FLAWFIX
BY L3MONTREE

ISO 27001 COMPLIANT
VULNERABILITY MANAGEMENT

FLAWFIX

DevSecOps Made Effortless:
Security at the Speed of Code!

Table of Conents

1 Introduction **2**

2 Requirements for vulnerability management **3**
2.1 Requirements through regulations and best practices 3
2.2 Requirements through DevSecOps practices 11

3 FlawFix **13**
3.1 Objective 13
3.2 Target group 13
3.3 Functionalities 13
3.4 Architecture and technical specification 17

4 Unique Selling Points (USPs) **18**
4.1 European 18
4.2 Open source development 18
4.3 Usability 18
4.4 Cost efficiency 18
4.5 Focus on DevSecOps 19
4.6 Focus on data protection and security 19

5 Differentiation from existing systems **20**
5.1 DefectDojo 20
5.2 ArcherySec 20

6 References **22**

1 Introduction

In today's world, where cyber threats and the number of known vulnerabilities in IT systems are constantly increasing [4], vulnerability management is becoming a central pillar of IT security. It is also reflected in the requirements of internationally recognized standards and security frameworks. In those, the management of software vulnerabilities (risks) is an explicit or implicit requirement [5, 6, 7, 12].

Manual vulnerability management is a time-consuming and error-prone process. Given the complexity and scope of today's IT systems, it is almost impossible to identify and fix all vulnerabilities effectively and in a timely manner. The ineffective management of vulnerabilities through manual procedures, without adequate support, evaluation, and summarization using inappropriate tools such as spreadsheets, also increases the likelihood of no management at all.

At the same time, the landscape of software development has changed fundamentally. The introduction of DevOps practices has led to faster and more agile software development and deployment. In this context, there is a need to consider security as an integral part of the development process, which has led to the emergence of DevSecOps. DevSecOps integrates security aspects directly into the continuous development, integration and deployment process [11].

To adhere to security regulations in this fast-paced and ever-changing environment, organisations must implement an effective vulnerability management system that also meets the requirements of DevOps and DevSecOps. This requires a strategy that not only enables the identification and remediation of vulnerabilities in an agile development environment, but also includes continuous monitoring and adaptation to new threats. Effective vulnerability management is therefore essential to ensure security in the modern IT landscape while supporting the agility and speed of the DevOps methodology.

FlawFix aims to address these challenges through modern, high-performance, open-source and integrated vulnerability management.

2 Requirements for vulnerability management

The effective classification and handling of vulnerabilities in software is a key challenge in the field of cybersecurity. Organisations are faced with the problem of adequately assessing and managing risks arising from software vulnerabilities. Classifying risks into categories such as critical, high, medium, or low is essential to quickly identify, prioritise and address the vulnerabilities with the highest risk. The aim is to reduce the likelihood of critical vulnerabilities in your system being exploited [12]. In addition, security standards place a wide range of requirements on the process of risk assessment and risk mitigation. The section 2.1 lists the requirements that the ISO 27001 and PCI DSS standards place on the risk and vulnerability management process and compares them with the OWASP Vulnerability Management Guide (OVMG) [10]. This document is not a comprehensive compilation of standards and guidelines, but rather a brief comparison of the relevant information pertaining to vulnerability management. The aim of this section is to collect and group all requirements and recommendations for the vulnerability management process, and then derive the necessary functionalities of FlawFix.

2.1 Requirements through regulations and best practices

Various security frameworks and best practices guides often have overlapping requirements with regard to the vulnerability management process. The following sections group together the regulatory requirements of ISO 27001 and PCI DSS, as well as the recommendations of the OWASP Vulnerability Management Guide (OVMG) [10].

The following sub-areas of the vulnerability management process are analysed:

- 2.1.1 Requirements for the asset management process
- 2.1.2 Requirements for the process of identifying risks / vulnerabilities
- 2.1.3 Requirements for the risk assessment process
- 2.1.4 Requirements for the risk mitigation process
- 2.1.5 Requirements for the reporting process

2.1.1 Requirements for the asset management process

ISO 27001 does not place any direct requirements on the asset management process in relation to the handling of software vulnerabilities. For this reason, the more specific requirements of the PCI DSS and the OVMG are considered here. However, the following requirements for asset inventory are also considered in the software

- Name of the asset
- Owner of the asset
- Category of the asset
- Location of the asset
- Relevant information

ID	PCI DSS	OVMG
A1	An inventory of bespoke and customised software and third-party software components integrated into bespoke and customised software is maintained to facilitate vulnerability and patch management. (6.3.2)	
A2		Know technical constraints (1.1.3)
A3		Distinguish primary assets vs. secondary (1.1.4)

A4		Determine functional asset groups (2.1.1)
A5		Determine asset groups by type of environment (2.1.2)
A6		Determine asset groups by type of type of system (2.1.3)

2.1.2 Requirements for the process of identifying risks / vulnerabilities

ID	PCI DSS	OVMG
A7	Vulnerabilities for customised and customer-specific third-party software (e.g. operating systems and databases) are covered. (6.3.1)	Scan/test web applications (1.3.3), Scan/test mobile apps (1.3.4)
A8	Scan publicly available web applications using manual or automated application vulnerability assessment tools or methods as follows: ■ At least once every 12 months and after significant changes. ■ All vulnerabilities are corrected. (6.4.1 shortened)	Determine the frequency of your security tests (1.2.2), Scan/test web applications (1.3.3), Scan/test mobile apps (1.3.4)
A9	Internal vulnerability scans are performed at least once every three months. (11.3.1 shortened)	Determine the frequency of your security tests (1.2.2)
A10	Scans are carried out by qualified personnel and there is organisational independence of the tester. (11.3.1 shortened) *Internal vulnerability scans may be performed by qualified internal staff who are reasonably independent of the system component(s) being scanned (for example, a network administrator should not be responsible for scanning the network), or an entity may choose to have internal vulnerability scans performed by a firm specialising in vulnerability scanning.*	
A11	The scan tool is kept up to date with the latest vulnerability information (11.3.1)	Ensure the latest vulnerability feed (1.2.3)
A12	Internal vulnerability scans are performed via authenticated scanning as follows: ■ Systems that cannot accept credentials for authenticated scanning are documented. ■ Sufficient privileges are used for systems that accept credentials for scanning. ■ If accounts used for authenticated scanning can be used for interactive logon, then these are managed according to requirement 8.2.2. (11.3.1.2)	Scan private subnets (1.3.2)

A13	External vulnerability scans are performed as follows: ■ At least once every three months. ■ From a PCI SSC-approved scanning provider (ASV). ■ Vulnerabilities are fixed and the requirements of the ASV programme manual for a passed scan are met. ■ Rescans are performed as required to confirm that vulnerabilities have been remediated in accordance with the requirements of the ASV programme manual for a passed scan. (11.3.2)	Determine the frequency of your security tests (1.2.2), Scan public IP addresses (1.3.1)
A14		Check if your test results have valuable data (1.4.1)
A15		Interpret and reconcile system/device fingerprinting across your tests (1.4.2)
A16		Determine that running services are what they are supposed to be (1.4.3)
A17		Find something that falls out of the pattern and investigate why (1.4.4)
A18		Randomly select vulnerabilities and confirm them with a different tool or manually (1.4.5)

2.1.3 Requirements for the risk assessment process

ID	ISO 27001 *The organization shall define and apply an information security risk assessment process that:*	PCI DSS	OVMG
A19	establishes and maintains information security risk criteria that include the risk acceptance criteria (6.1.2 a. 1.)		Check if vulnerability exceptions exist (1.2.4), Establish ground rules for vulnerability exceptions (3.4.2)
A20	establishes and maintains information security risk criteria that criteria for performing information security risk assessments (6.1.2 a. 2.)		Using CVSS, apply unique environmental traits to your vulnerability analysis (2.4.3), Use your reports (3.1.1), Use trend analysis (3.1.2), Use information from additional sources (3.1.3), Apply other environmental factors (3.1.4)
A21	ensures that repeated information security risk assessments produce consistent, valid and comparable results (6.1.2 b.)		End Goal: create a data-based argument for vulnerability prioritization. (3.1)
A22	apply the information security risk assessment process to identify risks associated with the loss of confidentiality, integrity and availability for information within the scope of the information security management system; (6.1.2 c. 1.)		
A23	identify the risk owners (6.1.2 c. 2.)		Communicate to responsible and accountable stakeholders (3.1.5)
A24	assess the potential consequences that would result if the risks identified in 6.1.2 c) 1) were to materialize (6.1.2 d. 1.)		
A25	assess the realistic likelihood of the occurrence of the risks identified in 6.1.2 c) 1) & (6.1.2 d. 2.)	New security vulnerabilities are identified using industry-recognised sources of security vulnerability information, including alerts from international and national Computer Emergency Response Teams (CERTs). (6.3.1)	Use information from additional sources (3.1.3)

A26	determine the levels of risk (6.1.2 d. 3.)	At a minimum, risk ratings identify all vulnerabilities that are considered high-risk or critical to the environment. Vulnerabilities are assigned a risk rating based on industry best practice and consideration of potential impact. (6.3.1)	
A27	compare the results of risk analysis with the risk criteria established in 6.1.2 a) (6.1.2 e. 1.)		
A28	prioritize the analysed risks for risk treatment (6.1.2 e. 2.)		Use your reports (3.1.1), Use trend analysis (3.1.2)
A29	The organization shall retain documented information about the information security risk assessment process (6.1.2 abschließender Satz)		Construct a repeatable business process (3.3.2), Document all FP submissions (3.3.3), Document each FP and store it in an auditable repository (3.3.6) Create an appropriate policy (3.3.7), Document each exception and store it in the company's audit system (3.4.5), Create an appropriate policy (3.4.6)
A30			Apply other environmental factors *Your organization has daily, weekly, monthly, and quarterly priorities. Based on the function of each team, these priorities may be dominant or secondary. Think about how vulnerability management may feed into other teams' goals.* (3.1.4)
A31			Find a SMEs who can agree or argue a false positive claim (3.3.4)
A32			Set a time frame at which FP should be reevaluated (3.3.5)
A33			Establish periodic reviews of vulnerability exceptions (3.4.3)
A34			Have vulnerability exception solicitors asking the executive authority for an approval every time (3.4.8)

2.1.4 Requirements for the risk mitigation process

ID	ISO 27001 *The organization shall retain documented information about the information security risk assessment process:*	PCI DSS	OVMG
A35	select appropriate information security risk treatment options, taking account of the risk assessment results (6.1.3 a.)	Vulnerabilities rated 4.0 or higher by the CVSS are fixed. (11.3.2.1)	Find the stakeholders responsible for remediation work (3.2.1)
A36	determine all controls that are necessary to implement the information security risk treatment option(s) chosen (6.1.3 b.)		
A37	compare the controls determined in 6.1.3 b) above with those in Annex A and verify that no necessary controls have been omitted (6.1.3 c.)		
A38	produce a Statement of Applicability that contains: ■ the necessary controls (see 6.1.3 b) and c)); ■ justification for their inclusion; ■ whether the necessary controls are implemented or not; and ■ the justification for excluding any of the Annex A controls. (6.1.3 d.)		
A39	formulate an information security risk treatment plan (6.1.3 e.)	High-risk and critical vulnerabilities (according to the entity's vulnerability risk ratings defined in requirement 6.3.1) are remediated. Rescans are performed to confirm that all high-risk and critical vulnerabilities (as mentioned above) have been remediated. (11.3.1)	Find the stakeholders responsible for remediation work (3.2.1), Communicate your findings via the tools and processes they use (3.2.2), Always assign remediation work (3.2.7)
A40	obtain risk owners' approval of the information security risk treatment plan and acceptance of the residual information security risks. (6.1.3 f.)		Include responsible, accountable stakeholders, and those who need to be informed on unresolved issues (3.2.8)

A41			Use the ticketing system or change management system to resolve remediation management issues (3.2.6)

2.1.5 Requirements for the reporting process

ID	ISO 27001 *The management review shall include consideration of:*	OVMG
A42	Top management shall review the organization's information security management system at planned intervals to ensure its continuing suitability, adequacy and effectiveness. (9.3.1)	Maintain a consistent frequency of reporting and use it to track changes (2.4.1), Submit both versions of the report to your manager/CISO (2.4.9), Report your test results to the responsible stakeholders (3.2.5)
A43	the status of actions from previous management reviews (9.3.2 a.)	
A44	changes in external and internal issues that are relevant to the information security management system (9.3.2 b.)	
A45	changes in needs and expectations of interested parties that are relevant to the information security management system (9.3.2. c.)	

A46	feedback on the information security performance, including trends in: 1. nonconformities and corrective actions 2. monitoring and measurement results 3. audit results 4. fulfilment of information security objectives (9.3.2 d)	*Define/Refine Metrics* Determine groups by CVE numbering authority or underlying technology (2.1.4), *Define/Refine Metrics* Determine groups by type of vulnerability (2.1.5), *Define/Refine Metrics* Determine the amount and percentage of vulnerable assets (2.2.1), *Define/Refine Metrics* Determine the amount and percentage of vulnerable assets by severity and CVSS (2.2.2), *Define/Refine Metrics* Determine the amount and percentage of new vulnerabilities: ■ by severity (2.2.3.1) ■ by functional groups (2.2.3.2) ■ by type of environment (2.2.3.3) ■ by type of system (2.2.3.4) ■ by CVE numbering authority (2.2.3.5) ■ by type of vulnerability (2.2.3.6) (2.2.3), *Define/Refine Metrics* Compare and analyze aging data by the severity of vulnerabilities and their share ■ enterprise-wide (2.2.4.1) ■ among all other vulnerable assets (2.2.4.2) ■ by functional groups (2.2.4.3) ■ by type of environment (2.2.4.4) ■ by type of system (2.2.4.5) ■ by CVE numbering authority (2.2.4.6) ■ by type of vulnerability (2.2.4.7) (2.2.4), *Define/Refine Metrics* Draw out trends by count and percentage utilizing KPI that matter to your enterprise risks and compliance (2.2.5), *Define/Refine Metrics* Determine exploitability of vulnerable assets by severity; specify count, percentage, decrease or increase
A47	feedback from interested parties (9.3.2 e.)	
A48	results of risk assessment and status of risk treatment plan (9.3.2 f.)	
A49	opportunities for continual improvement. (9.3.2 g.)	In one paragraph, add your recommendations (2.4.6)
A50		Aggregate and process collected data (2.4.2)
A51		State vulnerability trends (2.4.4)
A52		Hypothesize about these trends in one sentence (2.4.5)
A53		Apply data sensitivity classification to your report (2.4.7)
A54		Make a shorter version (1-2 pages) of your report (2.4.8)
A55		Create and maintain your own vulnerability management repository for internal or external audit (2.4.10)

A56		Be able to explain the details of vulnerability detection and the reporting process (2.4.11)

2.2 Requirements through DevSecOps practices

Behind the DevSecOps concept is the goal of recognising security problems as quickly as possible [11]. DevSecOps integrates important aspects of IT security into the CAMS principles of the DevOps concept. The following principles are fundamental to the implementation of a DevSecOps process [9]:

1. **C**ulture: Close collaboration between development, operations and security teams.
2. **A**utomation: Automation of build, deployment, testing and security checks.
3. **M**easurement: Monitoring of operational metrics such as business revenue and key performance indicators. In addition, use of metrics to track threats and vulnerabilities.
4. **S**haring: Sharing knowledge and tools between developers, operators and security teams.

In particular, the *Automation* and *Measurement* part of the DevSecOps principles is usually implemented through the use of so-called DevOps pipelines. A DevOps pipeline describes a series of phases or steps that are carried out each time a change is made to software. These steps are, for example, the execution of unit and integration tests as well as the execution of a build process.

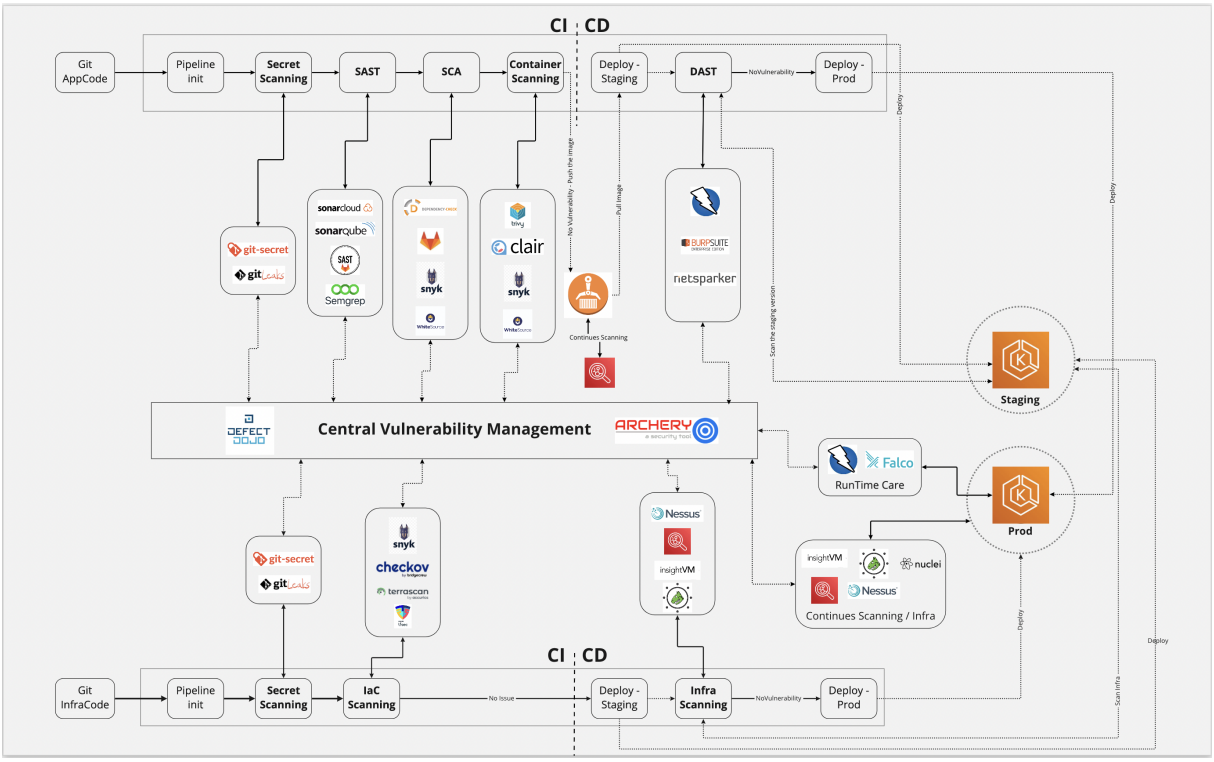

Figure 2.1: Sicherheitsschritte einer DevSecOps-Pipeline [11]

Figure 2.1 shows an example of the elements that need to be added to a DevOps pipeline to identify security problems as quickly as possible. Specifically, these are

1. **Secret scanning**: The source code of the application is scanned for secrets such as API keys and passwords
2. **SAST** (Static Application Security Testing): Static code analysis checks the application's source code for syntax errors, security vulnerabilities, programming errors and disregard for coding standards

3. **SCA** (Software Component/Composition Analysis): Software Component Analysis checks third-party and open-source components of the application for known vulnerabilities
4. **container scanning**: Containers have established themselves as the industry standard in modern software development. In the container scanning phase, these are scanned for known vulnerabilities.
5. **DAST** (Dynamic Application Security Testing): DAST is a “black box” test that uncovers security vulnerabilities in running applications by injecting malicious payloads to identify potential vulnerabilities for attacks such as SQL injections or cross-site scripting (XSS). DAST only takes place after the application has been deployed to a staging infrastructure.
6. **Secret-Scanning (InfraCode)**: Modern environments are defined using IaC (Infrastructure as Code). IaC is a DevOps practice where infrastructures (such as networks, virtual machines, load balancers, etc.) are configured and managed using code instead of using manual processes. This step has the task of finding secrets in plain text in IaC files.
7. **IaC Scanning**: IaC scans are automated scans of IaC files to identify security holes, misconfigurations and other vulnerabilities.
8. **InfraScanning**: Infrastructure scans are scans that aim to identify vulnerabilities in IT infrastructure such as servers, networks, and system configurations.

Figure 2.1 clearly shows that, compared to the traditional waterfall model for software development, an application is checked much more frequently and thoroughly for security vulnerabilities and the associated risks due to shorter release cycles. This poses the following new challenges for the company’s centralised vulnerability management system:

Automation: Because DevOps is based on rapid, continuous development and deployment cycles, vulnerability management must be automated to keep up with this pace. Automated tools and processes are required to quickly identify and fix vulnerabilities.
Integration into CI/CD pipelines: Vulnerability management needs to be integrated into Continuous Integration/Continuous Deployment (CI/CD) pipelines. This means that security scans and vulnerability assessments must take place as part of the regular development and deployment of software.
Continuous assessment: As both the threat landscape and the software itself are continuously changing, vulnerability management needs to take and assess continuous vulnerability reports.
Easy to get started: DevSecOps requires a cultural shift where security is seen as a shared responsibility of all team members, not just the security team. This requires the vulnerability management process to be easily accessible, especially to people without prior IT security knowledge.
Feedback loops: Effective feedback loops are required to integrate the results of security analyses back into the development process to enable continuous improvement.
Adaptability Vulnerability management in a DevSecOps environment must be flexible and adaptable to respond quickly to new security threats and trends.

3 FlawFix

3.1 Objective

The FlawFix software pursues the following core objectives:

1. The effort required by a company to implement security standards, such as DIN ISO/IEC 27001, is to be reduced by removing the required documentary tasks.
2. The use of FlawFix should not only simplify a company's vulnerability management by providing a standardised interface, a central collection of all vulnerabilities and security gaps with all the necessary information, but should also make it easier to manage cybersecurity.
3. The cybersecurity of a company should be greatly improved through the use of FlawFix, as the software utilises established standards for the implementation of vulnerability management.
4. The implementation and management of DevSecOps practices should be significantly simplified, **especially** for people without IT security experience (focus on UI/UX).
5. The partial automation of the above-mentioned processes should free up the capacities of specialists, who can then be deployed more effectively by the company.

Overall, FlawFix provides a standardised, well-documented vulnerability management process urged by security standards.

3.2 Target group

The target audience of FlawFix covers a wide range of companies and professionals, but special attention is paid to organisations that use or want to introduce DevSecOps practices. The target group can therefore be defined as follows

1. **Developers and operations teams**: FlawFix offers valuable support, especially in companies that use or want to introduce DevSecOps practices. It helps developers and operations teams without IT security experience to seamlessly integrate security aspects into their workflows.
2. **IT and security teams**: These teams, whether they work in a DevSecOps environment or not, will find FlawFix a useful tool to efficiently implement and manage security standards.
3. **Executives and managers**: For decision makers who want a holistic view of their organisation's security landscape, FlawFix offers a comprehensive solution. It supports them in gaining an overview of the current IT security status of the organisation, both in DevSecOps-oriented environments and in more traditional structures.
4. **Compliance officers**: FlawFix also serves the needs of compliance officers and auditors who have to implement security standards such as DIN ISO/IEC 27001 in various organisational contexts.

3.3 Functionalities

Section 2 presents the requirements for vulnerability management that arise due to regulations, best practices and agile working methods.

This section explains how these requirements are mapped and implemented as functionalities in the software. The requirements are structured in milestones:

1. **M1:** All functions absolutely necessary for the success of the product
2. **M2:** Necessary functions are enriched with additional functions that further simplify the use of FlawFix
3. **M3:** Backlog: Planned functions that may be integrated in the future.

Based on the diagram 2.1, the term asset is defined as follows:

- An asset is an application that is developed in-house. Scan reports are fed into FlawFix via the Git AppCode process shown in 2.1.

- An asset is an infrastructure (consisting of numerous host systems and the applications running on them). The scan reports are fed into FlawFix via the Git InfraCode.

3.3.1 A1: Inventory of customised and customer-specific software and software components (M1, M2)

FlawFix focuses in particular on companies that develop software themselves and pursue DevSecOps practices. Such companies fulfil A1 utilizing SCA (Software Composition Analysis) checks. FlawFix should already be able to handle SCA scans in M1.

In later versions, companies that purchase software will be able to upload an SBOM (Software Bill of Material, CyclonDX) to FlawFix. Based on the information it contains, vulnerability management can also be carried out without access to the source code (M2).

3.3.2 A2-A6: Grouping and adding further information to assets (M1)

Each asset stored in FlawFix can be saved or tagged with the following information:

1. Is the application or infrastructure accessible from the Internet?
2. General importance of the asset (A3)
3. Confidentiality requirements (Undefined, Low, Medium, High)
4. Integrity requirements (Undefined, Low, Medium, High)
5. Availability requirements (Undefined, Low, Medium, High)
6. Any other tags for possible grouping and more detailed reports.

3.3.3 A7-A18: Identification of risks and vulnerabilities (M1, M2, M3)

M1 initially focuses on the following steps of the DevSecOps cycle: SAST, SCA, container scanning. These steps are limited to the detection of CVEs. However, FlawFix will already offer an import interface for SARIF reports in M1. This means that the results of other scanning processes can already be managed in M1, although these will not initially be enriched with further data (M1).

A separate command line tool will be developed in the future (M2) for the identification of risks: FlawFind. This command line tool consolidates the open-source scanner projects for the individual sub-areas. It is therefore a curated list of scanners, the results of which are communicated directly to FlawFix. Core objective 4 in particular is pursued here:

Secret Scanning Secret scanning is performed using Git leaks (M2).

SAST Semgrep is used for static code analysis. (M1)

SCA Software composition analysis is partly possible through the package manager itself (npm audit). The integration of the OWASP dep-scan project is useful here. This project also performs a risk analysis for open-source packages, as well as a reachability analysis to determine whether the dependency is used in a way that is actually exploitable. (M1)

Container scanning To scan containers, either the open-source software Trivy is integrated into FlawFind or OWASP DepScan is used. An evaluation must first take place here (M1)

DAST The open-source software OWASP-ZAP is integrated into FlawFind for dynamic application security testing. (M2)

IaC The open-source software `checkov` is integrated into FlawFind for scanning infrastructure as code files (M2).

InfraScanning The correct software for InfraScanning has not yet been determined.

The tools for scanning the assets will be presented to the open-source community and possibly adapted. However, the software used must be open source to ensure the necessary adaptability.

Integration with the Common Security advisories framework (CSAF) is planned in milestone 3. The CSAF is a standard for the publication of information about security vulnerabilities in software. It was developed by the organisation OASIS (Organization for the Advancement of Structured Information

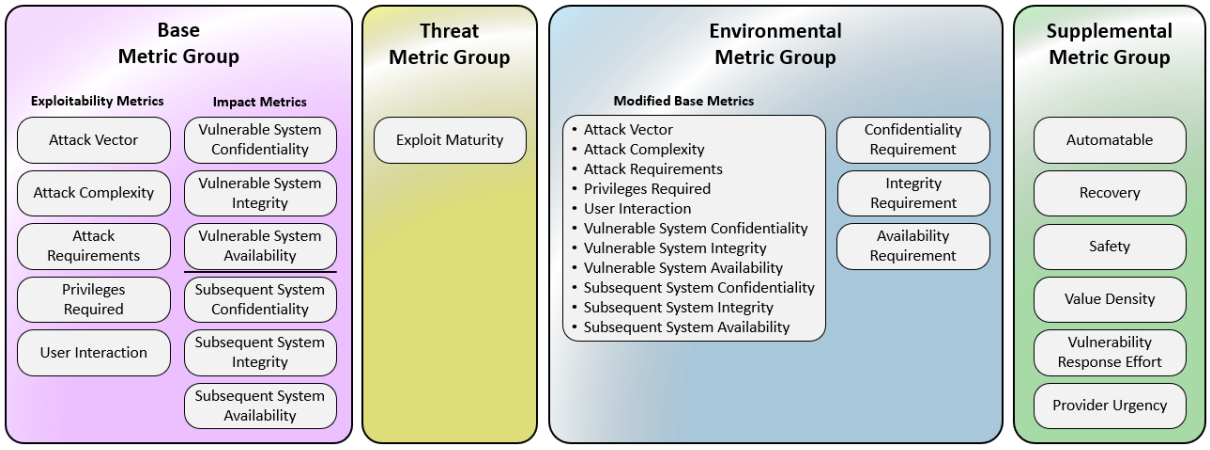

Figure 3.1: CVSS metric groups [1]

Standards). The aim of the CSAF is to provide a standardised and automatable way to disseminate information about security vulnerabilities and how to fix them.

3.3.4 A19: Risk acceptance criteria (M2)

An organisation using FlawFix is offered the opportunity to define its own risk acceptance criteria. This is done using the automatically calculated risk (CVSS 4.0). If the CVSS 4.0 is lower than a previously defined threshold value, the risk is automatically accepted. Each automatically accepted risk is specially tagged to enable subsequent allocation.

3.3.5 A20-A22, A24-A30: Risk assessment (M1)

FlawFix uses the CVSS 4.0 [1] for the risk assessment. The CVSS is divided into four metrics groups (see figure 3.1): Base, Threat, Environmental and Supplemental.

The base metrics group comprises basic, unchanging properties of a vulnerability that remain constant over time and across different user environments. It is composed of:

Exploitability metrics These metrics assess how easily and with what technical means a vulnerability can be exploited. They relate to characteristics of the affected system, which is referred to as the “vulnerable system”.

Impact metrics These metrics assess the direct consequences of a successful attack. They represent the impact on the affected system and/or the downstream impact on other systems, which are referred to as “downstream systems”.

Each detected CVE is automatically enriched with the values of the base metric group from the NIST National Vulnerability Database and displayed in FlawFix.

The threat metric of a CVE is updated daily using the CISA Known Exploited Vulnerabilities Catalogue (https://www.cisa.gov/known-exploited-vulnerabilities-catalog). This automatic procedure is recommended by the FIRST organisation and by CISA itself [1, 2]. In addition, the ExploitDB project provides valuable information about exploits. The possible values of the threat metric group are Attacked, Proof-of-Concept, Unreported, Not Defined.

The environmental metric group, on the other hand, must be created by the user. When a new asset is created, the security requirements are queried (Confidential Requirement, Integrity Requirement, Availability Requirement). The user is provided with explanations and examples to make this process as simple as possible. The possible values are: Not defined, Low, Medium, High.

To summarise, the risk assessment in FlawFix can be described as follows: Every day, a current CVSS 4.0 value is calculated, which represents the risk. However, the user can view and change the calculation at any time. The user must write a justification for any change to the assessment so that the process fulfils A21.

The CVSS 4.0-based risk assessment presented here is a process that fulfils requirements A21 and A22. The use of CVSS 4.0 creates repeatability of the assessment. It is also purely data-based, but can be overwritten by manual changes if necessary. As mentioned in 3.3.5, a justification is required for this.

3.3.6 A23, A35-A41: Risk treatment and identification of risk owners (M1, M2)

A risk owner must be identified for risk treatment. The risk treatment options follow the NIST recommended scheme [8]:

1. Accept: No identification of a risk owner, but a justification of the actor is necessary.
2. Mitigate: Risk handling takes place via integration with a ticket system. A new ticket is created with all the necessary information and a person responsible must be assigned to process this ticket (M1).
3. Transfer: An email with the ticket information is sent to a responsible person. FlawFix keeps track of who the recipient of this email is (M1). If there is a reply to the email, a comment is automatically created in FlawFix so that seamless communication across different media is possible (M2).
4. Avoid: Justification is required for the Avoid mitigation type.

3.3.7 A31-A34: Handling of false positives (M1, M2)

When creating an asset, at least one technically responsible person must be specified. Only this person is permitted to flag a vulnerability as a false positive by writing a justification.

Each company can define the intervals at which false positives require a new justification. FlawFix recommends revalidation every 6 months.

3.3.8 A42-A56 (M1): Reports and dashboards

Every month, a report containing the following information is sent by email to the management and other responsible persons as required

- Total risk score across all assets
- History of the risk score to show changes
- Percentage share of vulnerable assets
- Percentage of vulnerable assets by CVSS and severity
- Amount of closed vulnerabilities
- Amount of new vulnerabilities:
  - By severity level
  - By functional groups
  - By type of environment
  - By type of system
  - By type of vulnerability
- Comparison and analysis of age data by severity of vulnerabilities and their proportion:
  - Company-wide
  - Among all other vulnerable assets
  - According to functional groups
  - By type of environment
  - By type of system
  - By CVE numbering authority

- By type of vulnerability

You can also add your recommendation to these reports. This can be entered in the FlawFix user interface. The user is shown transparently when this message is sent to the management in a report. Each reply to the report e-mail leads to a comment in FlawFix (M2).

In addition to the monthly reports, FlawFix also displays the aforementioned metrics in the user interface in the form of dashboards. The individual dashboards should also be exportable in PDF format.

3.4 Architecture and technical specification

This section provides an overview of the architecture and technical specifications of the FlawFix vulnerability management solution. The architecture is divided into different key areas: Frontend, Backend, Database, and Deployment. Each area is considered in terms of the technologies used and their role in the overall system.

3.4.1 Frontend

The frontend of the application is based on ReactJS, a popular JavaScript library for creating user interfaces. ReactJS offers a flexible and efficient way to design interactive UIs. Thanks to its component-based architecture, it enables a modular and maintainable code base.

NextJS, a powerful framework for server-side rendering (SSR), is used to support ReactJS. NextJS improves the performance of the application by providing fast loading speed and an optimised user experience.

TypeScript is used as the programming language for the front end. TypeScript extends JavaScript with typing, and thus offers improved code quality and security. It facilitates the development of complex applications and supports the team in maintaining and scaling the code.

3.4.2 Backend

The backend uses Golang, a modern programming language characterised by high performance and suitability for cloud-native applications. Golang offers efficient handling of parallel processes and is ideal for developing scalable and maintainable backend systems.

Communication between the frontend and backend takes place via REST APIs. These APIs enable a clear, understandable and well-documented interface for data exchange, which facilitates the integration and expansion of the system.

3.4.3 Database

PostgreSQL is used as the database management system. This powerful, open-source database is known for its reliability, robustness, and support for complex queries. It is ideal for applications that require complex data structures and high transaction rates.

3.4.4 Deployment

A Helm Chart is used to deploy the application in Kubernetes environments. Helm simplifies the management of Kubernetes applications and enables efficient, reproducible and scalable deployment. Helm Charts can be used to define and manage the entire application, including its dependencies and configurations, as a package. Outside of Kubernetes environments, deployment is possible using Docker and Docker-Compose. Overall, FlawFix is intended for operation in containerised environments. This form of operation is expected to be particularly popular with companies that already use modern development methods such as DevSecOps.

To summarise, the technical architecture of FlawFix provides a robust, scalable and future-proof basis for vulnerability management. The use of modern technologies and frameworks ensures that the solution not only fulfils current requirements, but is also flexible enough to adapt to future developments.

4 Unique Selling Points (USPs)

Although vulnerability management solutions already exist, vulnerability management is a process that requires new innovation. The advantages of the FlawFix vulnerability management solution are described in detail in this chapter.

4.1 European

The range of vulnerability management solutions is very limited, especially in Europe. Most manufacturers of such software products are companies based in the USA. The use of such products as software-as-a-service services is only possible if the regulations applicable in Europe are taken into account. A vulnerability management solution based in Europe strengthens digital sovereignty and simplifies regulatory guidelines.

4.2 Open source development

The majority of the software distributed is developed as closed source. This means that the source code of the programme cannot be viewed. A high degree of trust in the software manufacturer is necessary due to the lack of transparency, particularly in the case of sensitive data such as a company's vulnerabilities. The open-source development of the vulnerability management solution means that every company is free to enquire about the internal workings of the software. An active community that forms around the open source project can constantly improve the vulnerability management process through its innovative power and respond to new technical challenges. This continuous development is particularly important in the IT security sector, as the threat landscape is subject to constant change. The disclosure of the source code not only promotes trust, but also actively contributes to the security of the vulnerability management solution, allowing vulnerabilities in the product itself to be proactively identified and rectified. Open source enables long-term availability and independence from individual software providers, thus promoting the company's digital sovereignty.

4.3 Usability

Ease of use is a key feature of the FlawFix vulnerability management solution. This feature addresses the common challenge that getting started with vulnerability management is difficult for many organisations, especially small and medium-sized ones, due to technical complexity and resource constraints. FlawFix stands out here due to its user-centred design and ease of use.

User-friendly interface design FlawFix attaches great importance to an intuitive and easy-to-understand user interface. This enables users without in-depth technical knowledge to get started quickly and easily with the system.

Integration into existing IT infrastructures The solution is designed so that it can be seamlessly integrated into a company's existing IT systems with minimal effort. This significantly reduces the barriers to entry.

Documentation and support FlawFix provides detailed instructions and accessible customer service to make it easier for users to get started and use the software on an ongoing basis.

User-friendliness is therefore a decisive factor that sets FlawFix apart from other solutions and makes it particularly attractive for small and medium-sized companies. It ensures that dealing with IT security issues is not only effective, but also accessible to any company, regardless of its size and technical expertise.

4.4 Cost efficiency

Existing vulnerability management software is usually sold commercially. The prices of such software are high and difficult for small and medium-sized companies to afford. However, these companies are particularly at risk [4]. Due to the open-source character, every company is free to operate the software itself without licence costs. In addition, a variant operated by the manufacturer is to be sold commercially as software-as-a-service.

4.5 Focus on DevSecOps

The way in which software is developed has changed recently, particularly due to the progressive establishment of DevOps principles. Whereas in the past software was released on an annual or semi-annual basis, updates and new functions are now released weekly or even daily [3]. DevSecOps has established itself as an approach to keep pace with these developments from a security perspective. At the centre of this process is central vulnerability management, which must meet the new requirements of the DevSecOps concept [11].

4.6 Focus on data protection and security

Since FlawFix manages information about the vulnerabilities of the software used by organisations, the data of users has a particularly high need for protection. The FlawFix team is therefore focussing on developing an exceptionally robust, secure and data-efficient application. In cooperation with a local university, it is planned to investigate ways in which FlawFix can work as encrypted as possible and integrate confidential computing approaches. The open-source nature of FlawFix allows security researchers to conduct an independent review.

5 Differentiation from existing systems

5.1 DefectDojo

The FlawFix vulnerability management solution differs in several aspects from DefectDojo, an established tool in the field of application security and DevSecOps.

5.1.1 Focus and target group

DefectDojo aims to be a comprehensive platform for application security management and integrates with over 160 security tools. It is an open-source project that focuses on automating security testing and vulnerability management.

FlawFix, on the other hand, focuses specifically on vulnerability management and offers a customised solution specifically designed to meet the needs of companies that require agile and efficient vulnerability handling.

5.1.2 User experience

While DefectDojo offers a variety of integrations and features, the user experience, especially for people without IT security experience, is not as distinctive as with a specialised solution like FlawFix.

FlawFix places great emphasis on an intuitive user interface and adaptability to the specific needs and processes of each organisation, which ensures efficient and user-friendly handling of vulnerability management.

5.1.3 Scalability and performance

DefectDojo is designed to be a robust solution suitable for a wide range of security and development requirements. FlawFix, on the other hand, emphasises cloud-native technologies and thus offers high scalability and performance, especially in dynamic and distributed cloud environments.

Overall, FlawFix offers a specialised alternative to DefectDojo, with a particular focus on modern technologies and user-friendly design for specific business requirements. While DefectDojo offers a wide range of features for application security, FlawFix aims to be a specialised, agile and efficient solution for vulnerability management.

5.2 ArcherySec

In this comparison, FlawFix and ArcherySec are compared in terms of their focus areas and target groups, user experience, scalability, and performance.

5.2.1 Focus and target group

ArcherySec is focused on integrating a wide range of security tools and provides capabilities for comprehensive security and vulnerability analyses, including web, network, and infrastructure scans. It is ideal for a broad audience that requires complex integration capabilities and detailed analysis features, including teams working with a variety of security tools.

In contrast, FlawFix focuses on simplifying vulnerability management and supporting security standards such as DIN ISO/IEC 27001, and is specifically designed for developers and operations teams looking to adopt or introduce DevSecOps practices, as well as IT and security teams, executives and compliance officers who need a holistic view of their organisation’s security landscape.

5.2.2 User experience

ArcherySec offers various features and integrations, which can lead to a more complex user interface. This makes it a powerful but potentially more technically demanding system for users who need detailed reports and analyses of various vulnerabilities and threats.

FlawFix, on the other hand, places a strong focus on user-friendliness and intuitive operation. It is specifically designed for users without an in-depth IT security experience and offers a clear, easy-to-understand interface that facilitates the integration of security aspects into daily workflows.

5.2.3 Scalability and performance

With its wide range of integrations and features, ArcherySec is a flexible and powerful tool that is well suited to complex security environments and large organisations that require comprehensive coverage of various security aspects.

FlawFix, on the other hand, focuses on efficient integration into modern IT infrastructures and cloud environments. By using a modern technology stack, it offers high scalability and performance, especially for organisations looking for an agile and responsive security solution.

In summary, FlawFix is a specialised, user-centric solution that is particularly suitable for companies looking for a simplified, efficient implementation of security standards and vulnerability management. It represents an alternative option to ArcherySec by addressing a different approach and target audience, with a strong focus on simplifying processes and improving cybersecurity in organisations looking to deploy or adopt DevSecOps practices.

6 References

[1] Cvss v4.0 specification document. https://www.first.org/cvss/v4.0/specification-document. (Accessed on 01/14/2024).

[2] Known exploited vulnerabilities catalog — cisa. https://www.cisa.gov/known-exploited-vulnerabilities-catalog. (Accessed on 01/14/2024).

[3] What is devsecops? — ibm. https://www.ibm.com/topics/devsecops. (Accessed on 01/09/2024).

[4] Bundesamt für Sicherheit in der Informationstechnik. Lagebericht zur it-sicherheit in deutschland 2023. Technical report, Bundesamt für Sicherheit in der Informationstechnik, Bonn, Deutschland, 2023.

[5] Center for Internet Security. Cis controls version 8. Center for Internet Security, 2021.

[6] International Organization for Standardization. ISO/IEC 27001:2022 (E) Information security, cybersecurity and privacy protection — Information security management systems - Requirements. International Standard, 2022.

[7] Joint Task Force Transformation Initiative Interagency Working Group. Security and privacy controls for federal information systems and organizations. NIST Special Publication 800-53, Rev. 5, National Institute of Standards and Technology, Gaithersburg, MD, 2020.

[8] Karen Scarfone (Scarfone Cybersecurity) Murugiah Souppaya (NIST). Sp 800-40 rev. 4, guide to enterprise patch management planning: Preventive maintenance for technology — csrc. https://csrc.nist.gov/pubs/sp/800/40/r4/final. (Accessed on 01/15/2024).

[9] Håvard Myrbakken and Ricardo Colomo-Palacios. Devsecops: A multivocal literature review. In Antonia Mas, Antoni Mesquida, Rory V. O'Connor, Terry Rout, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 17–29, Cham, 2017. Springer International Publishing.

[10] Open Web Application Security Project. Owasp vulnerability management guide (ovmg). Open Web Application Security Project (OWASP), 2020.

[11] Open Web Application Security Project. Owasp devsecops guidelines, 2024. Accessed: 2024-01-09.

[12] Payment Card Industry Security Standards Council. *Payment Card Industry Data Security Standard*, 2022.