attack Surface Management (EASM). What it is and what it isn't. Available at: https://detectify.com/resources/ebooks-whitepapers/external-attack-surface-management-what-it-is-and-isnt

DevSecOps: A Multivocal Literature Review. 2017. Conference paper. Available at: https://www.researchgate.net/publication/319633880_DevSecOps_A_Multivocal_Literature_Review [Accessed 4 February 2022]

DZone. 2022. How to bring the power of security guardrails to your application security program. WWW document. Available at: https://dzone.com/articles/how-to-bring-the-power-of-security-guardrails-to-y

Farley, D. 2021. Modern Software Engineering: Doing what works to build better software faster. Boston: Addison-Wesley Professional.

Flaus, J. 2019. Cybersecurity of Industrial Systems. New Jersey: Wiley-ISTE.

Gayathri, M. 2022. Full Stack Testing. A Practical Guide for Delivering High Quality Software. O'Reilly Media, Inc.

GitLab. 2020. GitLab's DevSecOps Methodology Assessment.

Hsu, T. 2018. Hands-On Security in DevOps. Ensure continuous security, deployment and delivery with DevSecOps. Birmingham: Packt Publishing.

Hsu, T. 2019. Practical Security Automation and Testing. Tools and techniques for automated security scanning and testing in DevSecOps. Birmingham: Packt Publishing.

Kim, G. 2016. The DevOps Handbook. How to create world-class agility, reliability & security in technology organizations. Portland: IT Revolution Press.

Lapan, S. 2011. Qualitative Research. An introduction to methods and designs. San Francisco: Jossey-Bass.

Magnusson, A. 2020. Practical Vulnerability Management. A strategic approach to managing cyber risk. San Francisco: No Starch Press.

McNiff, J. 2002. Action Research: Principles and Practice. London: Routledge.

Mohamed, S. 2015. DevOps Shifting Software Engineering Strategy Value Based Perspective. IOSR J. Comput.

National Institute of Standards and Technology, 2021. Framework for DevSecOps. National Institute of Standards and Technology, Gaithersburg, Maryland, United States.

Oka, D. 2021. Building Secure Cars. Assuring the automotive software development lifecycle. New Jersey: Wiley.

Podjarny, G. Cloud Native Application Security. Embracing Developer-First Security for the Cloud Era. 2021. California: O'Reilly Media, Inc.

Puppet. The 2021 State of DevOps. 2021. Report. Available at: https://puppet.com/resources/?refinementList%5Btype%5D%5B0%5D=Report&page=1&configure%5BhitsPerPage%5D=18 [Accessed 2 February 2022]

Ribeiro, M. 2022. Learning DevSecOps. Integrating continuous security across your organization. California: O'Reilly Media, In.

Schein, P. & Schein, E. 2016. Organizational Culture and Leadership, 5th Edition. New Jersey: Wiley.

Sonatype. DevSecOps Community Survey. 2020. Report. Available at: https://www.sonatype.com/hubfs/DevSecOps%20Survey/2020/DSO_Community_Survey_2020_Final_4.1.20.pdf [Accessed 1 February 2022]

Thomas, T., Tabassum, M., Chu, B. & Lipford, H. 2018. Security During Application Development: An Application Security Expert Perspective. https://dl.acm.org/doi/pdf/10.1145/3173574.3173836

Why visibility is critical to your security management program. Check Point. WWW document. Available at: https://blog.checkpoint.com/2016/03/07/why-visibility-is-critical-to-your-security-management-program/

Zeeshan, A. 2020. DevSecOps for .NET Core: Securing modern software applications. Ebook. Apress. Available at: https://learning.oreilly.com/library/view/devsecops-for-net/9781484258507/

LIST OF FIGURES