Incremental Common Criteria certification processes using DevSecOps practices

# Incremental Common Criteria Certification Processes using DevSecOps Practices

Sébastien Dupont*, Guillaume Ginis*, Mirko Malacario†, Claudio Porretti†, Nicolò Maunero ‡§
Christophe Ponsard*, and Philippe Massonet*
*CETIC, Charleroi, Belgium, {sebastien.dupont, guillaume.ginis, christophe.ponsard, philippe.massonet}@cetic.be
† LEONARDO, Rome, Italy,{claudio.porretti, mirko.malacario}@leonardocompany.com
‡ Dipartimento di Informatica e Automatica, Politecnico di Toriono, Turin, Italy, nicolo.maunero@polito.it
§ Cybersecurity National Laboratory, Consorzio Interuniversitario per l'Informatica (CINI), nicolo.maunero@consorzio-cini.it

***Abstract*—The growing digitalisation of our economies and societies is driving the need for increased connectivity of critical applications and infrastructures to the point where failures can lead to important disruptions and consequences to our lives. One growing source of failures for critical applications and infrastructures originates from cybersecurity threats and vulnerabilities that can be exploited in attacks. One approach to mitigating these risks is verifying that critical applications and infrastructures are sufficiently protected by certification of products and services. However, reaching sufficient assurance levels for product certification may require detailed evaluation of product properties. An important challenge for product certification is dealing with product evolution: now that critical applications and infrastructures are connected they are being updated on a more frequent basis. To ensure continuity of certification, updates must be analysed to verify the impact on certified cybersecurity properties. Impacted properties need to be re-certified. This paper proposes a lightweight and flexible incremental certification process that can be integrated with DevSecOps practices to automate as much as possible evidence gathering and certification activities. The approach is illustrated on the Common Criteria product certification scheme and a firewall update on an automotive case study. Only the impact analysis phase of the incremental certification process is illustrated.**

***Index Terms*—common, criteria, devops, devsecops, certification, incremental, security, cybersecurity**

## 1. Introduction

The growing digitalisation of our economies and societies is driving increased connectivity of critical applications and infrastructures. The increasing reliance on these applications and infrastructures means that any failure can lead to important disruptions and important business, safety, environmental or systemic consequences. One important source of failures is due to the increasing number of cybersecurity attacks on connected infrastructures and essential services. The growing digital economy is providing high value connected targets for cybersecurity attackers.

Cybersecurity certification is one approach to verifying that critical applications and infrastructures that organisations and citizens rely on for their daily activities are sufficiently protected and can be trusted. With the entry into force of the EU Cybersecurity Act on June 27 2019, a EU wide cybersecurity certification framework is under definition for information and communication technology (ICT) products, services, and processes. One of the motivations for the adoption of this new EU regulation is that "the limited use of certification leads to individual, organizational and business users having insufficient information about the cybersecurity features of ICT products, ICT services, and ICT processes, which undermines trust in digital solutions." The Cybersecurity Act aims to improve trust in products, services, and processes by defining an EU-wide certification framework consisting of cybersecurity certification schemes that specify common cybersecurity requirements and evaluation criteria across national markets and sectors.

The underlying assumption in the EU cybersecurity act is that products, services and processes that are certified will be viewed as more trustworthy by users. Companies going through certification would thus benefit from a competitive advantage. Cybersecurity certification suffers from an image as a costly and time consuming process. One of the keys to success for the EU cybersecurity act is to promote standards that are lightweight, flexible and incremental in order to encourage organisations to voluntarily certify their ICT products, services, and processes.

However, reaching sufficient assurance levels for product certification may require detailed evaluation of product properties. Cybersecurity product certification can be time consuming and costly. An important challenge for product certification is dealing with product evolution: now that critical applications and infrastructures are connected they are being updated on a more frequent basis. To ensure continuity of certification, updates must be analysed to verify the impact on certified cybersecurity properties. Impacted properties need to be re-certified.

This paper proposes a lightweight and flexible incremental certification process that can be integrated with DevSecOps practices to automate as much as possible evidence gathering and certification activities. The approach is illustrated on the Common Criteria product certification scheme and a firewall update on an automotive case study. Only the impact analysis phase of the incremental certification process is illustrated. This paper refines previous work [1] on the topic of incremental cybersecurity certification.

Section 2 presents product certification challenges

*A preprint of this paper has been deposited on ArXiv.*

linked to continuity of product cybersecurity certification and their continuous evolution, and proposes some requirements for flexible incremental certification. Section 3 presents DevSecOps practices and how they impact development and evolution processes. Section 4 presents models of the incremental certification for Common Criteria and the DevSecOps processes, and shows how they can be composed together. Section 5 illustrates the approach for the impact analysis part of incremental certification for a firewall update in an automotive case study. Section 6 discusses process oriented certification which is an alternative to product oriented certification, and other approaches to incremental product certification.

## 2. Challenges for incremental product certification

This section provides a brief overview on the evolution of security certifications for IT products in consideration of their continuous evolution, identifying some requirements that can facilitate a flexible incremental certification process.

### 2.1. Product certification and product evolution

The continuous technological evolution has brought huge changes within our lives, now fully immersed in IT systems (e.g. cell phones, PCs, televisions, cars). The distance between Operational Technologies (OT) and ICT world is shortened, improving our lives but requiring paying more attention in security field. Cybersecurity therefore becomes essential to protect the information that all these interconnected devices use and share. Since the 1980s, there has been a growing need to submit IT products to security certifications in order to assess, impartially, the cybersecurity posture of a product. At that time the DoD developed the Trusted Cyber Security Assessment Criteria, also known as the "Orange Book" [2] which described how to perform a security certification for IT systems, in order to ensure reliability on security enforcing measures deployed to protect information. In the 1990s the European Security Standard named "Information Technology Security Evaluation Criteria (ITSEC)" [3] was issued, followed at the turn of the 2000s by the Common Criteria for Information Technology Security Evaluation standard (now in its version 3.1 revision 5 [4]). The security certification aims to assess the effectiveness of security countermeasures implementations (Target of Evaluation - TOE) by analysing a defined threat scenario, evaluating the robustness of the mechanisms implemented within a defined operating environment. The strength of a security certification therefore lies in the fact that a third party (Evaluation Laboratory) guarantees an impartial examination of the evidences produced by the developer. Finally, an authority called the Certification Body supervises the certification and issues the certification. In a more static world, modelled by a more static development process, this approach guarantees a good confidence on the implementation effectiveness of the security mechanisms. Today, in a world where the products are constantly evolving, due for example to the continuous need for technological updates or the release of vulnerability patches, there is the need to face challenges of guaranteeing the maintenance of the security certification in view of the changes occurred.

### 2.2. Current practices of incremental certification with Common Criteria

The need to shape security certifications to the continuous evolution of IT products was therefore addressed by Common Criteria within the so-called Common Criteria Assurance Continuity [5]. Assurance continuity aims to define a mutually recognized approach within the Common Criteria for the maintenance and re-evaluation of certified products. The assurance continuity defines a process for carrying out an impact analysis intended to assess the level of changes that have occurred to a certified product. These changes are categorized into minor and major, with the result of having to carry out a number of different evaluation activities in terms of effort and time on the certified product under evaluation, up to the need for a new certification. The assurance continuity activities involve all the players who take part in a Common Criteria certification, starting with the sponsors and developers, passing through the Evaluation Laboratory and finally the Certification Body. The standard therefore provides a methodological approach to carrying out the impact analysis while not defining specific supporting tools or methodologies.

### 2.3. Requirements for flexible incremental certification processes

It is therefore important to identify requirements that make the incremental certification process more flexible, even if well structured, in order to have advantages in terms of time and cost. The first requirement that we can consider (as described in [1] and [6]), is to adopt a structured security development process which would allow to guarantee a greater assurance of the certified product since the earliest stages of requirement definition. At the same time, an agile process must be put in place to support the certification in all phases of the life-cycle, considering also the very important maintenance phase (patch management and improvement). To this extent, methods and technologies such as DevSecOps can greatly help in reducing costs and times by supporting the developer in the process of creating and maintaining a certified product, in carrying out Assurance Continuity activities and in the production of the Impact Analysis Report. The evidences of certification could be partially produced by means of DevSecOps' automatic tools, by allowing a reduction of costs and times. This would allow the developer to take into account the impacts of the changes in progress to the certified product and maintained simpler during the entire life-cycle development. Moreover, the evaluator can assess more quickly the impacts and can obtain the necessary evidences for the maintenance of the certification. From the point of view of the evaluator, a good development process (including the maintenance phase) would also ensure faster access to the evidences necessary for the evaluation of an impact analysis. One example of efficiency improvement could be related to

the management of minor changes. In a structured semi-automatized process, changes not affecting the Target of Evaluation could be evaluated faster. The assurance of the DevSecOps process could be verified by the evaluation laboratory during the Common Criteria evaluation of the product in order to ensure its effectiveness for the production of certification evidences (also when modified for the assurance continuity). Please note that this would greatly help concurrent evaluations as well (i.e. evaluations conducted during product development).

## 3. DevSecOps practices

In recent years, the need to improve software delivery in terms of speed and quality has given rise to a set of practices that combine continuous build, testing, integration, delivery, ... The DevOps approach, closely related to Agile software development method, integrates software development ("Dev") and operations ("Ops") processes to ensure that new features are added to a software solution in the shortest time possible, and with a high level of quality. This approach emphasizes the importance of communication between the involved parties, including the whole production chain (developers, system administrators, network team, ...), to break the classic "silos" of specialists. DevOps relies on the "CAMS" [7] (Culture, Automation, Measurement, Sharing) characteristics and on a "shift to the left" where aspects such as resilience or security are taken into account as soon as possible in the software development life cycle.

DevOps is focused on producing quality code, quickly and reliably. The security problematic is not directly addressed in this approach and DevSecOps is aiming at complementing DevOps with security procedures to ensure continuous security assessment.

The benefits of DevSecOps can impact software development in various ways. The left-shift in security integration provides a better approach to security by intervening earlier in the deployment cycle and thus detecting security issues sooner, similarly the automation of security enables a continuous monitoring of the system where vulnerabilities are detected with minimal human intervention [8]. DevSecOps also provides value by reducing the cost of making mistakes, detecting them, investigating their cause and fixing the problems [9]. Finally, security concerns are among the major hurdles that limit the adoption of DevOps processes [10], DevSecOps proposes tools and methodologies to ease this friction.

DevSecOps activities include for example threat modeling and risk assessment, continuous vulnerability assessment through static code analysis (SAST, SCA) and dynamic testing (DAST, penetration, scanning, security drills, Red Team assessment, ...). Figure 1 illustrates an example integrated process provided by DevSecOps where the security activities are aligned with the DevOps software life cycle phases.

A typical DevOps pipeline iterates through the following phases:

- PLAN - Planning activities take place before coding work starts, they include requirements gathering, establishing a road map, task allocation based on the backlog of change requests and bugs, etc.

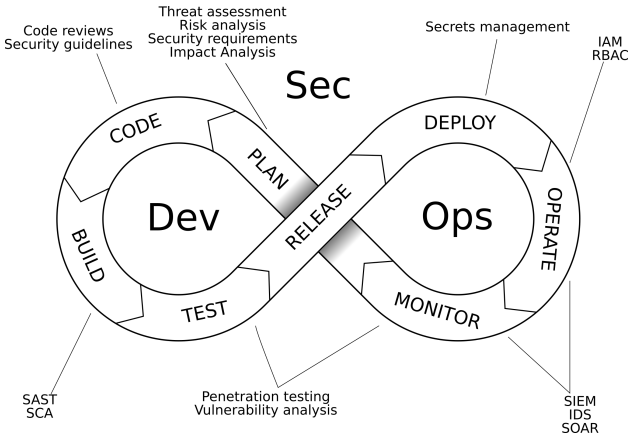


Figure 1. DevSecOps - Security activities integrated with development and operations

- CODE - Implementation of the new features, the code is pushed to a source control repository.
- BUILD - The new code is compiled and packaged into a package or image.
- TEST - User acceptance, performance and load tests are performed in testing environments in which the new version of the system is installed. Testing environments are ideally built following an "Infrastructure as Code" (IaC) approach where the environment is entirely defined through machine-readable configuration files.
- RELEASE - The build is ready for deployment to production, and made available for distribution in an artifact repository.
- DEPLOY - The software is deployed to production, using the same IaC approach used to provision the testing environments.
- OPERATE - The new version of the software runs in production, maintenance activities include performance or incidents troubleshooting.
- MONITOR - Data is collected to provide insights on user behaviour, performance, errors, ...

The following security activities relate to the DevOps phases described above, this list is not exhaustive and can vary depending on the level and strategies of integration:

- PLAN - Threat assessment and risk analysis to identify and manage threats and risks, security requirements gathering and impact analysis to identify the changes needed to the security posture.
- CODE - Code reviews by peers, coding security guidelines, ...
- BUILD - Static Analysis Security Testing (SAST) to find common security flaws, Software Composition Analysis (SCA) to automate the identification of third party library dependencies for detecting external vulnerable code.
- TEST - Vulnerability analysis and penetration testing to find the system's vulnerabilities and exploit them. This provides insights on threats and associated risks.
- RELEASE and DEPLOY - Secure delivery mechanism with artifacts repositories

- OPERATE and MONITOR - Security Information and Event Management (SIEM), Security Orchestration, Automation and Response (SOAR), Intrusion Detection Systems (IDS)

Integrating security with software development activities poses specific challenges that can explain the lack of security testing in CI/CD workflows, for example

- Speed of detection and remediation vs speed of delivery: those two seemingly opposing goals, "speed of delivery" and "secure code" need to be merged into one streamlined and automated process so that the system security is ensured without slowing down the delivery process [11]
- Seamless integration and adoption in the development workflow: use the same or similar tools, automations and procedures that are used in DevOps to provide an homogeneous continuous delivery environment [12].
- Prevention, detection, remediation and traceability of unknown security issues:
  - Dependencies analysis: IT systems are built using potentially lots of external libraries, sometimes implicitly. There is a need to automatically detect those dependencies and analyse them
  - Auditing and compliance activities: regulations like HIPAA, GDPR and SOX, or standards like PCI-DSS, ISO:27001 or Common Criteria provide guidance and rules on security requirements, those activities need to be integrated with the software development life cycle and automated to facilitate their application.
- Organisational culture: help create the mindset in the enterprise that everyone is responsible for security, integrate good practices on how to organise and communicate quality information regarding security [13]

## 4. Modeling certification processes

In this section we model the Common Criteria certification process and the DevSecOps process with activity diagrams, and show how they can be composed for the impact analysis phase of these processes.

### 4.1. Modeling certification processes

Common criteria certification [4] defines a process that includes the following classes:

- ASE (Security Target Evaluation): this class deals with the evaluation of the consistency of the Security Target which also contains the definition of the security requirements of the TOE.
- ADV (Development): this class deals with the evaluation of the six families of requirements for structuring and representing the security functionality realized by the TOE at various levels and varying forms of abstraction that the developer must produce during the product development phase.
- AGD (Guidance Documentation): this class takes care of the evaluation of the manuals that are delivered to the customer.
- ALC (Life-cycle support): this class evaluates all aspects of the management of the TOE during its life cycle; it includes maintaining the certification via security patch management.
- ATE (Tests): it is the class that takes into consideration all the tests that demonstrate that security functionalities operate according to their design descriptions.
- AVA (Vulnerability Assessment): this class takes care of vulnerability assessment activity to analyse vulnerabilities in the development and operation of the TOE.

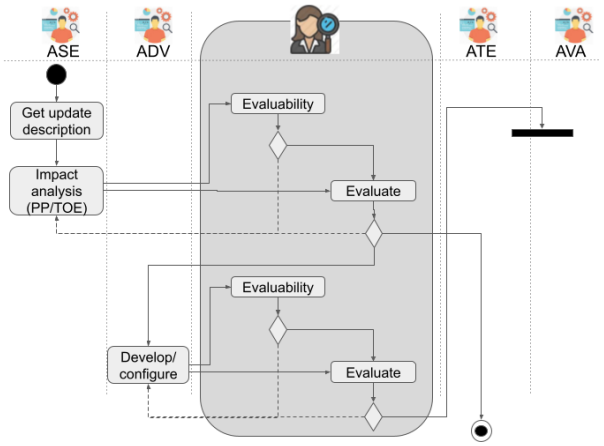


Figure 2. Certification process activity diagram

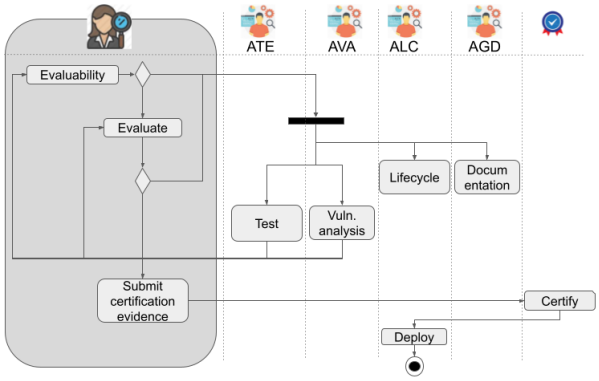


Figure 3. Certification process activity diagram

Figure 2 and Figure 3 show an activity diagram that models the incremental certification activities of the product owner, the Common Criteria evaluator and the accreditation body, i.e. that delivers the official certification. The pattern of interaction for each class is the following:

- The product owner carries out normal security engineering activities and produces certification evidence for the evaluator;
- Once the certification evidence is available, the evaluator is notified and performs an evaluability

check. In the evaluability check the evaluator verifies that the necessary evidence is available;
- If the evaluability check is positive, then the evaluator performs the evaluation on the available evidence.

In Figure 2 the incremental certification process is triggered by the arrival of a new version of the software. Within the ASE class a description of the update is made and impact analysis can start. The goal of impact analysis is to determine if incremental certification is necessary. Once impact analysis evidence has been created, the evaluator makes the evaluability check. If it is positive, then the evaluation of impact analysis evidence starts. If the evaluability check is negative then impact analysis must continue until it produces the required evidence. If the evaluation of impact analysis ASE evidence is positive, then incremental certification then moves to the ADV class with the same type of pattern of evaluation. If the ASE evaluation is negative, then impact analysis must continue until the necessary evidence is created. During the development phase certification evidence is produced and once the developer considers the evidence as complete, the evaluator starts the evaluability check. If the evaluability check is positive, i.e. all the required evidence is available, the evaluator starts the evaluation of the ADV evidence. If the evaluation is positive, then evaluation of ATE testing and AVA vulnerability evidence can start.

Figure 3 shows the incremental certification process for the other Common Criteria classes. The same pattern of evaluability and evaluation applies to the ATE, AVA, ALC and AGD classes (the two latter are not shown). Once the evaluator has validated the evidence from all classes, he submits it to the accreditation body that reviews the evidence and delivers the certification for the product or service with respect to the security requirements specified in the protection profile (PP) or target of evaluation (TOE).

## 4.2. Modeling DevSecOps processes

Figure 4 shows a sample activity diagram of DevSecOps activities in the context of continuous integration and deployment on a autonomous rover. Each step provides results (logs, reports, ...) that can be used as certification evidence. The sequence reads as follows:

- Upon modifying the software, the developer commits the modifications to a code repository. If any of the subsequent steps fail, the process restarts from here with modifications to the code and configuration.
- Static security tests such as source code or dependency analysis are run on the source code. The Static Analysis Results Interchange Format (SARIF) and the Static Analysis Server Protocol (SASP) [14] can facilitate the integration of the SAST output with other tools.
- If static tests succeed, artifacts are built, deployed into a sandbox and dynamic security tests (DAST) are applied to the environment: vulnerability analysis, penetration testing, etc.
- If dynamic tests are successful, a risk analysis can be performed on the system. It will take as input the various outputs of the previous steps (vulnerability analysis report, code analysis report, etc.) and produce an updated risk for the software version being released. If the risk is not acceptable, steps must be taken to mitigate it, for example by improving security monitoring or enforcing stricter security rules.
- The packaged application is published in an artifact repository, and deployed to the production environment.

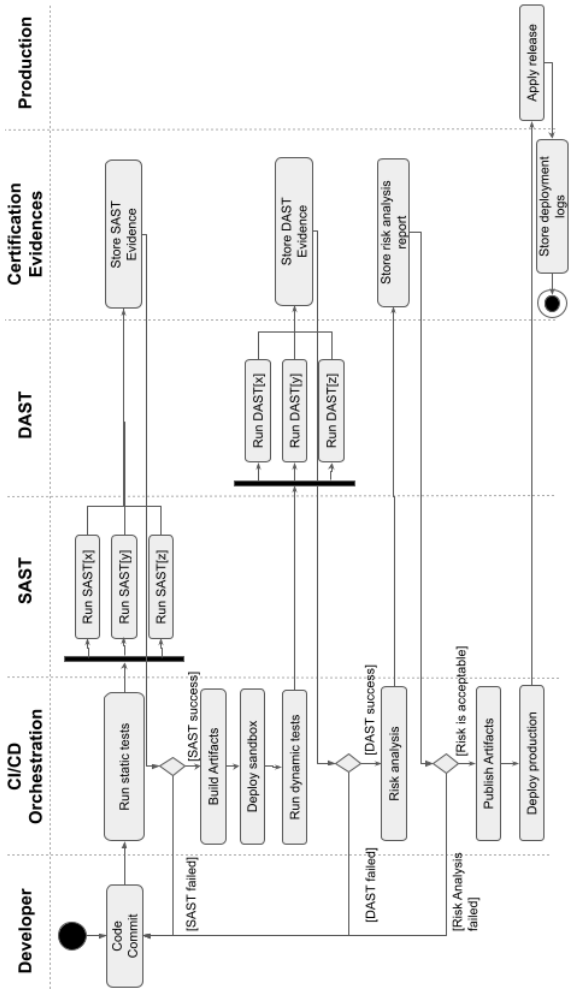


Figure 4. DevSecOps process activity diagram - code, build, test, release, deploy

## 4.3. Composing incremental certification and DevSecOps processes

Figure 5 shows the composition of the incremental certification process model with the DevSecOps process model for the impact analysis part of the process. The two processes evolve in parallel but interact in the following general manner: the DevSecOps process produces evidence for the certification process, and the certification process will authorize the deployment if incremental certification is required. The incremental certification process is triggered by a change request to update the deployed system with a new version of a component. The change

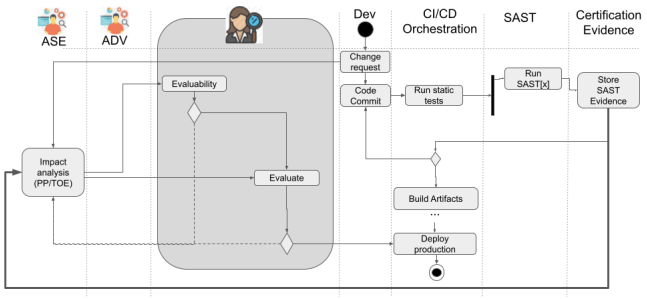


Figure 5. Composed certification and DevSecOps activity diagram - Impact Analysis

request triggers the beginning of the impact analysis in the certification process. In the DevSecOps process the change request triggers the code commit of the source code and the start of static analysis activities. Part of the static analysis results are stored as certification evidence for the impact analysis activity. The arrow between the "Store SAST Evidence" activity and the "Impact analysis" activity will trigger the checks on the certification evidence: when all impact analysis certification evidence is available, the evaluability check by the evaluator can start.

This composed process is illustrated on a concrete firewall update example in Section 5.3: the example shows how the impact analysis evidence is analysed to determine if incremental certification of a automotive platooning system is required. The "Store SAST Evidence" will produce the evidence illustrated in Figure 9 that will be used by the "Impact analysis" activity to produce the tables 4 and 5. The same type of interaction between the certification and DevSecOps processes occur for all Common Criteria classes, but this is not described in this paper and left for future work.

With the automation of security testing, DevSecOps already provides some foundation for automating certification. For example static and dynamic security tests produce reports in an automated way, those reports can be used as certification evidence in an automated certification process. Development documents such as architecture, design, specifications, security model, ... are usually produced by hand, which can limit automation of the DevSecOps and certification activities. Solutions for improving the integration of those evidences in a continuous process include the use of model based approaches to generate the evidences: infrastructure as code for the architecture, Business Process Model and Notation, etc.

## 5. Illustration of benefits with DevSecOps for Common Criteria impact analysis

### 5.1. Firewall case study

In this section the incremental certification process is illustrated on an automotive platooning case study. Platooning is a method for a leader vehicle driving a group of vehicles behind it. All the vehicles in the platoon communicate together, and the leader communicates with a traffic control center. Platooning is a safety critical system and it is important to protect communications. The communications of each vehicle are protected by a firewall. The firewall is certified with respect to a Common Criteria protection profile that defines the security requirements. However, firewalls need to regularly be updated with new software versions. In the firewall update scenario, a new version of the firewall is available and needs to be deployed on a vehicle. From the certification point of view, if some certified requirements are impacted then the new firewall version must be re-certified. The requirements listed in Table 1 below are an extract of the security requirements from the protection profile of the system. The security requirements might be impacted by some of the code changed in the new version of the firewall and this will determine the type of re-evaluation to be performed on the new version of the system including this firewall update. We assume that the evidences were provided by an iteration of the DevSecOps cycle for the TOE full certification on which we rely on for the incremental one. Thus, a new iteration of the cycle will provide the same updated evidences.

The changes taken as example for the firewall update are listed in Table 2 and extracted from [15].

| Requirement | Description |
|---|---|
| FDP_ACF.1.1 | "The TSF shall enforce the access control to objects based on security attributes." |
| FDP_ACF.1.2 | "The TSF shall enforce rules to determine if an operation among controlled subjects and controlled objects is allowed." |
| FDP_ACF.1.3 | "The TSF shall explicitly authorise access of subjects to objects based on additional rules." |
| FDP_ACF.1.4 | "The TSF shall explicitly deny access of subjects to objects based on the rules." |
| FDP_IFF.4.1 | "The TSF shall enforce the information flow control to limit the capacity of illicit information flows to a maximum capacity." |
| FDP_IFF.4.2 | "The TSF shall prevent the following types of illicit information flow : tcp shell or http shell." |
| PMM_IF.1.1 | "The TOE shall maintain an outgoing heart-beat data flow with other platooning vehicles as specified below: From TOE to VCS (and then to another vehicle TOE). Messages transmitted shall contain the following data computed from the TOE vehicle sensors/algorithms: Vehicle unique identifier - Vehicle speed - Direction - Geo-Position - Timestamp." |
| PMM_IF.3.1 | "The TOE shall maintain an incoming flow with other vehicles informing the TOE vehicle about emergency brake maneouvers as specified below: From (another vehicle TOE to vehicle) VCS to TOE. Messages transmitted shall contain the following data: Unique identifier of the vehicle to which the emergeny brake has been issued - Emergency brake identifier - Timestamp - Digitally signed certificates." |

TABLE 1. SECURITY REQUIREMENTS

### 5.2. Implementation of DevSecOps process for impact analysis

After the delivery of a Common Criteria certified product, the Assurance Continuity procedure as defined in [16] can be used for maintenance and re-evaluation activities.

The Assurance Continuity procedure as described in [16] is made in 5 steps that allow to produce an Impact Analysis Report (IAR) used for the re-evaluation activities:

| ID | Summary | Description |
|---|---|---|
| 2 | xtables-monitor fix rule printing | trace_print_rule does a rule dump. This prints unrelated rules in the same chain. Instead the function should only request the specific handle. Furthermore flush output buffer afterwards so this plays nice when output isnt a terminal. |
| 3 | xtables-monitor fix packet family protocol | This prints the family passed on the command line which might be 0. Print the table family instead. |
| 4 | nft Optimize class-based IP prefix matches | Payload expression works on byte-boundaries leverage this with suitable prefix lengths. |
| 5 | nft Fix selective chain compatibility checks | Since commit 80251bc2a56ed chain parameter passed to nft_chain_list_get() is no longer effective. Before it was used to fetch only that single chain from kernel when populating the cache. So the returned list of chains for which compatibility checks are done would contain only that single chain. Re-establish the single chain compat checking by introducing a dedicated code path to nft_is_chain_compatible() doing so." |

TABLE 2. FIREWALL UPDATE ISSUES

- Step 1 - Identify Certified TOE
- Step 2 - Identify and describe change(s)
- Step 3 - Determine impacted developer evidence
- Step 4 - Perform required modifications to developer evidence
- Step 5 - Conclude

The DevSecOps process will contribute to these steps as it is the core of the configuration and of the changes implemented on the solution, system or software developed. As the same process was used for the initial certification, it can provide updated evidences necessary for building the Impact Analysis Report each time changes are implemented. Furthermore, it can speed up the production of evidences and the re-evaluation as all these steps are automated starting from a commit on the source code.

Figure 4 describes the sequence of security activities in the release starting from a source code commit resulting from a change request. In [17], we proposed a generic DevSecOps pipeline based on a selection of open source tools, Figure 6 implements the DevSecOps process from Figure 4 with a similar selection of tools:

- SAST: Frama-C is an extensible and collaborative platform dedicated to source-code analysis of C software. SonarQube is an automatic code review tool to detect bugs, vulnerabilities, and code smells. Eclipse Steady analyses Java and Python applications to identify, assess and mitigate the use of open-source dependencies with known vulnerabilities.
- DAST: OpenSCAP is a bundle of tools dedicated to compliance and vulnerability assessment, in particular OpenSCAP Base performs basic operations such as configuration verification or security scanning on local or remote systems. OpenVAS is an open vulnerability scanner and OWASP ZAP is an attack proxy for web apps.

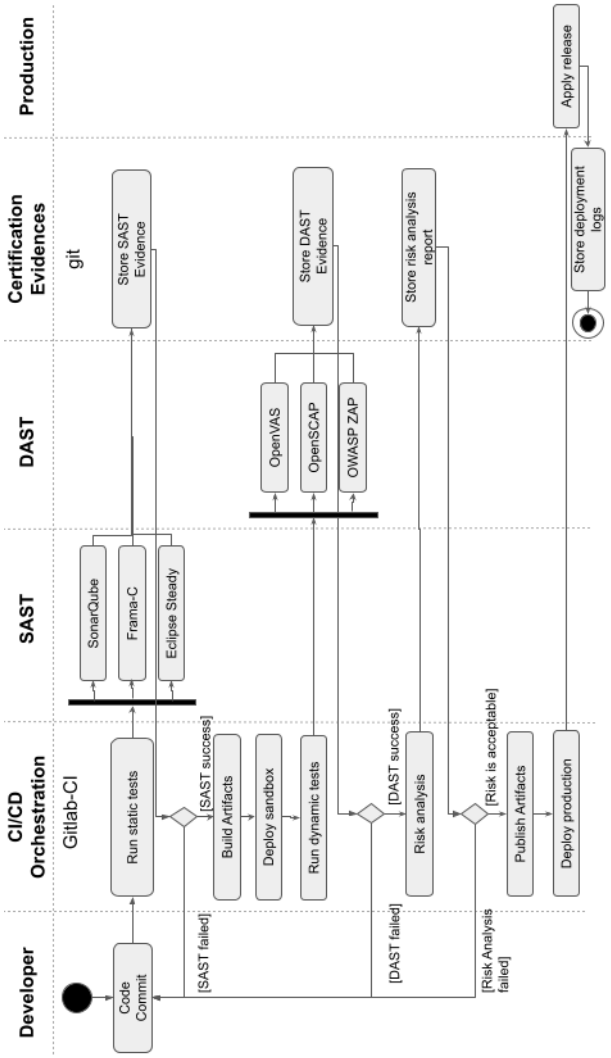


Figure 6. DevSecOps tools - code, build, test, release, deploy

- CI/CD Orchestration: GitLab CI/CD is a tool built for software development through continuous integration, delivery and deployment methodologies. Jenkins is one possible alternative.
- artifact repository: JFrog Artifactory or Sonatype Nexus provide artifact storage and distribution for various types of artifacts (APT, RPM, Docker, ...)
- certification evidence: various pieces of evidence can be stored into artifacts repositories or simply in version control systems (git, svn, ...).

Before the starting point of the DevSecOps pipeline described in Figure 6, other tools are used in the PLAN phase such as :

- a ticket/issue manager such as JIRA, Gitlab or Github is used to manage the changes requested and the problems or vulnerabilities detected;
- a threat modelling and risk analysis tool such as Threat Dragon or Threagile is used to model the threats on the system.

In the Assurance Requirement procedure [16], the process of the incremental certification or Common Criteria re-evaluation is described in a single step but, in order to

make it based on a DevSecOps cycle, the impact analysis report will preferably be generated in 2 versions for the 2 different purposes it is used :

- a preliminary version or draft will result from the PLAN phase of the DevSecOps cycle and provide the information necessary to decide the type of certification needed : none, incremental, full;
- a second version before the deploy phase of the cycle in order to have the evaluator certification before deployment of the update.

The preliminary version of the IAR should cover the 2 first steps described in [16] : Identify Certified TOE and Identify and Describe the changes. The first step is covered by the configuration stored in the DevSecOps tools in the Release phase. Artifact repository management tools, such as JFrog Artifactory, store the artifacts generated by the DevSecOps cycle and the configuration that was certified. The second step is covered by the result of the Plan phase of the DevSecOps cycle. In this phase, the issue at the origin of the change is analysed, identifying the work to be done in the standard DevOps cycle. Then, the impact on security components and requirements is identified through the threat assessment and risk analysis tools used in the additional security layer of the DevSecOps.

The results generated by the threat assessment and risk analysis tools would take the form of tables generated as in Section 5.3. The previous certification shall provide the link between Security Functional Requirements (SFRs) and the code and components of the solution (Table 3). This will be used as the basis of the threat assessment and risk analysis activities. The threat assessment and risk analysis tools will provide the list of impacted components of the solution, reducing the number of possible impacted SFRs (Table 4). Finally, it will provide the impacted SFRs and a pre-filled justification that will have to be completed manually (Table 5).

Concerning the second and final version, provided to the evaluator for the re-evaluation, it should cover the step 3 (determine impacted developer evidence) and 4 (perform required modifications to developer evidence) in a simple way. All the evidences generated by the DevSecOps tools are automatically generated so a new version of all evidences will be created and the differences with the previous set of evidence can be easily identified by the tools. Links to all these evidences can easily be added to the IAR. This final version of the IAR will also have to include a conclusion but this process will be done manually and is outside the scope of the DevSecOps process and tools.

Finally, the complete process is summarized in Figure 7. It shows a developed DevOps cycle with the corresponding artifacts generated by the activities of the cycle. The artifacts useful for the incremental certification are the following :

- PLAN : Starting from an issue in the ticket management system, this activity will provide the analysis of the issue in terms of effort, impact, etc;
- BUILD : This activity will perform unit testing, compile the code and provide a compiled application;
- TEST : This activity will perform functional testing on the application and provide test result;
- RELEASE : This activity will create a configured release of the application with its produced artifacts.

On this first DevOps layer, it adds the Security layer making a DevSecOps cycle this time with the additional activities performed for the security and the resulting artifacts.

- Threat Assessment and Risk Analysis : Starting from the issue in the ticket management system, this activity will provide the Impacted Security Requirements;
- SAST/SCA : This activity adds Static Tests to the unit tests in order to verify the security of the source code created and produce Static Test evidences;
- DAST : This activity add Dynamic Tests of the application to the functional tests in order to verify the application vulnerabilities and flaws and produce Dynamic Test evidences;

The complete description of the activities of these 2 first layers are described in chapter 3.

Finally, the third layer is the Common Criteria layer describing the interaction with the evaluator to perform a re-evaluation of the solution in an incremental way. The proposed way of working for this part is to work in 2 steps :

- a first one with preliminary evidences to decide the type of re-evaluation that will be performed;
- a second one including all evidences produced by the DevSecOps cycle to perform the evaluation and provide the renewed certification.

### 5.3. Examples of tables that can be generated

Table 3 provides the link between all the component of the certified solution and their associated Security Requirements. The information allowing to build this table should come from the first certification of the solution. If these associations are modified a complete re-certification is required. The Firewall contains 2 sub-components as described in Figure 8 : iptables which manages the configuration and netfilter which is the real-time component of the firewall.

| Component | Requirements |
|---|---|
| SafeSecPMM | PMM_IF.1.1 |
| SafeSecPMM | PMM_IF.3.1 |
| iptables | FDP_ACF.1.1 |
| netfilter | FDP_ACF.1.3 |
| netfilter | FDP_ACF.1.4 |
| netfilter | FDP_IFF.4.1 |
| netfilter | FDP_IFF.4.2 |
| iptables | FDP_ACF.1.2 |

TABLE 3. TRACEABILITY BETWEEN SECURITY REQUIREMENTS AND THE COMPONENTS THAT IMPLEMENT THE REQUIREMENT

Once the issue at the origin of the DevSecOps cycle analysed, the impacted components of the solution are known and it is possible to filter Table 3 to only provide the impacted Security Requirements as in Table 4.

Finally, Table 5, result of the threat modeling and risk analysis, provides the Security Requirements that

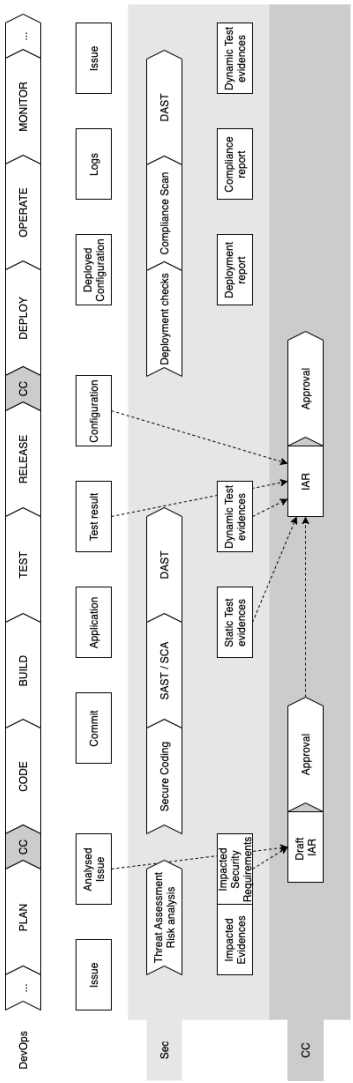


Figure 7. DevOps, DevSecOps and Incremental CC certification activities

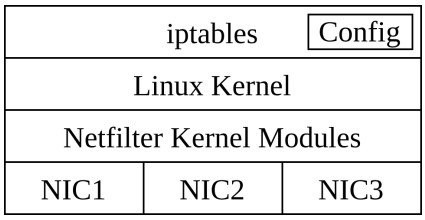


Figure 8. iptables architecture

are impacted by the changes. It provides a preliminary justification that will need to be completed manually.

For the generation of the tables 3, 4 and 5, a simple database was designed. The idea is to find :

- the minimum information necessary;
- the relationships between these pieces of information;
- how these pieces of information can be generated with the tools identified in the DevSecOps process.

The database model is presented in Figure 9.

The information concerning the components of the solution and the issues as well as their relationship is usually available in a ticket management system like Jira.

| Component | Requirements |
|---|---|
| iptables | FDP_ACF.1.2 |
| netfilter | FDP_ACF.1.3 |
| netfilter | FDP_ACF.1.4 |
| netfilter | FDP_IFF.4.1 |
| netfilter | FDP_IFF.4.2 |
| iptables | FDP_ACF.1.1 |

TABLE 4. TRACEABILITY BETWEEN SECURITY REQUIREMENTS AND THE IMPACTED COMPONENTS

| Issue | Requirement | Impacted | Justification |
|---|---|---|---|
| 2 | FDP_ACF.1.2 | False | "The changes to the code of the component do not affect the requirement as it concerns only display." |
| 3 | FDP_ACF.1.2 | False | "The changes to the code of the component do not affect the requirement as the requirement is not satisfied by this component." |
| 4 | FDP_ACF.1.3 | True | "The changes impact the component and the implementation of the requirement " |
| 4 | FDP_ACF.1.4 | True | "The changes impact the component and the implementation of the requirement " |
| 4 | FDP_IFF.4.1 | True | "The changes impact the component and the implementation of the requirement " |
| 4 | FDP_IFF.4.2 | True | "The changes impact the component and the implementation of the requirement " |
| 5 | FDP_ACF.1.3 | False | "The change to the code of the component do not affect the requirement as it is a compatibility change for checks." |
| 5 | FDP_ACF.1.4 | False | "The change to the code of the component do not affect the requirement as it is a compatibility change for checks." |
| 5 | FDP_IFF.4.1 | False | "The change to the code of the component do not affect the requirement as it is a compatibility change for checks." |
| 5 | FDP_IFF.4.2 | False | "The change to the code of the component do not affect the requirement as it is a compatibility change for checks." |
| 2 | FDP_ACF.1.1 | False | "The changes to the code of the component do not affect the requirement as it concerns only display." |
| 3 | FDP_ACF.1.1 | False | "The changes to the code of the component do not affect the requirement as it concerns only display." |

TABLE 5. IMPACT ANALYSIS RESULTS AND JUSTIFICATION

## 6. Discussion and future work

### 6.1. Cybersecurity product vs process certification

In a market where agility, reduced time to market and constant update are at the foundation of basically every software product or service (especially in the cloud ecosystem), security certification needs to adapt to this fast-paced environment. The major limitation of certifications, nowadays, is that they are still too statically linked to the product itself and to the particular version of the product with respect to which the certification is issued.

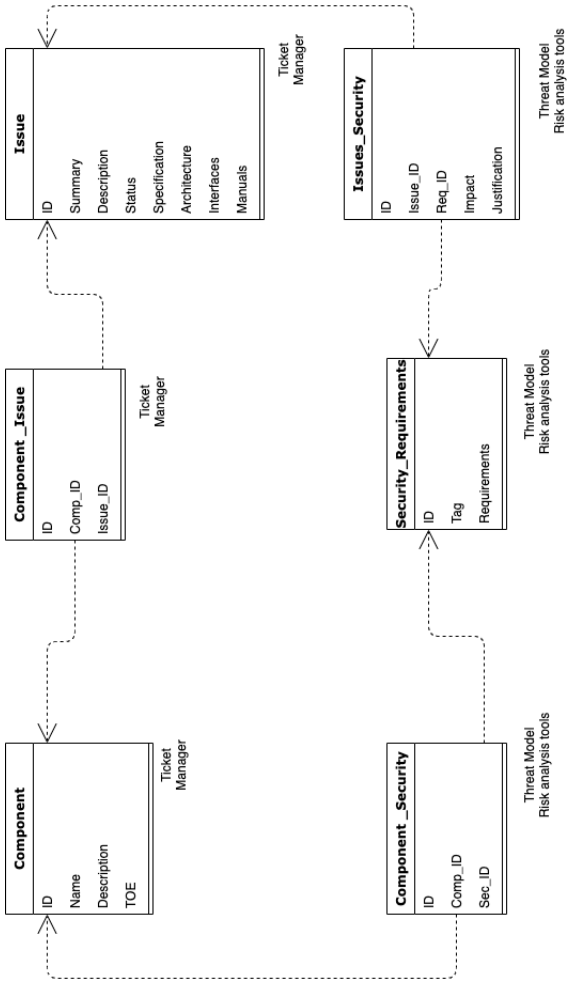


Figure 9. IAR minimal database model

In fact, the main problem, e.g., Common Criteria, is that it is virtually impossible to adopt this type of certification for products that see a new release, and therefore a total re-certification, practically every day. To overcome this problem, the adoption of an incremental approach has been growing: after the certification of a given product, it will be no longer necessary to re-issue a complete certification process after each update or modification, but only those modified part of the product, that invalidate the certification, are put through the certification process, avoiding to re-issue the process for the entire product. However, a ”classic” approach to incremental certifications is not yet sufficient to guarantee the necessary flexibility in the modern market. It is here that the proposal to certify not the product, but the process of developing a product finds its space [18]. Therefore, the objective is to try shifting the focus of the certification process not to something that is constantly evolving, the product, but to something that currently sees less sudden changes, which is the development process used. A *process-centric* methodology has the potential to easily scale and adapt to an agile development model, evaluating how a product or service was developed and used with respect to what was developed. Focusing on the development process it is possible to gather all the necessary details and information on how a product is developed, hence also how the best practice and methodologies, in terms of cybersecurity, are adopted and applied. By certifying the development process it is possible to guarantee, to a certain degree, the security level of a given product, while more classical certification schemes (e.g. Common Criteria) should be applied only for the product or system integrated in high risk environment, where a longer lifetime for the installed technology components is expected.

## 6.2. Other approaches to incremental certification

An early approach to incremental certification was to consider a full certification was too much to achieve in one step [19]. This was proposed at a time were security was far less critical and companies has a low level of maturity to implement security certification. Nowadays such an approach is addressed through the notion of risk in a process-oriented certification approach described previously. The ISO 27005 actually defines the risk management as an iterative process. In a given context, risks can be partially mitigated, ignored or transferred but there is also a monitoring activity that will trigger new risk analysis and treatment process [20]. Nowadays step-wise partial certification is not recommended in a certification context but it can make sense in a more general process, for example in a SME context it can lead to certification by first considering more lightweight approaches of self-assessment and non-certifying audits, [21].

A test-based assurance scheme supporting incremental security certification has been defined for Cloud systems [22]. Although focusing on a specific domain, the approach relies on a similar process of assessing the impact of changes on the target of evaluation (TOE) system (i.e. the cloud application) and also on the certification process. The goal of the proposed solution is to minimizes the risk of unnecessary certificate revocation and to reduce as much as possible the amount of re-certification activities. This is achieved through reuse of evidence available in existing certificates to re-validate them when relevant changes are observed. Although not explicitly mentioning a DevSecOps approach, this work is clearly in this spirit by replaying the test processes able to regenerate all necessary evidence that might be impacted by a change. The work relies on a sound modelling of the process based on a Certification Model (CM) template that is signed by the certification authority and which is instantiated on a specific TOE context by the accredited lab and the service provider. Different techniques of instance reduction, graph matching and annotation matching used to support the initial certification process. In order to perform dynamic and incremental certification, based on a formalised description of the change, it is possible to perform partial reevaluation and re-certification execution flows. Certification refinement through upgrade or downgrade processes are also supported. The approach could help refining our own recertification scheme although the fact that it is specific to the Cloud domain while our approach targets a larger scope might limit the applicability or level of automation. However our tight integration in a DevOps chain will definitely help raise overcome such barriers.

Incremental certification has also been considered in the safety domain, especially in avionics where it is mandatory to comply with the standards such as DO178 (for software) and DO254 (for hardware) (see [23]). A model-based approach was developed in the S@T Belgian project using a modelling approach to capture certification goals, the product using a product line model and a model of the certification process [24]. However the considered scenario is based on explicit change requests rather than dynamic evolution of the environment or discovery of vulnerabilities. While it shares some commonalities in the way to assess impact, it was not integrated in a strong DevOps automation process and was also considering less frequent and coarser grained evolution. Despite this, it highlights the benefits of model-based approach for impact assessment, the replay of specific V&V activities and the generation of incremental evidence reports. A more recent work elaborates further by considering a larger adoption of agile processes, continuous integration, delivery and deployment [23]. It stresses that standard do no impose any lifecycle and can thus consider more agile processes although the traditional V-shaped is currently the mainstream approach. It shows the rigorous activities required can be carried out in agile mode so that a certification-ready solution can be delivered at each iteration. For example, maintaining a clean traceability is possible and even more efficient using an agile process. Comparing this approach in safety with our proposal for security reveals some common principles in the technical approach but also differences in the recertification scenarios. There is also a cultural difference which needs to be addressed when considering co-engineering and co-certification which were not elaborated in the scope of the paper. However learning from the way safety is managed in an aircraft to keep the system operational in presence of failure can also inspired how to design security to avoid failing over and over again on the same kind of attack. [25] identifies five recommendations for a 'clean slate policy design' in response to the current state of verbose and hardly followed best practices. It is complemented with an incident handling and reporting structure similar to that found in aviation safety.

## 7. Conclusions

This paper has proposed to integrate Common Criteria incremental certification and DevSecOps processes in order to automate evidence gathering for product oriented certification activities. The approach was illustrated on the impact analysis of an automotive case study where firewall updates can trigger incremental certification with Common Criteria when certified cybersecurity properties are impacted. The case study was illustrated only on the impact analysis of the incremental certification process to show how the DevSecOps can provide the evidence needed for the evaluator to decide if incremental certification is needed. The other activities of the incremental certification process are left for future work. The main contributions of the paper are (1) modelling the Common Criteria incremental process and (2) composing it with a development and operations process such as DevSecOps. The expected benefits are to have a more flexible and automated incremental certification process that leads to a reduction in costs in carrying out this type of activity. A longer term expected benefit is to improve the global security posture by having a greater number of updated certified products continuously available on the market.

## Acknowledgment

This paper was supported in part by European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 830892, project "Strategic programs for advanced research and technology in Europe" (SPARTA).

## References

[1] A. Morgagni, P. Massonet, S. Dupont, and J. Grandclaudon, "Towards incremental safety and security requirements co-certification," in *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2020, pp. 79–84.

[2] N. C. S. Association, "Department of defense trusted computer system evaluation criteria," Tech. Rep. CSC-STD-001-83, 12 1985.

[3] O. for Official Publications of the European Communities, "Information technology security evaluation criteria (itsec) provisional harmonised criteria," Tech. Rep. COM(90) 314, 6 1991.

[4] C. C. portal, "Common criteria for information technology security evaluation. part 1: Introduction and general model, part 2: Security functional components and part 3: Security assurance components," Tech. Rep. CCMB-2017-04-001, CCMB-2017-04-002, 4 2017.

[5] ——, "Common criteria, assurance continuity: Ccra requirements," Tech. Rep. version 2.1, 6 2012.

[6] C. Zhou and S. Ramacciotti, "Common criteria: Its limitations and advice on improvement," *Information Systems Security Association ISSA Journal*, pp. 24–28, 2011.

[7] D. Edwards, "What is devops," *Retrieved*, vol. 3, p. 2014, 2010.

[8] J. Díaz, J. E. Pérez, M. A. Lopez-Peña, G. A. Mena, and A. Yagüe, "Self-service cybersecurity monitoring as enabler for devsecops," *IEEE Access*, vol. 7, pp. 100 283–100 295, 2019.

[9] H. Myrbakken and R. Colomo-Palacios, "Devsecops: a multivocal literature review," in *International Conference on Software Process Improvement and Capability Determination*. Springer, 2017, pp. 17–29.

[10] V. Mohan and L. Othmane, "Secdevops: is it a marketing buzzword," *Department of Computer Science, Technische Universität Darmstadt, Darmstadt*, 2016.

[11] B. Rahul, K. Prajwal, and M. Manu, "Implementation of devsecops using open-source tools," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 5, no. 3, 2019.

[12] N. MacDonald and I. Head, "Devsecops: How to seamlessly integrate security into devops," *Gartner, Tech. Rep.*, 2016.

[13] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of devsecops," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 2019, pp. 1–8.

[14] GrammaTech, "Static analysis results: A format and a protocol: Sarif & sasp," https://blogs.grammatech.com/static-analysis-results-a-format-and-a-protocol-sarif-sasp, 2018.

[15] P. S. Florian Westphal, Pablo Neira Ayuso, "Iptables 1.8.7 changelog," https://www.netfilter.org/projects/iptables/files/changes-iptables-1.8.7.txt, 2021.

[16] C. Criteria, "Assurance continuity: Ccra requirements," *Common Criteria*, pp. 0–22, 2012.

[17] S. Dupont *et al.*, "Sparta - d5.1 - assessment specifications and roadmap," https://sparta.eu/assets/deliverables/SPARTA-D5.1-Assessment-specifications-and-roadmap-PU-M12.pdf, [Online; accessed 02-June-2021].

[18] L. Volkmar, "Sparta - d11.2 - cybersecurity compliant development processes," https://www.sparta.eu/assets/deliverables/SPARTA-D11.2-Cybersecurity-compliant-development-processes-PU-M18.pdf, [Online; accessed 20-May-2021].

[19] S. V. Solms and R. V. Solms, "Incremental information security certification," *Comput. Secur.*, vol. 20, pp. 308–310, 2001.

[20] ISO, "Iso/iec 27005:2018 information technology — security techniques — information security risk management."

[21] C. Ponsard, P. Massonet, J. Grandclaudon, and N. Point, "From lightweight cybersecurity assessment to SME certification scheme in belgium," in *IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 2020, pp. 75–78. [Online]. Available: https://doi.org/10.1109/EuroSPW51379.2020.00019

[22] M. Anisetti, C. A. Ardagna, and E. Damiani, "A test-based incremental security certification scheme for cloud-based systems," in *IEEE International Conference on Services Computing*, 2015, pp. 736–741.

[23] C. Baron and V. Louis, "Towards a continuous certification of safety-critical avionics software," *Computers in Industry*, vol. 125, p. 103382, Feb. 2021.

[24] C. Ponsard *et al.*, "Smarter airborne technologies," https://www.cetic.be/SAT-1161, 2014.

[25] T. Fiebig, "How to stop crashing more than twice: A clean-slate governance approach to it security," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2020, pp. 67–74.